**android常见面试题汇总**

1. 谈谈Android系统架构？

答：分为四层：

1）应用程序层 2）应用程序框架层 3）系统运行库层 4）Linux内核层

1. 列举几个你常用的ADB命令？

答： 常用的命令有：adb shell,adb pull,adb push,adb devices,adb install

1. 列举Activity的生命周期的7个方法？

答：Activity生命周期 ：是指Android中Activity的启动到停止的全过程。

Android的生命周期由Android系统进行调度和控制。

Activity完整生命周期有7个方法，分别是onCreate() ,onRestart(),onStart(),onResume(),onPause(),onStop(),onDestroy().

1. Activity生命周期的四种状态？

答：

1、Running（运行态）

位于屏幕最前端时，此时处于可见状态，和用户可交互的状态。

2、Paused（暂停态）

当Acitivy被另一个透明的或者非全屏的Activity覆盖时的状态叫Paused状态，虽然可见但不可交互。（失去焦点状态）

3、Stop（停止态）

当Activity被另外一个Activity覆盖、界面不可见时处于Stop状态。

4、Killed（待用态）

Activity被系统杀死或者跟本没启动时就是Killed状态。

1. 简要叙述Android中的主要五种常用布局及其特点？

答：最常用的布局有以下这几种：

第一种：帧布局（框架布局）FrameLayout

第二种：线性布局LinearLayout

第三种：绝对布局AbsoluteLayout

第四种：相对布局RelativeLayout

第五种：表格布局TableLayout

1. 简述ImageView,ImageButton以及Button的区别？

答:ImageView和ImageButton都是继承于View类，Button是继承于TextView类。所以

ImageButton 不支持setText(),而Button支持

ImageButton 支持setImageURI,而Button不支持

ImageButton和Button 都有Button的状态而ImageView没有。

1. Android中常用的数据适配器有哪些？他们适用于适配什么数据？

答：我们常用的适配器一共有三个，那就是ArrayAdapter，SimpleAdapter，自定义的适配器，这三个，他们都是继承于BaseAdapter 。

对于前两个适配器，他们的数据来源无非就是String[]或者List集合。

对于自定义的适配器，如果当我们想用一些其它的展现方式，或者是我们需要的，呈现方式，这是就得自定义了。

首先我们定义一个类让它继承自BaseAdapter，让它实现里面所说的那几个方法。重要是重写里面的getView方法。

1. 在Android中，列举出几个用适配器来适配数据的组件？

答：AutoCompleteTextView,Spinner,ListView,GridView,Gallery等。

1. Android中常用的菜单的分类？以及他们的特点？

答：Android中菜单主要分为三类：选项菜单（Option Menu）,上下文菜单（Context Menu）,子菜单（Sub Menu）.

选项菜单是点击Menu按钮弹出来的菜单，最多显示6个。

上下文菜单是在某些组件上添加，长按动作发生后弹出上下文菜单。

子菜单可以创建在选项菜单和上下文菜单上。子菜单里不可以再有子菜单。

1. 谈谈对Intent的理解？

答：实现页面（Activity）跳转，作为四大组件中起“枢纽”作用的对象

Intent中文意思“意图”，在四大组件中，服务和广播接收是通过Intents（消息，意图）来激活的。

Intent分为两种：显式Intent和隐式Intent区别

1. 显式Intent和隐式Intent区别？

**答：**Intent是一种在不同组件之间传递的请求消息，是应用程序发出的请求和意图。

对于明确指出了目标组件名称的Intent，我们称之为显式Intent。

对于没有明确指出目标组件名称的Intent，则称之为隐式Intent。

Android系统使用IntentFilter 来寻找与隐式Intent相关的对象。

显式Intent直接用组件的名称定义目标组件，这种方式很直接。但是由于开发人员往往并不清楚别的应用程序的组件名称，因此，显式Intent更多用于在应用程序内部传递消息。比如在某应用程序内，一个Activity启动一个Service。

隐式Intent恰恰相反，它不会用组件名称定义需要激活的目标组件，它更广泛地用于在不同应用程序之间传递消息。

1. 谈谈Service的生命周期？

答：Service的生命周期由使用Service两种方式决定，启动方式不同生命周期不同。

Start方式启动Service的生命周期:

onCreate()—onStartCommond()—onDestroy()

调用者和服务之间没有任何的联系，即使调用者退出，那么服务也仍然进行。

Bind方式启动Service的生命周期：onCreate()->onBind()->-->onUnbind() -> onDestroy()

调用者和服务绑定在一起，调用者退出，服务即便退出。

1. 谈谈对Service的理解？

答：[服务](file:///D:\android\android-sdk\docs\reference\android\app\Service.html)是一个应用程序组件，可以在后台执行长时间运行的操作，不提供用户界面。

一项服务“开始”，当一个应用程序组件（如一个活动）开始通过调用[的StartService（）](file:///D:\android\android-sdk\docs\reference\android\content\Context.html#startService(android.content.Intent))

“绑定”一个服务，当一个应用程序组件绑定它通过调用[bindService（）](file:///D:\android\android-sdk\docs\reference\android\content\Context.html#bindService(android.content.Intent, android.content.ServiceConnection, int))

1. 启动服务有几种方式？他们之间的区别是什么？

答：启动服务有两种方式。

一种是通过context.startService() 来启动Service

其生命周期为context.startService() ->onCreate()- >onStart()->Service running-->context.stopService()->onDestroy() ->Service stop

调用者和服务之间没有任何的联系，即使调用者退出，那么服务也仍然进行。

另外一种是通过使用bindService()启动Service

其生命周期为：context.bindService()->onCreate()->onBind()->Service running-->onUnbind() -> onDestroy() ->Service stop

调用者和服务绑定在一起，调用者退出，服务即便退出。

1. AIDL全称是什么？谈谈AIDL适用的场景？

答：AIDL全称：Android Interface Define Language安卓接口定义语言。

主要适用于跨应用的服务的访问和调用。

1. 广播接收器的全称？广播接收器的作用？

答：广播的全称是：BroadcastReceiver 广播接收器。

BroadcastReceiver的作用：处理Android中发出的广播事件。

1. 有关广播的注册方式、优缺点，以及引用广播机制的用意。

答：注册广播共有两种方式,一种是通过在manifest.xml文件中注册,不需要手动进行注销。先在XML文件中添加广播注册，然后再在JAVA篇码段中添加Intent intent = new Intent("test");

sendBroadcast(intent);

另一种直接在代码中通过代码注册（有点类似如隐式Intent）;需要通过手动进行注销(unregisterReceiver(receiver);

第一种方式: 不需要启动程序也可以监听广播但有时显得不灵活；

第二种方式：需要启动程序代码才可以监听，有时比较灵活，如当希望在某个Activity启动时监听广播，当这Activity死亡时不监听，即可用这种方法。

BroadcastReceiver的用意：处理Android中发出的广播事件。

1. Android四大组件的作用？android用四大组件的用意何在？(ContentProvider可以不用描述)

答：Android中四大核心组件分别是：Activity,Service,BroadCastReceiver,ContentProvider。

Activity 是UI界面显示，它是Android程序与用户交互的窗口，是Android构造块中最基本的一种，它需要为保持各界面的状态，做很多持久化的事情，妥善管理生命周期以及一些跳转逻辑。

Service（后台运行的服务）一种长生命周期的、可以理解为没有界面的activity，

后台服务于Activity，封装有一个完整的功能逻辑实现，接受上层指令，完成相关的事务，定义好需要接受的Intent提供同步和异步的接口。

BroadcastReceiver 广播接收器（做广播，通知时候用到）是为了处理Android中发出的广播事件的组件；

（接受一种或者多种Intent作触发事件，接受相关消息，做一些简单处理，转换成一条Notification，统一了Android的事件广播模型。）

Content Provider 数据通信（组件之间通信）实现不同组件之间数据的共享

（是Android提供的第三方应用数据的访问方案，可以派生Content Provider类，对外提供数据，可以像数据库一样进行选择排序，屏蔽内部数据的存储细节，向外提供统一的接口模型，大大简化上层应用，对数据的整合提供了更方便的途径。 ）

1. **谈谈Activity与Service的区别**

**a. Activity**

Activity是用户和应用程序交互的窗口，一个Activity是一个应用程序组

件，提供一个屏幕，用户可以用来交互为了完成某项任务例如拨号、拍照、

发送email、看地图。每一个activity被给予一个窗口，在上面可以绘制用

户接口。窗口通常充满屏幕，但也可以小于屏幕而浮于其它窗口之上。一个

activity相当于我们实际中的一个网页，当打开一个屏幕时，之前的那一个

屏幕会被置为暂停状态，并且压入历史堆栈中，用户可以通过回退操作返回

到以前打开过的屏幕。activity的生命周期：即“产生、运行、

销毁”，但是这其中会调用许多方法onCreate（创建） 、onStart（激活） 、

onResume

（恢复） 、onPause（暂停） 、onStop（停止） 、onDestroy（销毁） 、

onRestart

（重启）。

**b.Service**

Service是运行在后台的一段代码，它可以运行很长的时间，相当于后台的一个服务。它可以运行在它自己的进程,也可以运行在其他应用程序进程的上下文（context）里面，这取决于自身的需要。其它的组件可以绑定到一个服务（Service）上面，通过远程过程调用（RPC）来调用这个方法。例如媒体播放器的服务，当用户退出媒体选择用户界面，仍然希望音乐依然可以继续播放，这时就是由服务 （service）来保证当用户界面关闭时音乐继续播放的。

1. Intent 能传递什么数据类型？

a.简单或者基本数据类型

b.传递一个Bundle

c.传递Serializable对象

d.Parcelable对象

e.Intent

1. **当一个Activity在虚拟器上运行起来以后，点击home键，Activity会执行什么生命**

**还走起方法？ 当再次进入这个应用，Activity执行什么生命周期方法？**

点击home键，会执行onPouse()----onStop() 当再次打开应用，会执行

onRestart()---onStart()----onResume()

1. **SharedPreferences能存储什么数据类型？**

Shared Preferences：用来存储"key-value"paires格式的数据，它是一个

轻量级的键值存储机制，只可以存储基本的数据类型。如 String 、int 、

boolean 、float 、double、long 、byte、short

1. **Android中的Notification一般会在什么时候用到？**

Notification就是在桌面的状态通知栏。这主要涉及三个主要类：

Notification：设置通知的各个属性。NotificationManager：负责发送通

知和取消通知Notification.Builder：Notification内之类，创建

Notification对象。非常方便的控制所有的flags，同时构建Notification

的风格。

主要作用：

a.创建一个状态条图标

b.在扩展的状态条窗口中显示额外的信息（和启动一个Intent）。

c.闪灯或LED。

d.电话震动。

e.发出听得见的警告声（铃声，保存的声音文件

Notification是看不见的程序组件（Broadcast Receiver，Service和不活

跃的Activity）警示用户有需要注意的事件发生的最好途径

Android高级技术-简答题

**1：Android中五种数据存储方式分别是什么？他们的特点？**

**答：**Android提供了五种存取数据的方式

1. SharedPreference，存放较少的五种类型的数据，只能在同一个包内使

用，生成XML的格式存放在设备中

(2) SQLite数据库，存放各种数据，是一个轻量级的嵌入式数据库

(3) File文件，通过读取写入方式生成文件存放数据

(4) ContentProvider，主要用于让其他应用程序使用保存的数据

(5)通过网络获取数据和写入数据到网络存储空间

**2:简述SharedPreferences存储方式**

SharedPreferences也是一种轻型的数据存储方式，它的本质是基于XML文件存储key-value键值对数据，通常用来存储一些简单的配置信息。其存储位置在/data/data/<包名>/shared\_prefs目录下。SharedPreferences对象本身只能读取数据而不支持写入数据，存储修改是通过Editor对象实现。

**3：SQLite主要通过那两个类实现其功能的？**

SQLiteOpenHelper：这个类是数据库的帮助类；用来进行数据库的创建；表的创建，以及版本的更新

SQLiteDatabase： 这个类里 封装了对表的具体的操作的方法，比如 增删改查的方法。

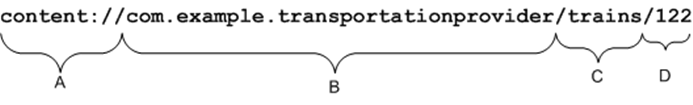
**4：SharedPreferences与SQLite数据库的区别**

SharedPreferences对象与SQLite数据库相比，免去了创建数据库，创建表，写SQL语句等诸多操作，相对而言更加方便，简洁。但是SharedPreferences也有其自身缺陷，比如其职能存储boolean，int，float，long和String五种简单的数据类型，比如其无法进行条件查询等。所以不论SharedPreferences的数据存储操作是如何简单，它也只能是存储方式的一种补充，而无法完全替代如SQLite数据库这样的其他数据存储方式。

**5：ContentProvider与SQLite有何区别**

答：Sql只能在该工程的内部共享数据，ContentProvider能在工程之间实现数据共享。

**6：描述ContentProvider URI有哪几部分组成**

答：URI由是部分组成：

A，标准的前缀: ContentProvider（内容提供者）的scheme已经由Android所规定， scheme为：content://

B，唯一标识整个Content Provider: 主机名（或叫Authority）用于唯一标识这个ContentProvider，外部调用者可以根据这个标识来找到它。

C，描述了数据的路径，确定返回哪类数据: 路径（path）可以用来表示我们要操作的数据，路径的构建应根据业务而定

1. ID唯一标注请求的数据: 我们要请求数据的ID

**7：简述通过contentResolver获取ContentProvider内容的基本步骤**

第一：得到ContentResolver类对象：ContentResolver cr = getContentResolver（）；

第二：定义要查询的字段String数组。

第三：使用cr.query();返回一个Cursor对象。

第四：使用while循环得到Cursor里面的内容。

**8：描述一下自定义ContentProvider的步骤**

**答：**自定义contentprovider步骤：

1. 继承Android的ContentProvider基类实现自己的ContentProvider类。
2. 实现 ContentProvider里的onCreate方法，创建provider的数据，一般为数据库sqlite。
3. 重写ContentProvider 中相应的insert/delete/update/query增删改查这些方法。
4. 在AndroidManifest.xml文件中注册该ContentProvider类，为ContentProvider指定Uri。Uri类似网页访问中使用的URL地址。

在<application></application>中配置 <provider/>标签以及属性：android:name属性：ContentProvider的类名

android:authorities属性：用于访问ContentProvider的名称。

**9：Android中如何访问自定义ContentProvider?**

通过ContentProvider的Uri访问开放的数据。

(1)ContenResolver对象通过Context提供的方法getContenResolver()来获得。

(2)ContenResolver提供了以下方法来操作：insert delete update query这些方法分别会调用ContenProvider中与之对应的方法并得到返回的结果。

**10：android中的动画有哪几类，它们的特点和区别是什么？**

答：Android中动画可以分为两大类：帧动画、补间动画

1）补间动画：你定义一个开始和结束，中间的部分由程序运算得到。主要有：旋转、平移、放缩和渐变；AlphaAnimation（渐变型动画）、scaleAnimation（缩放型动画）、 TranslateAnimation（平移型动画）、 RotateAnimation（旋转型动画）、

2）逐帧动画：Frame（把一连串的图片进行系列化连续播放，如同放电影的效果），它是通过播放一张一张图片来达到动画的效果；

**11：如何实现一个帧动画?**

答：

1)在布局文件中添加ImageView组件，设置background为res/anim/fram.xml动画效果

2)通过ImageView对象的getBackground()方法得到AnimationDrawable对象

3)通过AnimationDrawable对象的start()来启动帧动画，stop()来停止动画

**12：简述你所学的音乐播放的方式，以及区别。**

1）：MediaPlayer：主要用于播放音频，可以播放视频，但是一般不用其进行视频播放。

2）：SoundPool： 主要用于播放一些短促的声音片段，主要优势是cpu资源占有量低，反应延迟小。

**13：简述soundpool的使用步骤。**

使用SoundPool播放声音的步骤如下。

(1)调用SoundPool的构造器创建SoundPool对象。

(2)调用SoundPool对象的load方法加载声音资源。

(3)调用SoundPool对象的play方法播放声音。

**14：简述你学过Android中视频播放的方式？**

答：

1：VideoView android提供给我们的一个供我们进行视频播放的组件

2：surfaceView+ MediaPlayer

3：系统提供的播放器

**15：简述VideoView实现视频播放的基本步骤**

使用VideoView播放视频

步骤一：\* 在onCreate方法中，首先获取布局管理器中添加的VideoView组件。并创建一个要播放视频所对应的对象。

步骤二：\* 然后创建一个MediaController对象，用于控制视频的播放。

步骤三：\*使用VideoView播放该视频

**16：简述**surfaceView+ MediaPlayer**实现视频播放的基本步骤**

步骤一：在布局文件内定义SurfaceView，

步骤二：在代码中实现SurfaceHolder.Callback接口，重写三个方法

步骤三：通过方法getHolder();得到SurfaceHolder，然后设置SurfaceHolder

步骤四：为MediaPlayer设置SurfaceHolder

步骤五：使用setDataSource方法为MediaPlayer设置播放文件

步骤六：进行播放

**17：res/raw与assets目录的区别**

答：

相同点：两者都会原封不动的保存在apk包中，不会被编译成二进制码。

不同点：

1.raw目录下只能存放文件，不能存放下一级的文件夹，而assets可以存放下一级的文件夹。

2.raw目录下的资源会映射到R.java中生成资源id。而assets不会。

3.获得资源的方法不同。代码如下：

this.getAssets().open( "xml/channels.xml")

this.getResources() .openRawResource(R.raw.students)

this.getClassLoader().getResourceAsStream(“student.xml”)

18:简述Timer的用法

Timer是在一个应用运行期间进行的定时操作，当应用退出则定时操作也同时被取消。

使用Timer的步骤：

1：实例化Timer timer = new Timer（）；

2：实例化TimerTask t = new TimerTask(){

@Override

public void run() {

// TODO Auto-generated method stub

要执行的内容

}};

3: 调用：timer.schedule(t, 5000);执行

19：简述Handler机制原理

答：Andriod提供了 Handler 和 Looper 来满足线程间的通信。

Android不允许UI线程之外的线程改变UI组件的值，所以要想在其他线程里面改变UI组件的值，必须使用Handler来实现。

1)Looper: 一个线程可以产生一个Looper对象，由它来管理此线程里的Message Queue(消息队列)。

2)Handler: 你可以构造Handler对象来与Looper沟通，以便push(上传)新消息到Message Queue里;或者接收Looper从Message Queue取出的消息。

3) Message Queue(消息队列):用来存放线程放入的消息。

4)线程：UI thread 通常就是main thread，而Android启动程序时会替它建立一个Message Queue。

Handler、Looper、MessageQueue的初始化流程如下所示。

　　Hander持有对UI主线程消息队列UI\_MQ和消息循环Looper的引用，子线程可以通过Handler将消息发送到UI线程的消息队列UI\_MQ中。

Handler处理消息

　　UI主线程通过Looper循环查询消息队列UI\_MQ，当发现有消息存在时会将消息从消息队列中取出。首先分析消息，通过消息的参数判断该消息对应的Handler，然后将消息分发到指定的Handler进行处理。

20：ListView的优化方案？

答：

1、**判断contentView是否为空：**如果自定义适配器，那么在getView方法中要考虑方法传进来的参数contentView是否为null，如果为null就创建contentView并返回，如果不为null则直接使用。在这个方法中尽可能少创建view。

2、**使用viewHolder缓存数据：**给contentView设置tag（setTag（）），传入一个viewHolder对象，用于缓存要显示的数据，可以达到图像数据异步加载的效果。

**3、分页加载：**如果listview需要显示的item很多，就要考虑分页加载。比如一共要显示100条或者更多的时候，我们可以考虑先加载20条，等用户拉到列表底部的时候再去加载接下来的20条。分页加载还没学习。

**21:MVC设计模式在Android中的应用？**

答：

1.模型层（model）:对数据库操作，对网络等操作都应在model中处理，对业务的计算等操作都应在该层。可以简单的理解为Android中的Java源文件

2.视图层（view）:一般采用XML进行描述，使用的时候可以方便的引入，可以简单的理解为Android中Layout文件。 Android中也可以用JavaScript+HTML 等方式作为view 层。

3.控制层（controller）:控制model和view的协调工作。简单的理解为Android中manifest文件.

**22:如何实现Sensor编程?**

答：

a.获取系统服务（SENSOR\_SERVICE)返回一个SensorManager 对象

sensormanager = (SensorManager)getSystemSeriver(SENSOR\_SERVICE);

b.通过SensorManager对象获取相应的Sensor类型的对象

sensorObject = sensormanager.getDefaultSensor(sensor Type);

c.声明一个SensorEventListener 对象用于侦听Sensor 事件，并重载

onSensorChanged方法

SensorEventListener sensorListener = new SensorEventListener(){};

d.注册相应的SensorService

sensormanager.registerListener(sensorListener, sensorObject, Sensor

TYPE);

e.销毁相应的SensorService

sensormanager.unregisterListener(sensorListener, sensorObject);

**23:什么是ANR 如何避免它**

答：Application not Responding(程序无应答)

1)主线程中尽量少做操作

2)对繁琐的功能代码分段完成

3)潜在的耗时操作，例如网络或数据库操作，或者高耗时的计算如改变位图尺寸，应该在子线程里（或者以数据库操作为例，通过异步请求的方式）来完成。

**1. 什么是WebView?**

WebView是一个使用WebKit引擎的浏览器组件，用来加载网页。

**2. WebView中加载网页的两种方式?**

loadUrl(url) 通过url地址加载互联网或本地网页

如：loadUrl("http://www.baidu.com");

loadUrl("file:///android\_asset/html/index.html");

loadData(data,type,encoding) 直接加载网页内容，容易乱码，一般用 [loadDataWithBaseURL](http://localhost/android-new/reference/android/webkit/WebView.html#loadDataWithBaseURL(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String))代替

**3. WebView的三个辅助类是什么?**

WebSettings、WebViewClient、WebChromeClient

**4. WebSettings的作用是什么?**

设置WebView的一些属性、状态等,例如允许使用javascript，允许使用缓存，允许使用内置的缩放组件

setJavaScriptEnabled(true); 设置支持js

**5. WebViewClient的作用是什么?**

主要帮助WebView处理各种通知、请求事件（例如，点击链接时候如何显示界面，页面开始加载，加载完毕之后有何动作等）

shouldOverrideUrlLoading() onPageStarted() onPageFinished()

**6. WebChromeClient的作用是什么？**

辅助WebView处理Javascript的对话框、网站图标、网站Title、加载进度等

核心方法有onJsAlert() onJsConfirm() onJsPrompt() onProgressChanged() onReceivedIcon()

onReceivedTitle()

**7. WebViewClient中shouldOverrideUrlLoading方法作用**

shouldOverrideUrlLoading方法控制超连接是否在当前WebView中打开，返回false表示当前WebView会处理此超链接，返回true则表示宿主应用程序处理此超链接。

（详见api ：return true means the host application handles the url, while return false means the current WebView handles the url.）

**8. android中如何调用js中的方法？**

loadUrl("javascript:test()");

**9. 页面js中调用android客户端的方法实现步骤？**

1,添加javascript支持（ webSettings.setJavaScriptEnabled(true);）

2,暴露一个java对象给js，使得js可以直接调用java方法，

（webview.addJavascriptInterface(object,"name");）

3,加载页面（ webview.loadUrl(url);}

4,页面可以通过别名使用（window.name.方法名）

**10. Socket编程的两种协议？**

TCP (Transmission Control Protocol ) 传输控制协议

UDP (User Datagram Protocol) 用户数据报协议

**11. TCP与UDP的区别。**

TCP---传输控制协议,提供的是面向连接、可靠的字节流服务,传输数据前经过“三次握手”建立连接，保证数据传输的可靠性，但效率比较低。一般用于对于数据传输安全性较高的场合。

UDP---用户数据报协议，是一个简单的面向数据报的运输层协议，面向无连接。UDP不提供可靠性，数据传输可能发生错序，丢包，但效率较高。一般用于对于实时性要求较高的场合。

**12. Http编程的两种实现方式?( 网络互连的两种实现方式？)**

HttpURLConnection(java.net.\*) , HttpClient (org.apache.http.\*)

**13. HttpClient实现移动通信的步骤？**

1.生成请求对象（HttpGet get，HttpPost post）

2. 设置连接参数，连接超时，读取超时(HttpParams)

3.生成客户端对象 HttpClient client

4.执行请求接收响应，判断响应码 HttpResponse response = client.execute(post)

HttpEntity entity = response.getEntity()

5.得到数据流

InputStream inputStream = entity.getContent();

6.最后关闭过期连接

**14. 什么是ANR**

ANR：Application Not Responding(应用程序无响应).当在某个应用主线程出现下列情况时，就会显示ANR对话框了：用户在进行了一种操作后5秒钟没有响应，BroadcastReceiver所进行的操作在10秒内没有完成，Service在20秒内没返回结果。这就意味着，任何在主线程中运行的，需要消耗大量时间的操作都会引发ANR，比如访问网络，SD卡读写， Sqlite数据库增删改查等。

**15. 如何避免ANR （线程间通讯的两种方式是什么）**

1. AsyncTask，其中doInBackground()和onPostExecute(Result)两个方法非常重要

doInBackground() 这个方法运行在后台线程中，主要负责执行那些很耗时的操作，如访问网络。该方法必须重载。

onPostExecute(Result) 这个方法运行于UI主线程，在doInBackground(Params…)方法执行后调用，该方法用于接收后台任务执行后返回的结果，刷新UI显示。

2. 子线程 + handler

在子线程中需要刷新UI时通过handler.sendMessage()发消息给主线程， 然后在主线程Handler中的handleMessage()方法中执行刷新UI操作

**16. AsyncTask三个泛型类型是什么意思。**

第一个泛型类型Params----规定doInBackground()方法的参数类型，也是执行异步任务excute()方法的参数类型

第二个泛型类型Progress----规定onProgressUpdate()方法的参数类型，一般为Integer，表示进度

第三个泛型参数类型Result---规定onPostExcute()方法的参数类型，也是doInBackground()方法的返回值类型。

**17. 谈谈AsyncTask中两个最重要的方法是什么，运行的特点。**

doInBackground() 这个方法运行在后台线程中，主要负责执行那些很耗时的操作，如访问网络。该方法必须重载。

onPostExecute(Result) 这个方法运行于UI主线程，在doInBackground(Params…)方法执行后调用，该方法用于接收后台任务执行后返回的结果，刷新UI显示。

**18. Handler的运行机制(运行原理)(Handler,Looper,MessageQueue,Message之间的关系)**

一个Handler允许你发送和处理Message和Runable对象，每个线程都有自己的Looper，每个Looper中封装着MessageQueue。Looper负责不断的从自己的消息队列里取出队头的任务或消息执行。每个handler也和线程关联，Handler负责把Message和Runable对象传递给MessageQueue（用到post ，sendMessage等方法），而且在这些对象离开MessageQueue时，Handler负责执行他们（用到handleMessage方法）。   
其中Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象提供额外的两个int域和一个Object域。

**19. 移动互联数据交互格式有哪些及其区别？（Json与xml的区别？）**

移动互联数据交互格式有XML和JSON

1.JSON和XML的数据可读性基本相同

2.JSON和XML同样拥有丰富的解析手段

3.JSON相对于XML来讲，数据的体积小

4.JSON与JavaScript的交互更加方便

5.JSON对数据的描述性比XML较差

6.JSON的速度要远远快于XML

**20. XML解析有哪几种方式？**

基本的解析方式有三种: DOM,SAX,Pull

**21. XML解析各种方式的优缺点？**

dom解析解析器读入整个文档，然后构建一个驻留内存的树结构，然后就可以使用 DOM 接口来操作这个树结构。优点是对文档增删改查比较方便，缺点占用内存比较大。

sax解析基于事件驱动型,优点占用内存少，解析速度快，缺点是只适合做文档的读取，不适合做文档的增删改，不能中途停止。

pull解析同样基于事件驱动型,android 官方API提供,可随时终止,调用next() 方法提取它们（主动提取事件）

**22. android官方推荐使用的解析XML的方式是？**

PULL解析

**23. PULL解析的5个事件类型是什么（eventType）**

START\_DOCUMENT END\_DOCUMENT START\_TAG END\_TAG TEXT

**24. SAX解析的5个方法是什么**

startDocument endDocument startElement endElement characters

**25. JSON的结构？**

json是一种轻量级的数据交换格式，

json简单说就是javascript中的对象和数组，所以这两种结构就是对象和数组两种结构，通过这两种结构可以表示各种复杂的结构

1、对象：对象在js中表示为“{}”扩起来的内容，数据结构为 {key：value,key：value,...}的键值对的结构，在面向对象的语言中，key为对象的属性，value为对应的属性值，所以很容易理解，取值方法为 对象.key 获取属性值，这个属性值的类型可以是 数字、字符串、数组、对象几种。

2、数组：数组在js中是中括号“[]”扩起来的内容，数据结构为 ["java","javascript","vb",...]，取值方式和所有语言中一样，使用索引获取，字段值的类型可以是 数字、字符串、数组、对象几种。

经过对象、数组2种结构就可以组合成复杂的数据结构了。

**26. 解析JSON的两种方式。**

1，SDK提供JSONArray，JSONObject

2，google提供的 Gson

通过fromJson()实现对象的反序列化（即将json串转换为对象类型）

通过toJson()实现对象的序列化 （即将对象类型转换为json串）

**27. 通过google提供的Gson解析json时，定义JavaBean的规则是什么？**

1). 实现序列化 Serializable

2). 属性私有化，并提供get，set方法

3). 提供无参构造

4). 属性名必须与json串中属性名保持一致 （因为Gson解析json串底层用到了反射机制）

**28. ListView优化策略？**

主要是针对自定义BaseAdapter中getView（）方法的优化

1)、convertView复用，对convetView进行判空，当convertView不为空时重复使用，为空则初始化，从而减少了很多不必要的View的创建

2)定义一个ViewHolder，封装Listview Item条目中所有的组件，将convetView的tag设置为ViewHolder,不为空时通过ViewHolder的属性获取对应组件即可

3)、当ListView加载数据量较大时可以采用分页加载和图片异步加载

**29. ListView实现图片异步加载思路？**

可定义图片异步加载工具类，核心方式实现思路如下：

1.先从内存缓存（Map<String,SoftReference<Bitmap>>中获取图片显示

2.获取不到的话从本地SD卡里获取并显示

3.都获取不到的话通过子线程从网络加载图片并保存到内存及SD卡中并通过handler显示

**30. webService有什么优点**

Webservice就是为应用提供数据的Web服务。它的优点有：

1，基于标准 2，独立于语言及平台(即跨平台和语言) 3，松耦合

**31. webService中SOAP是指什么**

SOAP是web service的标准通信协议，SOAP为simple object access protocol的缩写，简单对象访问协议. 它是一种轻量的、简单的、基于 XML 的协议即一种标准化的传输消息的XML消息格式

**32. webService中WSDL是指什么**

WSDL的全称是web service Description Language,是一种基于XML格式的关于web服务的描述语言。其主要目的在于web service的提供者将自己的web服务的所有相关内容,如所提供的服务的传输方式，服务方法接口，接口参数，服务路径等，生成相应的完全文档，发布给使用者。

**33. webService中UDDI是指什么?**

UDDI 是一种[目录服务](http://baike.baidu.com/view/1351410.htm)，企业可以使用它对 Web services 进行注册和搜索。UDDI，英文为 "Universal Description, Discovery and Integration"，可译为“通用描述、发现与集成服务”。

**34. WebService中传输的数据什么格式？**

目前大部分WebService通过xml格式组织数据，而json格式可能成为将来WebService传输数据的一种趋势。

**35. GPS定位与网络定位有何区别？**

GPS定位需设置权限android.permission.ACCESS\_LOCATION\_FINE

网络定位需设置权限android.permission.ACCESS\_LOCATION\_COARSE

即GPS定位比网络定位在精确度方面更精准，但在速度方面稍差一些，而且GPS定位适合在室外环境下使用，网络定位在室内，室外都可以。

**36. 百度定位的主要过程是？**

1，实例化定位客户端对象（LocationClient）

2，注册监听（registerLocationListener(BDLocationListener)）

3，设置定位参数（LocationClientOption）

4，请求定位（requestLocation()）

5，在监听类中处理结果（监听类的onReceiveLocation方法会接收到BDLocation结果）

**37. 使用百度地图的过程**

1,注册百度开发者账号

2,申请Key （2.1.3版本前后申请key的方式不一样）

3,从开发者中心官网上下载相关的jar包so文件和demo（百度地图与定位采用分离的sdk）

4,在自己工程中导入相关的jar包和so文件。

5,加入地图相关的权限

6,按照百度地图API的规则书写代码.

**38. 展示百度地图基本图层涉及到的核心类有哪些？**

BMapManager 地图引擎管理类

MapView 显示地图的View

**39. 百度地图中搜索功能最重要的类和接口是**

MKSearch MKSearchListener

**40. 解释一下正向地理编码和反向地理编码（地址解析和逆地址解析）**

正向地理编码是指将通过地址检索地理坐标点，即经纬度

反向地理编码是指通过坐标点检索详细地址

**41. 介绍一下百度地图中常用的覆盖图层**

我的位置图层（MyLocationOverlay）：用于显示用户当前位置的图层（支持自定义位置图标）

Poi搜索结果图层（PoiOverlay）：用于显示兴趣点搜索结果的图层；

自定义图层（ItemizedOverlay）：可将一个或多个兴趣点绘制到地图上，且支持自定义图标（支持动态更新Item位置、图标）

注：如果需要给覆盖图层添加点击事件，可重写对应覆盖图层的onTap() 方法

（5分）Android中如何实现一键退出？

答：定义一个类继承Application，定义一个集合存放所有的activity， 定义一个添

加的方法，再写一个退出的方法，使用for循环全部调用finish方法，然后在每个

Activity的onCreate方法中调用自定义类里的添加方法，然后在需要使用一键退出

的地方调用类中的退出方法即可。

（5分）谈谈MVC在Android中的应用？

答: Android中界面部分也采用了当前比较流行的MVC框架。

在Android中：

1) 视图层（View）：一般采用XML文件进行界面的描述，使用的时候可以非常方

便的引入。也可以使用JavaScript+HTML等的方式作为View层，通过WebView组

件加载,同时可以实现Java和JavaScript之间的通信。

2) 控制层（Controller）：这句话也就暗含了不要在Acitivity中写代码，要通过Activity

交割Model业务逻辑层处理，这样做的另外一个原因是Android中的Acitivity的响

应时间是5s，如果耗时的操作放在这里，Android的控制层的重任通常落在了众多

的Acitvity的肩上，程序就很容易被回收掉。

1. 模型层（Model）：对数据库的操作、对网络等的操作都应该在Model里面处理，

当然对业务计算等操作也是必须放在的该层的。  
 在Android SDK中的数据绑定，也都是采用了与MVC框架类似的方法来显示数据。在控制层上将数据按照视图模型的要求（也就是Android SDK中的Adapter）封装就可以直接在视图模型上显示了，从而实现了数据绑定。比如显示Cursor中所有数据的ListActivity，其视图层就是一个ListView，将数据封装为ListAdapter，并传递给ListView，数据就在ListView中显示。

1. 灵活运用Java编程规范进行项目开发

（4分多选）下列有关开发中，遵循的编程规范中，描述正确的有：（ABCD）

1. 一 个唯一包名的前缀总是全部小写的ASCII字母并且是一个顶级域名，通常

是com，edu，gov，mil，net，org，或1981年ISO 3166标准所指定的标识国家的英文双字符代码

B.类名是个一名词，采用大小写混合的方式，每个单词的首字母大写。尽量使

你的类名简洁而富于描述。使用完整单词，避免缩写词(除非该缩写词被更广

泛使用，像URL，HTML)

C.方法名是一个动词，采用大小写混合的方式，第一个单词的首字母小写，其

后单词的首字母大写

D.所有实例，包括类，类常量，均采用大小写混合的方式，第一个单词的首字

母小写，其后单词的首字母大写。变量名不应以下划线或美元符号开头，尽管

这在语法上是允许的。

1. 掌握企业项目的开发流程

（简单10分）简述一下企业项目的开发流程？

答：项目的开发流程：立项，需求，设计（规模），开发，测试，上线，维护。

立项：主要确定软件的开发目标及其可行性

需求：对软件需要实现的各个功能进行详细分析

软件设计：对整个软件系统进行设计，软件设计一般分为总体设计(原型设计

Dreamweaver )和详细设计(类设计Rational Rose (IBM))。如原型设计Acure，系统

框架设计，数据库设计( PowerDesigner)等等。

开发:进行功能的实现与调试

测试：发现软件在整个设计过程中存在的问题并加以纠正（整个测试过程分单元测

试、组装测试以及系统测试三个阶段进行。测试的方法主要有白盒测试和黑盒测试

两种。）

上线：软件交付使用，试运行

软件维护：软件维护是软件生命周期中持续时间最长的阶段。

1. 了解软件生命周期模型以及类别

（简答5分）软件生命周期模型有哪些？

答：软件开发模型是指软件开发全部过程、活动和任务的结构框架。软件开发包括

　　需求、设计、编码和测试等阶段，有时也包括维护阶段。 软件开发模型能清 晰、直观地表达软件开发全过程，明确规定了要完成的主要活动和任务，用来

作为软件项目工作的基础。 常见的模型有：瀑布模型、迭代模型、快速原型

模型。

1. 掌握Fragment的使用

（10分）Activity的 onSaveInstanceState() 和 onRestoreInstanceState()区别？

答: Activity的 onSaveInstanceState() 和 onRestoreInstanceState()并不是生命周期方

法，它们不同于 onCreate()、onPause()等生命周期方法，它们并不一定会被触发。

当应用遇到意外情况（如：内存不足、用户直接按Home键）由系统销毁一个Activity

时，onSaveInstanceState() 会被调用。但是当用户主动去销毁一个Activity时，例如

在应用中按返回键，onSaveInstanceState()就不会被调用。因为在这种情况下，用户

的行为决定了不需要保存Activity的状态。通常onSaveInstanceState()只适合用于保

存一些临时性的状态，而onPause()适合用于数据的持久化保存。

另外，当屏幕的方向发生了改变， Activity会被摧毁并且被重新创建，如果你

想在Activity被摧毁前缓存一些数据，并且在Activity被重新创建后恢复缓存的数

据。可以重写Activity的 onSaveInstanceState() 和 onRestoreInstanceState()方法。

1. 掌握ViewPager的使用

（10分）ViewPager、ViewFlipper与ViewFlow之间的差异

1. ViewPager与ViewFlow都能够使用适配器（ViewPager--PagerAdapter；

ViewFlow--BaseAdapter）进行大量数据的适配，并且ViewFlow也带有原点和标题的位置提示，二者比较相像；

2、ViewFlipper

使用时主要在有限的少数页面切换中比较合适，并且能够自定义每一个切换动画，

用于一个应用间的画面切换比较合适，类似于ActvityGroup；

1. ViewFlow由于提供源码，所以在扩展上更强，可根据需要自行定制，比如加入

循环播放等。

1. 掌握ActionBar的用法
2. 掌握侧滑菜单的实现

(10分)掌握侧滑菜单的实现？

答：实现侧滑的步骤：

　　要实现ViewPager+Fragment的滑动效果，必须继承FragmentActivity，同时要

给ViewPager设置Adapter。由于ViewPager上面要加载Fragment（需要继承Fragment

类），所以不是重载PagerAdapter,而是重写FragmentPagerAdapter类

　　写好了adapter，在主函数的onCreate（）函数里面初始化ViewPager,同时给

它配置adapter和设置滑动事件监听（这需要实现OnPageChangeListener接口）

　　接下来要做的就是在onCreate()里面初始化RadioButton，并给它设置

OnClickListener监听点击事件

最后就是实现侧滑出左边页面的效果，这里使用了开源组件SlidingMenu，

SlidingMenu组件重写了Activity、FragmentActivity、RelativeLayout等，要实现侧滑

效果，可以把主函数继承的FragmentActivity改成SlidingFragmentActivity，也可以

去在xml布局文件里面直接设置

1. 掌握“Splash动画”实现的几种思路

（10分）Android中的动画API,及项目中的应用

答: Android中有两种动画, 帧动画和补间动画,补间动画又有平移,旋转,缩放,渐变

动画。3.0之后又有新特性属性动画，项目中使用场景: 项目启动向导动画, Activity

切换时动画效果(overriedPendingTransaction()),类似Tab标签页选中项背景的切换,

Fragment切换时动画效果设置,ViewPager翻页时动画效果

1. 灵活掌握ListView优化机制

（10分）listview优化策略？

答：1)、自定义Adapter的getView(), 对convetView进行判空，是当convertView

不为空的时候直接重新使用convertView，从而减少了很多不必要的View的创建

1. 定义一个ViewHolder，将convetView的tag设置为ViewHolder,不为空时重新使用

即可

3)、当ListView加载数据量较大时可以采用分页加载和图片异步加载

1. 灵活掌握ListView分页加载实现思路(“加载更多”与自动加载两种)

（15分）ListView分页加载实现思路？

答：通常实现分页加载有两种方式，一种是在ListView底部设置一个按钮，用户点

击即加载。另一种是当用户滑动到底部时自动加载。

在ListView底部设置一个按钮，用户点击即加载实现思路:

1. 灵活掌握 ListView图片异步加载实现思路,可使用开源框架ImageLoader

（15分）ListView图片异步加载实现思路？

1.先从内存缓存中获取图片显示（内存缓冲）

2.获取不到的话从SD卡里获取（SD卡缓冲，，从SD卡获取图片是放在子线程里执行的，否则快速滑屏的话会不够流畅）

3.都获取不到的话从网络下载图片并保存到SD卡同时加入内存并显示（视情况看是否要显示）

1. 掌握自定义组件实现思路总结

（10分）简述自定义组件的实现思路？

答: Android自定义组件有三种实现思路:

1) 继承某个现有组件，在其基础上添加额外功能,如继承Gallery实现CoverFlow效果

2) 复合型组件定义: 继承某个Layout，实现复合组件自定义，如TextView和EditText组合实现登录注册组件

3) 继承View，实现onDraw()方法，实现自己绘制组件，如翻页效果组件

22:当Activity被系统回收时，如何保存Activity的状态

 开发者可以覆写onSaveInstanceState()方法. onSaveInstanceState()方法接受一个Bundle类型的参数, 开发者可以将状态数据存储到这个Bundle对象中, 这样即使activity被系统摧毁, 当用户重新启动这个activity而调用它的onCreate()方法时, 上述的Bundle对象会作为实参传递给onCreate()方法, 开发者可以从Bundle对象中取出保存的数据, 然后利用这些数据将activity恢复到被摧毁之前的状态.

**1.怎么实现通知栏到聊天界面的跳转**

Intent intent = new Intent(Intent.ACTION\_MAIN);

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK|Intent.FLAG\_ACTIVITY\_CLEAR\_WHEN\_TASK\_RESET); //这样可以清掉所有历史activity

intent.addCategory(Intent.CATEGORY\_HOME);

startActivity(intent);

new一个新的Intent然后用上述方法即可实现跳转。

**2.聊天界面左右分隔怎么实现，怎么实现优化使其不出现错位**

我们可以在adapter里对消息的发放者进行判断，如果不是本人发送的那就设置靠左，否则设置靠右，具体实现如下：

封装消息实体类时，将消息的发送者类型作为Boolean类型存入。在适配数据时，准备两种item\_layout布局文件，一种用于别人的信息展示，另一种用于自己的信息。getView方法加载布局文件之前对消息的发放者进行判断，如果是本人发送，则加载靠右的布局，否则则使用靠左的布局文件，从而实现动态的位置改变

**3.视频，音频的上传**

Android的视频音频上传主要就是通过java的socket来进行与服务器的交流，socket通信可以指定一个服务器的地址，然后可以通过io流实现视频音频的上传。

**4.http协议和soap协议的区别**

HTTP(超文本传输协议)是利用TCP在两台电脑(通常是Web服务器和客户端)之间传输信息的协议。客户端使用Web浏览器发起HTTP请求给Web服务器，Web服务器发送被请求的信息给客户端。

SOAP(Simple Object Access Protocal，简单对象访问协议) 技术有助于实现大量异构程序和平台之间的互操作性,根据我有限的了解，SOAP是把成熟的基于HTTP的WEB技术与XML的灵活性和可扩展性组合在了一起。比如我们.NET中的WEB服务，就是基于SOAP。

简单对象访问协议（SOAP）是W3C组织的一个Note, 它描述了一种在分散的或分布式的环境中如何交换信息的轻量级协议。SOAP是一个基于XML的协议，它包括三个部分：SOAP封装(Envelop)，封装定义了一个描述消息中的内容是什么，是谁发送的，谁应当接受并处理它以及如何处理它们的框架；SOAP编码规则（Encoding Rules），用于表示应用程序需要使用的数据类型的实例；SOAP RPC表示(RPC Representation)，表示远程过程调用和应答的协定；SOAP可以和多种传输协议绑定（Binding），使用底层协议交换信息。在这个文档中，目前只定义了SOAP如何和HTTP以及HTTP扩展进行绑定的框架。

SOAP是个通信协议， SOAP在HTTP协议的基础上，把编写成XML的REQUEST参数, 放在HTTP BODY上提交个WEB SERVICE服务器(SERVLET，ASP什么的) 处理完成后，结果也写成XML作为RESPONSE送回用户端， 为了使用户端和WEB SERVICE可以相互对应，可以使用WSDL作为这种通信方式的描述文件，利用WSDL工具可以自动生成WS和用户端的框架文件，SOAP具备把复杂对象序列化捆绑到XML里去的能力。

SOAP的前身是RPC, 就是远程呼叫处理的协议，这个协议安全性不是很好，多数防火墙都会阻挡RPC的通信包，而SOAP则使用HTTP协议作为基本的协议，使用端口80使得SOAP可以透过防火墙，完成RPC的功能。

SOAP协议和HTTP协议一样，都是底层的通信协议，只是请求包的格式不同而已，SOAP包是XML格式的，现在我们编写WEB SERVICE不需要深入理解SOAP也没关系。如果SERVICE和CLIENT在同样的环境下使用SOAP，由于一般情况下都有自动生成SOAP程序框架的工具，因此不知道细节也没关系. 可是, 如果CLIENT和SERVICE的环境不同，比如说JAVA的Client和.NET的SERVICE进行通信，或者是VB CLIENT和TOMCAT下的JAVA SERVICE通信，还是要知道一点细节为好. 特别是, WSDL或者UDDI都不是标准，如果不让用就只好手工配制SOAP MESSAGE啦。

**5.http和https的区别**

在URL前加https://前缀表明是用SSL加密的。你的电脑与服务器之间收发的信息传输将更加安全。 Web服务器启用SSL需要获得一个服务器证书并将该证书与要使用SSL的服务器绑定。 http和https使用的是完全不同的连接方式,用的端口也不一样,前者是80,后者是443。

HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议

要比http协议安全

HTTPS（Secure Hypertext Transfer Protocol）安全超文本传输协议

它是一个安全通信通道，它基于HTTP开发，用于在客户计算机和服务器之间交换信息。它使用安全套接字层(SSL)进行信息交换，简单来说它是HTTP的安全版。

它是由Netscape开发并内置于其浏览器中，用于对数据进行压缩和解压操作，并返回网络上传送回的结果。HTTPS实际上应用了Netscape的安全全套接字层（SSL）作为HTTP应用层的子层。（HTTPS使用端口443，而不是象HTTP那样使用端口80来和TCP/IP进行通信。）SSL使用40 位关键字作为RC4流加密算法，这对于商业信息的加密是合适的。HTTPS和SSL支持使用X.509数字认证，如果需要的话用户可以确认发送者是谁。

HTTPS和HTTP的区别：

https协议需要到ca申请证书，一般免费证书很少，需要交费。

http是超文本传输协议，信息是明文传输，https 则是具有安全性的ssl加密传输协议

http和https使用的是完全不同的连接方式用的端口也不一样,前者是80,后者是443。

http的连接很简单,是无状态的

HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议 要比http协议安全

HTTPS解决的问题：

1 . 信任主机的问题. 采用https 的server 必须从CA 申请一个用于证明服务器用途类型的证书. 改证书只有用于对应的server 的时候,客户度才信任次主机. 所以目前所有的银行系统网站,关键部分应用都是https 的. 客户通过信任该证书,从而信任了该主机. 其实这样做效率很低,但是银行更侧重安全. 这一点对我们没有任何意义,我们的server ,采用的证书不管自己issue 还是从公众的地方issue, 客户端都是自己人,所以我们也就肯定信任该server.

2 . 通讯过程中的数据的泄密和被窜改

1. 一般意义上的https, 就是 server 有一个证书.

a) 主要目的是保证server 就是他声称的server. 这个跟第一点一样.

b) 服务端和客户端之间的所有通讯,都是加密的.

i. 具体讲,是客户端产生一个对称的密钥,通过server 的证书来交换密钥. 一般意义上的握手过程.

ii. 加下来所有的信息往来就都是加密的. 第三方即使截获,也没有任何意义.因为他没有密钥. 当然窜改也就没有什么意义了.

2. 少许对客户端有要求的情况下,会要求客户端也必须有一个证书.

a) 这里客户端证书,其实就类似表示个人信息的时候,除了用户名/密码, 还有一个CA 认证过的身份. 应为个人证书一般来说上别人无法模拟的,所有这样能够更深的确认自己的身份.

b) 目前少数个人银行的专业版是这种做法,具体证书可能是拿U盘作为一个备份的载体.

HTTPS 一定是繁琐的.

a) 本来简单的http协议,一个get一个response. 由于https 要还密钥和确认加密算法的需要.单握手就需要6/7 个往返.

i. 任何应用中,过多的round trip 肯定影响性能.

b) 接下来才是具体的http协议,每一次响应或者请求, 都要求客户端和服务端对会话的内容做加密/解密.

i. 尽管对称加密/解密效率比较高,可是仍然要消耗过多的CPU,为此有专门的SSL 芯片. 如果CPU 信能比较低的话,肯定会降低性能,从而不能serve 更多的请求.

ii. 加密后数据量的影响. 所以，才会出现那么多的安全认证提示

**6.listview的逐行显示**

可以将每一条数据先加载到list集合中，然后通过适配器将每一条数据加载到listview中实现listview对数据的逐行显示。

**7.edittext中文字的下划线效果**

自定义一个·EditText的[派生类](http://zhidao.baidu.com/search?word=%E6%B4%BE%E7%94%9F%E7%B1%BB&fr=qb_search_exp&ie=utf8)，LineEditText，在LineEditText类的OnDraw函数中实现画一条[下划线](http://zhidao.baidu.com/search?word=%E4%B8%8B%E5%88%92%E7%BA%BF&fr=qb_search_exp&ie=utf8)的功能

**8.activity以及service的生命周期**

**activity**

onCreate(Bundle savedInstanceState) 第一次创建时调用

onStart() 被用户可见时调用

onRestart() 当Activity处于stop状态又被重新启动时调用

onResume() 当获得焦点即可与用户交互时调用

onPause() 当失去焦点时调用

onStop() 当不可见时调用

onDestroy() 当销毁时调用

**service**

startService：onCreate()- >onStartCommand()->startService()->onDestroy()

BindService：onCreate()->onBind()->onUnbind()->onDestroy()

**9.安卓系统的理解，优缺点**

Android一词的本义指“[机器人](http://baike.baidu.com/view/2788.htm)”，同时也是[Google](http://baike.baidu.com/view/105.htm)于2007年11月5日 宣布的基于[Linux](http://baike.baidu.com/view/1634.htm)平台的[开源](http://baike.baidu.com/view/9664.htm)手机操作系统的名称，该平台由[操作系统](http://baike.baidu.com/view/880.htm)、中间件、[用户界面](http://baike.baidu.com/view/362528.htm)和[应用软件](http://baike.baidu.com/view/7886.htm)组成。

版本：每个版本代表的甜点的尺寸越变越大，然后按照26个字母数序：[纸杯蛋糕](http://baike.baidu.com/view/1180146.htm)（Android 1.5），[甜甜圈](http://baike.baidu.com/view/30807.htm)（Android 1.6），[松饼](http://baike.baidu.com/view/41553.htm)（Android 2.0/2.1），[冻酸奶](http://baike.baidu.com/view/1442740.htm)（Android 2.2），[姜饼](http://baike.baidu.com/view/622742.htm)（Android 2.3），[蜂巢](http://baike.baidu.com/view/47019.htm)（Android 3.0），[冰激凌三明治](http://baike.baidu.com/view/5830554.htm)（Android 4.0），[果冻豆](http://baike.baidu.com/view/7857757.htm)（Jelly Bean，Android4.1和Android 4.2）。

Android分为四个层，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和Linux[内核](http://baike.baidu.com/view/1366.htm)层。

Android开发四大组件分别是：活动（[Activity](http://baike.baidu.com/view/1136177.htm))： 用于表现功能。服务（[Service](http://baike.baidu.com/view/3862339.htm))： 后台运行服务，不提供界面呈现。广播接收器（BroadcastReceiver)：用于接收广播。内容提供商（[Content Provider](http://baike.baidu.com/view/1445128.htm)）： 支持在多个应用中存储和读取数据，相当于[数据库](http://baike.baidu.com/view/1088.htm)。

优点：开放性，挣脱束缚，丰富硬件，Google应用

缺点：版本过多，升级过快

用户体验不一致

**10.sim卡EF文件是什么**

SIM卡里的所有文件按树来组织:

1. 主文件MF(Master File)——每一块SIM卡只有一个唯一的主文件, 其他所有文件都是它的子孙, 主文件只有[文件头](http://zhidao.baidu.com/search?word=%E6%96%87%E4%BB%B6%E5%A4%B4&fr=qb_search_exp&ie=utf8),里面存放着整个SIM卡的控制和管理信息  
   2>专用文件DF(Dedicated File)——也是只有一个[文件头](http://zhidao.baidu.com/search?word=%E6%96%87%E4%BB%B6%E5%A4%B4&fr=qb_search_exp&ie=utf8), 里面存放着整个目录的管理控制信息, 专用文件相当于一个目录的根.  
   3>基本文件EF(Elementary File)——既有[文件头](http://zhidao.baidu.com/search?word=%E6%96%87%E4%BB%B6%E5%A4%B4&fr=qb_search_exp&ie=utf8),也有文件体, 文件头存放该文件的位置和控制信息, 文件体存放真正的数据, 整个SIM卡中只有基本文件有文件体, 也只有基本文件才用来存放数据.

**11.四大组件哪些能动态注册**

广播

**12.安卓文件存储方式**

文件的操作模式：

Context.MODE\_PRIVATE 为默认的操作模式，代表该文件私有，只能被应用本身使用，覆盖源文件

　　Context.APPEND 为追加模式，私有的。存在就往文件追加内容

　　Context.MODE\_WORLD\_READABLE:表示当前文件可以被其他应用读取

　　Context.MODE\_WORLD\_WRITEABLE:表示当前文件可以被其他应用写入

　　如果希望文件被其他应用读和写，可以传入：

　　openFileOutput("zhgang.txt",Context.MODE\_WORLD\_READABLE+

　　Context.MODE\_WORLD\_WRITEABLE);

　　android有一套自己的安全模型，当应用程序(.apk)在安装时系统就会分配给他一个userid，当该应用要去访问其他资源比如文件的时候

　　就需要userid匹配。默认情况下，任何应用创建文件，sharedprefernces，数据库都应该是私有的(位于/data/data/<package name>/files),

　　其他程序无法访问，除非在创建时指定了Context.MODE\_WORLD\_READABLE，Context.MODE\_WORLD\_WRITEABLE，只有这样其他程序才能正确访问

**13.intent的功能，如何定义显示intent,隐式intent?**

在一个Android应用中，主要是由一些组件组成，（Activity,Service,ContentProvider,etc.)在这些组件之间的通讯中，由Intent协助完成。

Intent传递过程中，要找到目标消费者（另一个Activity,IntentReceiver或Service），也就是Intent的响应者，有两种方法来匹配：

显示intent：代码简洁明了，执行了函数，就会马上跳转到指定的Activity中（  Intent i = new Intent(Test.this, TestB.class);  
            this.startActivity(i);）。

隐式intent：隐式匹配，首先要匹配Intent的几项值：Action（要执行的动作）, Category（要执行动作的目标所具有的特质或行为归类）, Data/extras（即执行动作要操作的数据和传递到目的的附加信息）,type（要执行的目标Activity所能处理的MIME数据类型）Component  
如果填写了Componet就是上例中的Test.class)这就形成了显示匹配。

**14.****内存泄露遇见过哪些问题，解决办法，回收的算法**

问题：运行两次后的输出结果，程序退出了，后台仍然有两个线程在跑，无法被释放，如果是大程序，就会导致：一、耗电；二、内存泄露。

解决方法：

在程序销毁的时候，要通过Handler的removeCallbacks(Runnable r)方法来手动释放掉该线程，当然要把Runnable单独提出来写。使用完后，程序必须负责相应的调用free或delete释放该内存块，否则，这块内存就不能被再次使用，我们就说这块[内存泄漏](http://zhidao.baidu.com/search?word=%E5%86%85%E5%AD%98%E6%B3%84%E6%BC%8F&fr=qb_search_exp&ie=utf8)了。

**回收算法：tracing算法(Tracing Collector)，compacting算法(Compacting Collector)，copying算法(Coping Collector)，generation算法(Generational Collector)，adaptive算法(Adaptive Collector)**

**15.接口和抽象类的区别？**

abstract可以修饰抽象方法，而一个类只要有一个抽象方法，就必须用abstract定义该类，即抽象类。

用interface修饰的类，里面的方法都是抽象方法，因此在定义接口的时候，可以直接不加那些修饰，系统会默认的添上去。接口里面的字段都是公有常量，即public static final修饰的字段。

**16.string和stringbuffer和stringbuilder区别？**

STRING的长度是不可变的，STRINGBUFFER的长度是可变的。如果你对字符串中的内容经常进行操作，特别是内容要修改时，那么使用StringBuffer，如果最后需要String，那么使用StringBuffer的toString()方法。

stringbuilder类被设计用作 StringBuffer 的一个简易替换，用在字符串缓冲区被单个线程使用的时候（这种情况很普遍）。如果可能，建议优先采用该类，因为在大多数实现中，它比 StringBuffer 要快。两者的方法基本相同。

**17.html5？**

HTML5是用于取代1999年所制定的 HTML 4.01 和 XHTML 1.0 标准的 [HTML](http://baike.baidu.com/view/692.htm) [1]（[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)下的一个应用）标准版本；现在仍处于发展阶段，但大部分[浏览器](http://baike.baidu.com/view/7718.htm)已经支持某些 HTML5 技术。HTML 5有两大特点：首先，强化了 Web 网页的表现性能。其次，追加了本地数据库等 Web 应用的功能。广义论及HTML5时，实际指的是包括[HTML](http://baike.baidu.com/view/692.htm)、[CSS](http://baike.baidu.com/subview/15916/5236733.htm)和[JavaScript](http://baike.baidu.com/view/16168.htm)在内的一套技术组合。它希望能够减少[浏览器](http://baike.baidu.com/view/7718.htm)对于需要[插件](http://baike.baidu.com/view/18979.htm)的丰富性网络应用服务（plug-in-based rich internet application，RIA)，如Adobe Flash、Microsoft Silverlight，与Oracle JavaFX的需求，并且提供更多能有效增强网络应用的标准集。

是一种标记语言。

**18.自定义一个不带系统边框的Dialog？**

首先定义样式文件style.xml,边框设计windowframe属性为@null,将背景设置为自己想要的背景,将自定义dialog继承Dialog，将布局文件载入，创建dialog，并将自己设置的样式文件加载进去，最终实现自定义的Dialog.

**19.定义一个跑马灯效果的textview？**

显示跑马灯效果的前提条件就是你的文本内容要比显示文本的外部组件长，即外部组件无法完整的显示。

内部的文本内容。

1、layout\_width=”"设置为成比文本内容短的固定值，最好不要写成wrap\_content或者fill\_parent。

2,android:singleLine="true" 表示使用单行文字

android:ellipsize="marquee"

设置了文字过长时如何切断文字，可以有none,start,middle, end, 如果使用走马灯效果则设为marquee。

android:focusableInTouchMode="true"

android:focusable="true"

android:marqueeRepeatLimit="marquee\_forever"

Android的缺省行为是在控件获得 Focus时才会显示走马灯效果。

**20.图片的异步加载的方法？**

1.利用软引用来缓存图片Bitmap，用图片的URL作为缓存查找的Key；   
2.设两级缓存，一级是SoftReference，二级是本地SD卡；   
3.如果两级缓存都没取到图片，则从服务器获取，并加入缓存；   
4.加载完后通过回调接口通知UI更新；

1. **瀑布流实现方式？**

1）自定义scrollView，动态代码添加LinearLayout及ImageView

a.自定义scrollView,并且使用监听器模式，对其滚动到最顶部、及最底部进行监听操作。

b.在Activity里对scrollView的横向的LinearLayout动态添加列数，每一列是一个竖向的linearLayout.

c.根据列数以及list.size()对每一列的linearLayout添加ImageView.

d.根据监听器对其滚动到顶部（不做操作）、滚到最底部（加载更多数据）、正在滚动进行操作（滚动超过两屏，回收两屏之前图片回收及回滚到之前屏幕图片重载）。

1. **把一张特别大的图片，分成几十张小的图片，在最短的时间内给处理？**

[**http://www.linuxidc.com/Linux/2012-11/73939p2.htm**](http://www.linuxidc.com/Linux/2012-11/73939p2.htm)

方法1：如果你将return-data设置为“true”，你将会获得一个与内部数据关联的Action，并且bitmap以此方式返回：(Bitmap)extras.getParcelable("data")。注意：如果你最终要获取的图片非常大，那么此方法会给你带来麻烦，所以你要控制outputX和outputY保持在较小的尺寸。鉴于此原因，在我的代码中没有使用此方法（(Bitmap)extras.getParcelable("data")）。

方法2： 如果你将return-data设置为“false”，那么在onActivityResult的Intent数据中你将不会接收到任何Bitmap，相反，你需要将MediaStore.EXTRA\_OUTPUT关联到一个Uri，此Uri是用来存放Bitmap的。但是还有一些条件，首先你需要有一个短暂的与此Uri相关联的文件地址，当然这不是个大问题（除非是那些没有sdcard的设备）。

1. **Android dvm的进程和Linux的进程, 应用程序的进程是否为同一个概念？**

DVM指dalivk的虚拟机。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。而每一个DVM进程都是在Linux系统中的一个进程，所以可以认为是同一个概念。

Dalvik和Java运行环境的区别　　[  
 1：Dalvik主要是完成对象生命周期管理，[堆栈](http://zhidao.baidu.com/search?word=%E5%A0%86%E6%A0%88&fr=qb_search_exp&ie=utf8)管理，线程管理，安全和异常管理，以及[垃圾回收](http://zhidao.baidu.com/search?word=%E5%9E%83%E5%9C%BE%E5%9B%9E%E6%94%B6&fr=qb_search_exp&ie=utf8)等等重要功能。 　　  
 2：Dalvik负责进程隔离和线程管理，每一个Android应用在底层都会对应一个独立的Dalvik虚拟机实例，其代码在虚拟机的解释下得以执行。 　　  
 3：不同于Java虚拟机运行java[字节码](http://zhidao.baidu.com/search?word=%E5%AD%97%E8%8A%82%E7%A0%81&fr=qb_search_exp&ie=utf8)，Dalvik虚拟机运行的是其专有的[文件格式](http://zhidao.baidu.com/search?word=%E6%96%87%E4%BB%B6%E6%A0%BC%E5%BC%8F&fr=qb_search_exp&ie=utf8)Dex 　　 4:dex[文件格式](http://zhidao.baidu.com/search?word=%E6%96%87%E4%BB%B6%E6%A0%BC%E5%BC%8F&fr=qb_search_exp&ie=utf8)可以减少整体文件尺寸，提高I/o操作的类查找速度。 　　  
 5:odex是为了在运行过程中进一步提高性能，对dex文件的进一步优化。 　　  
 6：所有的Android应用的线程都对应一个Linux线程，虚拟机因而可以更多的依赖[操作系统](http://zhidao.baidu.com/search?word=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&fr=qb_search_exp&ie=utf8)的[线程调度](http://zhidao.baidu.com/search?word=%E7%BA%BF%E7%A8%8B%E8%B0%83%E5%BA%A6&fr=qb_search_exp&ie=utf8)和[管理机制](http://zhidao.baidu.com/search?word=%E7%AE%A1%E7%90%86%E6%9C%BA%E5%88%B6&fr=qb_search_exp&ie=utf8) 　　  
 7：有一个特殊的虚拟机进程Zygote，他是虚拟机实例的[孵化器](http://zhidao.baidu.com/search?word=%E5%AD%B5%E5%8C%96%E5%99%A8&fr=qb_search_exp&ie=utf8)。它在系统启动的时候就会产生，它会完成虚拟机的初始化，库的加载，预制类库和初始化的操作。如果系统需要一个新的虚拟机实例，它会迅速复制自身，以最快的数据提供给系统。对于一些只读的系统库，所有虚拟机实例都和Zygote共享一块内存区域。

1. **嵌入式操作系统内存管理有哪几种，各有何特性 ?**

1、虚拟内存管理机制： 有一些嵌入式处理器提供了MMU，在MMU具备内存地址映射和寻址功能，它使操作系统的内存管理更加方便。如果存在MMU ，操作系统会使用它完成从虚拟地址到物理地址的转换， 所有的应用程序只需要使用虚拟地址寻址数据。 这种使用虚拟地址寻址整个系统的主存和辅存的方式在现代操作系统中被称为虚拟内存。MMU 便是实现虚拟内存的必要条件。 虚拟内存的管理方法使系统既可以运行体积比物理内存还要大的应用程序，也可以实现“按需调页”策略，既满足了程序的运行速度，又节约了物理内存空间。 在L inux系统中，虚拟内存机制的实现实现为我们提供了一个典型的例子：在不同的体系结构下， 使用了三级或者两级页式管理，利用MMU 完成从虚拟地址到物理地址之间的转换。基于虚拟内存管理的内存最大好处是：由于不同进程有自己单独的进程空间，十分有效的提高了系统可靠性和安全性。 2、非虚拟内存管理机制： 在实时性要求比较高的情况下，很多嵌入式系统并不需要虚拟内存机制：因为虚拟内存机制会导致不确定性的 I/O阻塞时间， 使得程序运行时间不可预期，这是实时嵌入式系统的致命缺陷；另外，从嵌入式处理器的成本考虑，大多采用不装配MMU 的嵌入式微处理器。所以大多嵌入式系统采用的是实存储器管理策略。因而对于内存的访问是直接的，它对地址的访问不需要经过MMU，而是直接送到地址线上输出，所有程序中访问的地址都是实际的物理地址；而且，大多数嵌入式操作系统对内存空间没有保护，各个进程实际上共享一个运行空间。一个进程在执行前，系统必须为它分配足够的连续地址空间，然后全部载入主存储器的连续空间。

1. **什么是嵌入式实时操作系统, Android 操作系统属于实时操作系统吗?**

[嵌入式实时操作系统](http://zhidao.baidu.com/search?word=嵌入式实时操作系统&fr=qb_search_exp&ie=utf8)是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统作出快速响应，并控制所有实时任务协调一致运行的[嵌入式操作系统](http://zhidao.baidu.com/search?word=嵌入式操作系统&fr=qb_search_exp&ie=utf8)。  
 [嵌入式操作系统](http://zhidao.baidu.com/search?word=嵌入式操作系统&fr=qb_search_exp&ie=utf8)主要用于工业控制、军事设备、航空航天等领域对系统的响应时间有苛刻的要求，这就需要使用[实时系统](http://zhidao.baidu.com/search?word=实时系统&fr=qb_search_exp&ie=utf8)。又可分为软实时和硬实时两种，而 android 是基于 linux 内核的，因此属于软实时。

**26.一条最长的短信息约占多少byte?**

一条最长的短信息约占140byte

1. **android中的动画有哪几类，它们的特点和区别是什么？**

答：Android中动画可以分为两大类：帧动画、补间动画

1）补间动画：（你定义一个开始和结束，中间的部分由程序运算得到。就是对场景里的对象不断的进行图像变化来产生动画效果（旋转、平移、放缩和渐变））AlphaAnimation（渐变型动画）、scaleAnimation（缩放型动画）、 TranslateAnimation（平移型动画）、 RotateAnimation（旋转型动画）、

2）逐帧动画：Frame（把一连串的图片进行系列化连续播放，如同放电影的效果），它是通过播放一张一张图片来达到动画的效果；

**28.handler机制的原理，looper通过什么方法开始的？**

一个Handler允许你发送和处理Message和Runable对象，每个线程都有自己的Looper，每个Looper中封装着MessageQueue。Looper负责不断的从自己的消息队列里取出队头的任务或消息执行。每个handler也和线程关联，Handler负责把Message和Runable对象传递给MessageQueue（用到post ，sendMessage等方法），而且在这些对象离开MessageQueue时，Handler负责执行他们（用到handleMessage方法）。

其中Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象提供额外的两个int域和一个Object域。

默认情况下一个线程是不存在消息循环（message loop）的，需要调用Looper.prepare()来给线程创建一个消息循环，调用Looper.loop()来使消息循环起作用，从消息队列里取消息，处理消息。调用Looper.loop()之后循环已经开始，下一个由handler发的message将会被这个handler的handleMessage方法处理，这是一个----循环。

Looper从MessageQueue中取出Message之后，交由Handler的handleMessage进行处理。处理完成后，调用Message.recycle()将其放入Message Pool中。

**29.说说mvc模式的原理，在android中的运用？**

答：mvc是model,view,controller的缩写，mvc包含三个部分：  
　　模型（model）对象：是应用程序的主体部分，所有的业务逻辑都应该写在该层。  
　　视图（view）对象：是应用程序中负责生成用户界面的部分。也是在整个mvc架构中用户唯一可以看到的一层，接收用户的输入，显示处理结果。  
　　控制器（control）对象：是根据用户的输入，控制用户界面数据显示及更新model对象状态的部分，控制器更重要的一种导航功能，响应用户出发的相关事件，交给m层处理。  
　　android鼓励弱耦合和组件的重用，在android中mvc的具体体现如下：  
　　1)视图层（view）：一般采用xml文件进行界面的描述，使用的时候可以非常方便的引入，当然，如果你对android了解的比较的多了话，就一定可以想到在android中也可以使用javascript+html等的方式作为view层，当然这里需要进行java和javascript之间的通信，幸运的是，android提供了它们之间非常方便的通信实现。  
　　2)控制层（controller）：android的控制层的重任通常落在了众多的acitvity的肩上，这句话也就暗含了不要在acitivity中写代码，要通过activity交割model业务逻辑层处理，这样做的另外一个原因是android中的acitivity的响应时间是5s，如果耗时的操作放在这里，程序就很容易被回收掉。  
　　3)模型层（model）：对数据库的操作、对网络等的操作都应该在model里面处理，当然对业务计算等操作也是必须放在的该层的。

**30.如何让Activity变成一个窗口？**

答：在清单文件AndroidManifest.xml中相应的<activity>标签内设置属性android:theme=”@android:style/Theme.Dialog”

**31.后台的Activity被系统回收怎么办？**

除了在栈顶的Activity，其他的Activity都有可能在内存不足的时候被系统回收，一个Activity越处于栈底，被回收的可能性就越大。如果我们没有覆写onSaveInstanceState()方法，此方法的默认实现会自动保存Activity中的某些状态数据，比如Activity中各种UI空间的状态。Android应用框架中定义的几乎所有的UI控件都恰当的实现了onSaveInstanceState()方法，因此当Activity被摧毁和重建时，这些UI控件会自动保存和恢复状态数据。比如EditText控件会自动保存和恢复输入的数据，而CheckBox控件会自动保存和恢复选中状态。开发者只需要为这些控件指定一个唯一的ID（通过设置android:id属性即可），剩余的事情就可以自动完成了。如果没有为控件指定ID，则这个控件就不会进行自动的数据保存和恢复操作。  
由上所述，如果我们需要覆写onSaveInstanceState()方法，一般会在第一行代码中调用该方法的默认实现：super.onSaveInstanceState(outState)。

**32.ListView优化？**

答：1)、对convetView进行判空，是当convertView不为空的时候直接重新使用convertView

从而减少了很多不必要的View的创建

2)定义一个ViewHolder，将convetView的tag设置为ViewHolder,不为空时重新使用即可

3)、当ListView加载数据量较大时可以采用分页加载和图片异步加载

**33.IPC及其原理？**

1).Binder通信是如何实现的？

1.Binder通信是通过linux的binder driver来实现的，

2.Binder通信操作类似线程迁移(thread migration)，两个进程间IPC看起来就象是一个进程进入另一个进程执行代码然后带着执行的结果返回；

1）IPC（Inter-Process Communication，进程间通信）,

aidl是 Android Interface definition language的缩写，它是一种android内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口.编译器可以通过扩展名为aidl的文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的.

BroadcastReceiver也可以实现进程间通信

ContentProvider 提供进程间数据共享

2).Android的 Service工作流程

1.Android虚拟机启动之前系统会先启动service Manager进程；

2.service Manager打开binder驱动，并通知binder kernel驱动程序这个进程将作为System Service Manager；

3.然后该进程将进入一个循环，等待处理来自其他进程的数据。

4.用户创建一个System service后，通过defaultServiceManager得到一个远程ServiceManager的接口，通过这个接口我们可以调用addService函数将System service添加到Service Manager进程中；

5.然后client可以通过getService获取到需要连接的目的Service的IBinder对象，这个IBinder是Service的BBinder在binder kernel的一个参考，

6.所以service IBinder 在binder kernel中不会存在相同的两个IBinder对象，每一个Client进程同样需要打开Binder驱动程序。对用户程序而言，我们获得这个对象就可以通过binder kernel访问service对象中的方法。

7.Client与Service在不同的进程中，通过这种方式实现了类似线程间的迁移的通信方式，对用户程序而言当调用Service返回的IBinder接口后，访问Service中的方法就如同调用自己的函数。

**34.View如何刷新？**

安卓要更新界面不要在主线程中去做

如果是在线程thread中获取到了新的数据,需要配合使用hanlder进行刷新.

在线程中刷新View用postinvalidate()；

在UI线程中刷新View的方法是invalidate()函数，

当调用线程处于空闲状态时，调onDraw()刷新界面

如果是listView刷新数据，用adapter.notifyDataSetInvalidated();

adapter.notifyDataSetChanged();

**35.DDMS与TraceView的区别？**

DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器 。

**36.在Java中如何引入C语言？**

java中利用JNI引用本地语言 （java native interface java 本地接口）接口 。  
1.在应用程序中定义本地接口（native）编译成.h头文件，交由C程序员实现，将.c实现通过NDk编译成.so动态链接库(windows下是.dll, linux下是.so，导入项目中的libs/armeabi,代码中调用本地接口，  
应用场景：音频,拍摄车牌号

**37.链表和数组的区别？**

数组是将元素在内存中连续存放，由于每个元素占用内存相同，可以通过下标迅速访问数组中任何元素。但是如果要在数组中增加一个元素，需要移动大量元素，在内存中空出一个元素的空间，然后将要增加的元素放在其中。同样的道理，如果想删除一个元素，同样需要移动大量元素去填掉被移动的元素。如果应用需要快速访问数据，很少或不插入和删除元素，就应该用数组。

链表恰好相反，链表中的元素在内存中不是顺序存储的，而是通过存在元素中的指针联系到一起。比如：上一个元素有个指针指到下一个元素，以此类推，直到最后一个元素。如果要访问链表中一个元素，需要从第一个元素开始，一直找到需要的元素位置。但是增加和删除一个元素对于链表数据结构就非常简单了，只要修改元素中的指针就可以了。如果应用需要经常插入和删除元素你就需要用链表数据结构了。

二者都属于一种数据结构  
　　(1) 从逻辑结构角度来看  
　　 a, 数组必须事先定义固定的长度（元素个数），不能适应数据动态地增减的情况。当数据增加时，可能超出原先定义的元素个数；当数据减少时，造成内存浪费。  
　　 b,链表动态地进行存储分配，可以适应数据动态地增减的情况，且可以方便地插入、删除数据项。（数组中插入、删除数据项时，需要移动其它数据项）  
　　(2)从内存存储角度来看  
　　 a,(静态)数组从栈中分配空间, 对于程序员方便快速,但自由度小。  
　　 b, 链表从堆中分配空间, 自由度大但申请管理比较麻烦.

1. **Hash表是什么？有什么用？**

散列表（Hash table，也叫哈希表），是根据关键码值(Key value)而直接进行访问的数据结构。也就是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做散列函数，存放记录的数组叫做散列表。

给定表M，存在函数f(key)，对任意给定的关键字值key，代入函数后若能得到包含该关键字的记录在表中的地址，则称表M为哈希(Hash）表，函数f(key)为哈希(Hash) 函数

作用：HASH表主要就是提供更快的查找速度，简单来说就是分桶。

比如说你们学校的一个年级有很多班是吧？ 那么现在要查找一位同学。如果不看班级直接找，就需要每个班级一个一个的查找是不？ 那现在因为分了班级，我们有一个函数可以通过名字产生它的班级号。比如说王勇，二班。 那么现在我们直接hash( 王勇) = 2; 然后我们就直接从2班开始找。 如果2班有50个学生，那么我们最多查找50次对吗？ 好。假设你们年级一共有10个班，如果没有这个hash函数，那么直接找，最坏可能要查找50\*10=500次。！！ 这个班级其实就是每个桶！！ 这就是分桶。。。通过哈希函数产生哈希值，然后相同哈希值的元素放在相同的桶里边。。。！

这样可以通过hash这个间接作用减少查找的时间和次数！这就是它的意义！！

1. **什么是锁？有什么用？有哪些锁？为什么需要锁？**

锁是为了保证安全性，如程序运行时保证另外的程序不能再对本程序所使用到的数据进行某些操作，版本软件的“不能合并”文件不能同时被两人修改等。

锁分为：线程锁，数据库锁，SVN锁等等

1、当几个线程都用到了某个量，但是这个量却能影响程序的运行时就需要线程锁来控制一次只能由一个线程访问这个量。如果没有使用线程锁会出现争用情况。

两个或更多的线程或进程读或写一些共享数据，而最终结果取决于这些线程是如何被调度计时的。争用情况可能会导致不可预见的结果和隐蔽的程序错误。

2、数据库锁的作用是防止程序运行的时候其他的程序不能再对该程序所使用到的数据进行操作，保护数据的安全性。

3、SVN锁的作用是防止版本软件中的“不能合并”的文件(如：图形文件）不能被多人同时更改的时候进行锁定，当一个人对其进行操作时，其他人不能对这个文件再进行操作，保证了文件的安全性。

**41.MVC作用？**

答: Android中界面部分也采用了当前比较流行的MVC框架。

在Android中：

1) 视图层（View）：一般采用XML文件进行界面的描述，使用的时候可以非常方

便的引入。也可以使用JavaScript+HTML等的方式作为View层，通过WebView组

件加载,同时可以实现Java和JavaScript之间的通信。

2) 控制层（Controller）：这句话也就暗含了不要在Acitivity中写代码，要通过Activity

交割Model业务逻辑层处理，这样做的另外一个原因是Android中的Acitivity的响

应时间是5s，如果耗时的操作放在这里，Android的控制层的重任通常落在了众多

的Acitvity的肩上，程序就很容易被回收掉。

1. 模型层（Model）：对数据库的操作、对网络等的操作都应该在Model里面处理，

当然对业务计算等操作也是必须放在的该层的。  
 在Android SDK中的数据绑定，也都是采用了与MVC框架类似的方法来显示数据。在控制层上将数据按照视图模型的要求（也就是Android SDK中的Adapter）封装就可以直接在视图模型上显示了，从而实现了数据绑定。比如显示Cursor中所有数据的ListActivity，其视图层就是一个ListView，将数据封装为ListAdapter，并传递给ListView，数据就在ListView中显示。

**42.px,dp,sp区别？**

**px：**

即像素，1px代表屏幕上一个物理的像素点；

px单位不被建议使用，因为同样100px的图片，在不同手机上显示的实际大小可能不同

**dp = dip :** device independent pixels(设备独立像素). 不同设备有不同的显示效果,这个和设备硬件有关，一般我们为了支持WVGA、HVGA和QVGA 推荐使用这个，不依赖像素。

**sp：**

与缩放无关的抽象像素（Scale-independent Pixel）。sp和dp很类似但唯一的区别是，Android系统允许用户自定义文字尺寸大小（小、正常、大、超大等等），当文字尺寸是“正常”时1sp=1dp=0.00625英寸，而当文字尺寸是“大”或“超大”时，1sp>1dp=0.00625英寸。类似我们在windows里调整字体尺寸以后的效果——窗口大小不变，只有文字大小改变。

**43.android 系统架构，按顺序？**

答：1）应用程序层 java语言 应用程序开发

1. 应用程序框架层 java语言 OS定制 framework层开发
2. 系统运行库层 C C++ 实现 so库
3. Linux内核层

**44.android有几种布局，有什么特点？**

答：最常用的布局有以下这几种：

第一种：帧布局（框架布局）FrameLayout，在这个布局中，所有的子元素统统放于这块区域的左上角，并且后面的子元素直接覆盖在前面的子元素之上，将前面的子元素部分和全部遮挡。

第二种：线性布局LinearLayout，最常用的一种布局方式，所有子控件的对齐方式，取决于如何定义 orientation的属性：vertical   垂直方向 ，如果按照这种方向所有的子控件将按照垂直的方式分布在布局上，每行只允许有一个子元素，horizontal水平方向 ，这时子控件将会以水平的方向分布在布局中。

第三种：绝对布局AbsoluteLayout，又可以叫做坐标布局，可以直接指定子元素的绝对位置，这种布局简单直接，直观性强，但是由于手机屏幕尺寸差别比较大，使用绝对定位的适应性会比较差。

第四种：相对布局RelativeLayout，允许子元素指定它们相对于其父元素或兄弟元素的位置，这是实际布局中最常用的布局方式之一。它灵活性大很多，当然属性也多，操作难度也大，属性之间产生冲突的的可能性也大，使用相对布局时要多做些测试。

第五种：表格布局TableLayout，表格布局TableLayout以行列的形式管理子元素，每一行是一个TableRow布局对象，当然也可以是普通的View对象，TableRow里每放一个元素就是一列，总列数由列数最多的那一行决定。

第六种：网格布局 GridLayout，在Android 4.0中，新引入的GridLayout网格布局，GridLayout布局使用虚细线将布局划分为行，列和单元格，也支持一个控件在行，列上都有交错排列。而GridLayout使用的其实是跟LinearLayout类似的API，只不过是修改了一下相关的标签而已，所以对于开发者来说，掌握GridLayout还是很容易的事情。

(Android 4.0的SDK已经发布，在众多的新增特性中，其中对开发者来说比较重要的特性之一，是新增的两种界面布局方式：Space和Gridlayout)

**45.常用的设计模式？**

答：单例模式；工厂模式；

观察者模式：ContentObserver, 监听ContentProvider ContentResolver；

适配器模式Adapter(SimpleAdapter BaseAdapter ArrayAdapter CursorAdapter)以及各种Adapter的数据源；

代理模式 AOP；

**46.常用的排序法？**

答: **冒泡排序** O(n^2） : 冒泡排序也是最简单最基本的排序方法之一。冒泡排序的思想很简单，就是以此比较相邻的元素大小，将小的前移，大的后移，就像水中的气泡一样，最小的元 素经过几次移动，会最终浮到水面上。

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

// 内循环控制比较后移位

for (int j = num.length-1; j > i ; j--) {

if (num[j-1]>num[j]) {

temp = num[j-1];

num[j-1] = num[j];

num[j] = temp;

}

}

**快速排序**O(n log n) : 快速排序采用的思想是分治思想。快速排序算法的核心算法是分区操作，即如何调整基准的位置以及调整返回基准的最终位置以便分治递归。

**插入排序** O(n^2） :将新来的元素按顺序放入一个已有的有序序列当中。

**选择排序** O(n^2） :第i趟简单选择排序是指通过n-i次关键字的比较，从n-i+1个记录中选出关键字最小的记录，并和第i个记录进行交换。共需进行i-1趟比较，直到所有记录排序完成为止。

**47.写一个快速排序法？**

快速排序（Quicksort）是对[冒泡排序](http://baike.baidu.com/view/254413.htm)的一种改进。它的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以[递归](http://baike.baidu.com/view/96473.htm)进行，以此达到整个数据变成有序序列。

**public** **class** QuickSort {

**public** **static** **void** main(String[] args) {

Integer[] list={34,3,53,2,23,7,14,10};

QuickSort qs= **new** QuickSort();

qs.quick(list);

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

System. *out*.print(list[i]+ " ");

}

System. *out*.println();

}

**public** **void** quick(Integer[] str) {

**if** (str.length > 0) { //查看数组是否为空

\_quickSort(str, 0, str. length - 1); //从第一个数到最后一个数

}

}

**public** **void** \_quickSort(Integer[] list, **int** low, **int** high) { //low，最小索引，high最高索引

**if** (low < high) {

**int** middle = getMiddle(list, low, high); //将list数组进行一分为二 ,取得中间值对应的索引

\_quickSort(list, low, middle - 1); //对低字表进行递归排序

\_quickSort(list, middle + 1, high); //对高字表进行递归排序

}

}

//获得数组的中间值的索引（即位置）(参数：数组，最低索引，最高索引)

**public** **int** getMiddle(Integer[] list, **int** low, **int** high) {

**int** tmp = list[low]; //先用数组的第一个值作为中轴(对比值)

**while** (low < high) {

//后一半的数组，把中值与最后一个比，如果不比它大则不交换则再与前一个再比

**while** (low < high && list[high] > tmp) {

high--;

}

list[low] = list[high]; //最后list[high]小于对比值时,则把这个值移到低端------------>这里list[low]值发生了改变 ,为下面循环作准备

**while** (low < high && list[low] < tmp) { //前一半，中值与低端值比，如果比中值小则与下一个数比

low++; //修改索引为下一个数

}

list[high] = list[low]; //否则，如果list[low]比中轴大，则把这个值移到高端------------->这里list[high]值发生了改变

}

list[low] = tmp; //上面循环处理完后，把当前中轴值存到当前数组的低端（其实当前的low在上面处理完后，已逐渐修改为中间的索引）

System.*out*.println(low +"============"+tmp);

**return** low; //返回中轴的位置

}

}

**48.synchronized 是什么？有什么用？**

答：Java语言的关键字，当它用来修饰一个方法或者一个代码块的时候，能够保证在同一时刻最多只有一个线程执行该段代码。

     一、当两个并发线程访问同一个对象object中的这个synchronized(this)同步代码块时，一个时间内只能有一个线程得到执行。另一个线程必须等待当前线程执行完这个代码块以后才能执行该代码块。

     二、然而，当一个线程访问object的一个synchronized(this)同步代码块时，另一个线程仍然可以访问该object中的非synchronized(this)同步代码块。

     三、尤其关键的是，当一个线程访问object的一个synchronized(this)同步代码块时，其他线程对object中所有其它synchronized(this)同步代码块的访问将被阻塞。

     四、第三个例子同样适用其它同步代码块。也就是说，当一个线程访问object的一个synchronized(this)同步代码块时，它就获得了这个object的对象锁。结果，其它线程对该object对象所有同步代码部分的访问都被暂时阻塞。

     五、以上规则对其它对象锁同样适用.

**49.线程的状态？**

答：1)、新建状态(New)：新创建了一个线程对象。

　　2)、就绪状态(Runnable)：线程对象创建后，其他线程调用了该对象的start()方法。该 状态的线程位于可运行线程池中，变得可运行，等待获取[CPU](http://product.it168.com/list/b/0217_1.shtml)的使用权。

　　3)、运行状态(Running)：就绪状态的线程获取了CPU，执行run()方法。

　　4)、阻塞状态(Blocked)：阻塞状态是线程因为某种原因放弃CPU使用权，暂时停止运 行。直到线程进入就绪状态，才有机会转到运行状态。阻塞的情况分三种：

　　 (一)、等待阻塞：运行的线程执行wait()方法，JVM会把该线程放入等待池中。

　　 (二)、同步阻塞：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用， 　　(三)、其他阻塞：运行的线程执行sleep()或join()方法，或者发出了I/O请求时， 处理完毕时，线程重新转入就绪状态。

5)、死亡状态(Dead)：线程执行完了或者因异常退出了run()方法，该线程结束生命周 期。

当调用start方法的时候，该线程就进入就绪状态。等待CPU进行调度执行，此时还没有真正执行线程。

当调用run方法的时候，是已经被CPU进行调度，执行线程的主要任务。

**50.onSaveInstanceState() 和 onRestoreInstanceState()；**

答: Activity的 onSaveInstanceState() 和 onRestoreInstanceState()并不是生命周期方法，它们不同于 onCreate()、onPause()等生命周期方法，它们并不一定会被触发。当应用遇到意外情况（如：内存不足、用户直接按Home键）由系统销毁一个Activity时，onSaveInstanceState() 会被调用。但是当用户主动去销毁一个Activity时，例如在应用中按返回键，onSaveInstanceState()就不会被调用。因为在这种情况下，用户的行为决定了不需要保存Activity的状态。通常onSaveInstanceState()只适合用于保存一些临时性的状态，而onPause()适合用于数据的持久化保存。

另外，当屏幕的方向发生了改变， Activity会被摧毁并且被重新创建，如果你想在Activity被摧毁前缓存一些数据，并且在Activity被重新创建后恢复缓存的数据。可以重写Activity的 onSaveInstanceState() 和 onRestoreInstanceState()方法。

**51.了解****版本控制的工具吗？**

版本控制是程序开发、管理必不可少的工具，特别是在多人协作的团队中，适宜的版本控制工具可以提高开发效率，消除很多有代码版本带来的问题

目前最流行的开源版本控制工具要数Subvision（SVN）、Git以及Mercurial（hg）了，其中SVN是集中式版本控制工具，Git和Mercurial则是分布式的。

早期常用的还有CVS

**52.github git都是啥？**

Github 是开源代码库以及版本控制系统

Git是一个开源的分布式版本控制系统，用以有效、高速的处理从很小到非常大的项目版本管理.

**53.git svn 区别？**

git是国外开源版本库，不需要自己搭建服务器，你在上面搭建上传的工程代码都是公开的，谁都可以访问，可以设置团队成员分配修改的权限。

如果要像SVN一样指定的人可以访问Git就需要收费了，SVN需要你有一台服务器，上面安装SVN Server实现版本控制。

**54.****举个简单的****socket编程**

服务器：

try {

ServerSocket server=new ServerSocket(8189);

Socket socket=server.accept();

String line;

BufferedReader is=new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter os=new PrintWriter(socket.getOutputStream());

BufferedReader sin=new BufferedReader(new InputStreamReader(System.in));

System.out.println("客户端："+is.readLine());

line=sin.readLine();

while(!line.equals("bye")){

os.println(line);

os.flush();

System.out.println("服务器："+line);

System.out.println("客户端："+is.readLine());

line=sin.readLine();

}

os.close();

is.close();

socket.close();

server.close();

} catch (IOException e) {

e.printStackTrace();

}

客户端：

try {

//向本机的8189端口发出客户请求

Socket socket=new Socket("127.0.0.1",8189);

//由系统标准输入设备构造BufferedReader对象

BufferedReader sin=new BufferedReader(new InputStreamReader(System.in));

//由Socket对象得到输出流，并构造PrintWriter对象

PrintWriter os=new PrintWriter(socket.getOutputStream());

//由Socket对象得到输入流，并构造相应的BufferedReader对象

BufferedReader is=new BufferedReader(new InputStreamReader(socket.getInputStream()));

String readline;

readline=sin.readLine();//从系统标准输入读入字符串

while(!readline.equals("bye")){

//将从系统标准输入读入的字符串输出到Server

os.println(readline);

//并刷新输出流，使Server马上收到该字符串

os.flush();

System.out.println("客户端："+readline);

//从Server读出字符串，并打印到标准输出设备上

System.out.println("服务器端："+is.readLine());

readline=sin.readLine();

}

os.close();

is.close();

socket.close();

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

**55.怎样用数据库语句实现分页**

一个不带limit 一个带limit。mysql为例首先，连接数据库，写一条sql语句把你要查询的信信息总量查找出来sql =” select count(\*) from tb”;

设定每页显示条数int display=20;

然后，当前页为int page=1;

在写一句sql = “select \* from tb limit “+dispaly\*(page - 1)+”,”+display;

最后,在页面显示分页信息把当前页传回给分页处理页，一定要把相关的条件一起传回去，用get方法传值。

**56.****怎样实现自定义View**

很多时候系统自带的View满足不了设计的要求，就需要自定义View控件。自定义View首先要实现一个继承自View的类。添加类的构造方法，override父类的方法，如onDraw，（onMeasure）等。如果自定义的View有自己的属性，需要在values下建立attrs.xml文件，在其中定义属性，同时代码也要做修改。

**57.****json的上限是多少（大小）**

Json本身没有长度限制，对后台程序来讲，JSON就是一个字符串

如果文件过大的会变成String报错，这时改成流来做就可以了。

**58.afinal框架和xUtils框架的区别**

xUtils 与 afinal 目前的主要区别：

afinal 的View注解要继承FinalActivity, 如果要使用FragmentActivity或其他View自定义视图就要对已有代码进行较大的改造，耦合太强，非常不方便，而xUtils使用静态初始化方法。

xUtils对http模块进行了较大的重构，支持大文件上传，支持7种http谓词，mutipart支持设置subType，下载支持302重定向...

**59.listView上拉加载监听**

OnScrollListener监听

在以下两个方法中做出判断和处理

@Override

public void onScroll(AbsListView view, int firstVisibleItem,

int visibleItemCount, int totalItemCount) {

// 计算最后可见条目的索引

lastVisibleIndex = firstVisibleItem + visibleItemCount - 1;

// 所有的条目已经和最大条数相等，则移除底部的View

if (totalItemCount == MaxDateNum + 1) {

mListView.removeFooterView(moreView);

}

}

@Override  
    public void onScrollStateChanged(AbsListView view, int scrollState) {  
     // 当不滚动时  
     if (scrollState == OnScrollListener.SCROLL\_STATE\_IDLE) {  
      //判断是否滚动到底部  
      if (view.getLastVisiblePosition() == view.getCount() - 1) {  
       adapter.count += 10;   
                   adapter.notifyDataSetChanged();  
                   int currentPage=adapter.count/10;  
                   Toast.makeText(getApplicationContext(), "第"+currentPage+"页", Toast.LENGTH\_LONG).show();  
      }  
     }  
    }

**60.JSON/XML的区别**

1.JSON和XML的数据可读性基本相同

2.JSON和XML同样拥有丰富的解析手段

3.JSON相对于XML来讲，数据的体积小

4.JSON与JavaScript的交互更加方便

5.JSON对数据的描述性比XML较差

6.JSON的速度要远远快于XML

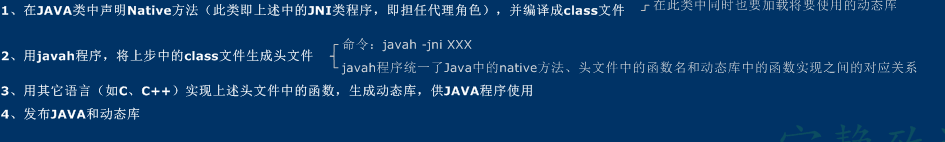
**61.JNI怎样用**

JNI是JAVA标准平台中的一个重要功能，它弥补了JAVA的与平台无关这一重大优点的不足，在JAVA实现跨平台的同时，也能与其它语言（如C、C++）的动态库进行交互，给其它语言发挥优势的机会。

Java Native Interface (JNI)标准是Java平台的一部分， JNI 是本地编程接口，它使得在 Java 虚拟机 (VM) 内部运行的 Java 代码能够与用其它编程语言(如 C、C++ 和汇编语言)编写的应用程序和库进行交互操作。

作用：它允许Java代码和其他语言写的代码进行交互。

优缺点：大大提高程序的性能，但是却破坏了Java的可移植性，也对Java的安全带来了一定的隐患。



**62.广播的两种注册方式**

1)动态注册/非常驻型注册/在代码中进行注册:应用程序关闭后不能再接受广播信息

在应用程序的代码中注册   
注册BroadcastReceiver： registerReceiver(receiver, filter);   
取消注册BroadcastReceiver： unregisterReceiver(receiver);

2)静态注册/常驻型注册/Manifast.xml中注册:应用程序关闭后广播接收器仍能接受广播信息

在AndroidManifest.xml的application里面定义receiver并设置要接收的action,程序中的核心代码示例:

Intent intent = new Intent();

intent.setAction(Intent.ACTION\_EDIT);

sendBroadcast(intent);

**63.android中Serializable和Parcelable的联系？**

相同点:都是Android中自定义对象的序列化

不同点:

1) Serializable序列化时会产生大量的临时变量,因而频繁调用GC,

使用内存时Parcelable比Serializable性能高

2) 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， 也不提倡用，但在这种情况下，还是建议用Serializable

3) Serializable 的实现，只需要继承 implements Serializable 即可

Parcelabel 的实现，需要在类中实现实现writeToParcel、describeContents函数以及静态成员变量 CREATOR，这个变量需要继承 Parcelable.Creator 接口

1. Serializable的作用是为了保存对象的属性到本地文件、数据库、网络流、rmi以方便数据传输，当然这种传输可以是程序内的也可以是两个程序间的。而Android的Parcelable的设计初衷是因为Serializable效率过慢，为了在程序内不同组件间以及不同Android程序间(AIDL)高效的传输数据而设计，这些数据仅在内存中存在，Parcelable是通过IBinder通信的消息的载体。

**64.一个应用中有多少个Context?**

1、创建Application 对象时， 而且整个App共一个Application对象

2、创建Service对象时

3、创建Activity对象时

所以总Context实例个数 = Service个数 + Activity个数 + 1（Application对应的Context实例）

**65.AsyncTask与使用线程处理耗时操作的优劣是？**

AsyncTask:就是一个线程池，调用简单快捷,过程可控,操作灵活,但是更占用系统资源

需要多个异步操作并需要UI变更时就会变得复杂，如果异步任务的数据特别庞大，

AsyncTask这种线程池结构的优势就体现出来了

Thread:涉及到Looper,Thread,Handler,message结构清晰,功能定义明确,对于多个异步任务简单

Handler和Looper机制，它们均基于消息实现，有时可能消息队列阻塞或其他原因无法准确的使用

**66.谈谈你对Android事件分发机制的理解?**

当你点击了某个控件，首先会去调用该控件所在布局的dispatchTouchEvent方法，然后在布局的dispatchTouchEvent方法中找到被点击的相应控件，再去调用该控件的dispatchTouchEvent方法。

**67.谈谈你对ZYGOTE进程的理解和认识?**

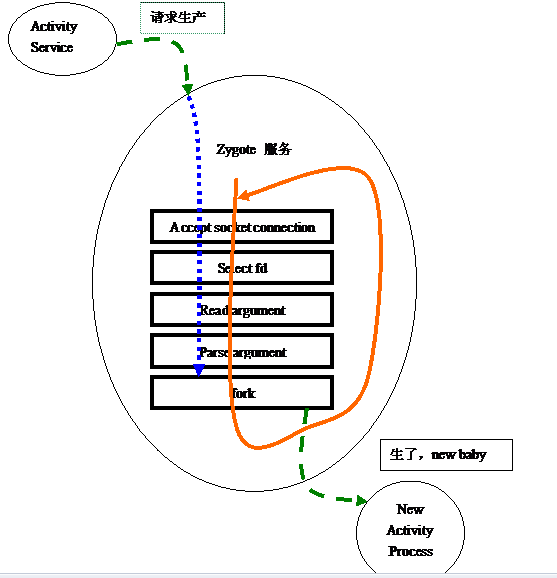
Zygote(英文意思: 孵化)是一个简单的典型C/S结构。其他进程作为一个客服端向Zygote发出请求，Zygote接收到命令就”孵化”出一个Activity进程来。

Zygote系统代码组成及其调用结构：

Zygote.java提供访问Dalvik “zygote”的接口。主要是包装Linux系统的Fork，以建立一个新的VM实例进程。

ZygoteConnection.javaZygote的套接口连接管理及其参数解析。其他Actvitiy建立进程请求是通过套接口发送命令参数给Zygote。

ZygoteInit.javaZygote的main函数入口。



**68.第三方注册及登陆怎么实现的?**

所谓的第三方登录，就是利用用户在第三方平台上已有的账号来快速完成自己应用的登录或者注册的功能。

要数据,不要功能:

1、用户触发第三方登录事件

2、showUser(null)请求授权用户的资料（这个过程中可能涉及授权操作）

3、如果onComplete()方法被回调，将其参数Hashmap代入你应用的Login流程

4、否则提示错误，调用removeAccount()方法，删除可能的授权缓存数据

5、Login时客户端发送用户资料中的用户ID给服务端

6、服务端判定用户是已注册用户，则引导用户进入系统，否则返回特定错误码

7、客户端收到“未注册用户”错误码以后，代入用户资料到你应用的Register流程

8、Register时在用户资料中挑选你应用的注册所需字段，并提交服务端注册

9、服务端完成用户注册，成功则反馈客户端引导用户进入系统

10、否则提示错误，调用removeAccount()方法，删除可能的授权缓存数据

要功能,不要数据:

1、用户触发第三方登录事件

2、调用platform.getDb().getUserId()请求用户在此平台上的ID

3、如果用户ID存在，则认为用户是合法用户，允许进入系统；否则调用authorize()

4、authorize()方法将引导用户在授权页面输入帐号密码，然后目标平台将验证此用户

5、如果onComplete()方法被回调，表示授权成功，引导用户进入系统

6、否则提示错误，调用removeAccount()方法，删除可能的授权缓存数据

**69.Fragment怎么实现的?**

继承Fragment类，重写生命周期方法，主要的不同之处就是需要重写一个onCreateView()方法来返回这个Fragment的布局

在Activity中你可以通过getFragmentManager()来获得Fragment对象，然后通过FragmentManager对象的beginFragmentTransaction()方法来获得FragmentTransaction对象。通过它的add()方法来添加一个Fragment到当前的Activity中。

一个FragmentTransaction对象可以执行多个增删修的方法，如果你想把这些修改提交到Activity上，必须在最后调用一下这个对象的commit()方法

两个单独的Fragment之间是不应该进行通信的。应该使用他们所存在的Activity作为沟通的纽带。

为了实现两个Fragment的交互，您可以在Fragment中定义一个接口，然后再这个接口中定义一个方法，在Fragment的onAttach()方法中调用这个接口中的方法。然后让Activity实现这个方法来完成Activity和Fragment之间的通信

可以通过FragmentManager的findFragmentById()来查找一个Fragment

**70.缓存怎么实现的?**

缓存可分为二大类:

一、通过文件缓存,

顾名思义文件缓存是指把数据存储在磁盘上，不管你是以XML格式，序列化文件DAT格式还是其它文件格式；

二、内存缓存，也就是实现一个类中静态Map,对这个Map进行常规的增删查下面为一个简单的缓存代码

三. 大数据量的缓存肯定还是要保存到文件，这里当然要用到第三方开源框架来实现了，常用的有Oscache,Ehcache,Jcache,Jbosscache等等很多

**71.解析XML PULL解析?**

pull解析同样基于事件驱动型,android 官方API提供,可随时终止,调用next() 方法提取它们（主动提取事件）

Pull解析器运行方式与SAX相似。它提供了类似的事件，如 开始元素事件, 结束元素事件。使用 parser.next()可以进入下一个元素并触发相应事件。事件将作为数值代码被发送，因此可以使用一个 switch对感兴趣的事件进行选择处理。当元素开始解析时，调用 parser.nextText()可以获得下一个Text类型元素的值。

**72.分享用的什么平台？**

让应用里添加第三方分享，主要是腾讯微博与新浪微博、人人网，但是很麻烦。推荐使用友盟第三方开放平台，平台很好，它里面的功能非常多，比如什么渠道分享，问卷调查，异常查看，软件更新等···

**73.横竖屏切换**

1. 在AndroidManifest.xml中为Activity设置configChanges属性，

<application android:icon="@drawable/icon" android:label="@string/app\_name">

<activity android:name=".MainActivity"  
android:label="@string/app\_name" android:configChanges="orientation|keyboardHidden">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

configChanges有如下选项： 1. orientation ：屏幕在纵向和横向间旋转， 2. keyboardHidden：键盘显示或隐藏 ，3.fontScale：用户变更了首选的字体大小 4.locale ： 用户选择了不同的语言设定，5. keyboard ：键盘类型变更，例如手机从12键盘切换到全键盘 6. touchscreen或navigation：键盘或导航方式变化，

如果缺少了keyboardHidden选项 不能防止Activity的销毁，并且在之后提到的onConfigurationChanged事件中 只能捕获竖屏变横屏的事件 不能捕获横屏变竖屏

2. 在对应的Activity中重写：onConfigurationChanged 方法：

public class MainActivity extends Activity {

private TextView textView;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

Log.i("--Main--", "onCreate");

textView=(TextView)findViewById(R.id.tv\_id);

}

@Override

public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig);

Log.i("--Main--", "onConfigurationChanged");

if(newConfig.orientation==Configuration.ORIENTATION\_LANDSCAPE){

textView.setText("当前屏幕为横屏");

}else{

textView.setText("当前屏幕为竖屏");

}

}

}

**74.** **高并发业务请求处理方法**

高并发业务请求处理方法，涉及即时通讯领域。所述方法包括步骤：将业务请求和目的IP压入客户端的分组消息队列；根据所述目的IP，将所述业务请求压入所述分组消息队列的数据存储模块的子消息队列中；将所述子消息队列中的多个业务请求，在所述分组消息队列的网络处理模块中，合并成一个网络请求后发送给所述目的IP对应的服务器。本发明的高并发业务请求处理方法，通过设置分组消息队列，首先将业务请求按照目的IP进行分组，然后将同组的业务请求合并成网络请求后再发送给服务器。从而，减少了网络请求次数，提高了对业务请求的处理效率，节省客户端和服务器内存。

**75.Android中如何设置两个应用程序为同一个进程？**

1.在<manifest>里面添加： android:sharedUserId，注意这个属性的取值必须包含点（dot），也就是诸如java packae的形式。比如com.aaa.bbb。没有dot的话，将来adb install xxx.apk就会出错：Failure [INSTALL\_PARSE\_FAILED\_BAD\_SHARED\_USER\_ID]。最晕死的就是这一点在Android文档中没有提到，感谢万能的google赐予了我答案。所有application都要填写的一样。

2.在<manifest>里面添加：android:sharedUserLabel="@string/shared\_user\_label"，这个label必须是一个string资源，不能是raw string。所有application都要填写的一样。

3.在<application>里面添加：android:process="xxx.xxx.xxx"，这里所有的application都要填写的一样，内容就是process的名字，一般来说Android中process的名字就是manifest中的package的取值。

4.所有的application用同样的一个key来sign。如果用Eclipse ADT plugin开发，由于所有的application都使用同一个debug key来sign，所以这一步没有什么额外的工作。更具体的有关sign apk的细节，参考Android文档：http://androidappdocs.appspot.com/guide/publishing/app-signing.html

**76.手机适配怎么做？**

在多平台适配中遵循如下原则

1） 客户端的设计规范应该以平台规范为基础

2） 在多平台中，品牌标识，包括logo，视觉风格，核心功能点等等应保持统一。

3） 更多地与平台的特性相融合，运用平台提供的特色功能，来减少用户的输入或者其他体验提升点。如拍照输入，传感器的使用等。

其他需要注意的问题：

1） 移动的特性决定了手机信号的强弱不均，如何处理在弱信号下的设计？

2） 需要考虑在断网情况下，如何快速恢复中断？

3） 如何设计手机推送的问题？

4） 如何尽量避免手机的固有限制，如屏幕小，输入不方便，电源紧张等？

5） 如何尽量通过手机的特有特性来提升体验，如各类传感器，声音提示等？

Android系统采用以下方法来实现应用界面的自动适配：

1）布局文件中定义长度的时候，最好使用wrap\_content,fill\_parent, 或者dp 进行描述，这样可以保证在屏幕上面展示的时候有合适的大小

2）像素单位都使用DIP，文本单位使用SP

3）为不同屏幕密度的手机，提供不同的位图资源，可以使得界面清晰无缩放。

对应bitmap 资源来说，自动的缩放有时会造成放大缩小后的图像变得模糊不清，这是就需要应用为不同屏幕密度配置提供不同的资源：为高密度的屏幕提供高清晰度的图像等。

1. 尽量不要使用AbsoluteLayout

77.[手机测试](http://baike.baidu.com/view/107025.htm)包括哪些？

[手机测试](http://baike.baidu.com/view/107025.htm)是一个很大的题目，涉及到硬件测试和[软件测试](http://baike.baidu.com/view/16563.htm)，还有结构的测试，比如抗压，抗摔，抗[疲劳](http://baike.baidu.com/view/492409.htm)，抗低温高温等。

**硬件测试**一般都有严格的物理电气指标，也有专门的 [仪器](http://baike.baidu.com/view/56517.htm)，一般都有明确的指标，相应的[测试用例](http://baike.baidu.com/view/106882.htm)，严格的操作规程，还有各种[仪器](http://baike.baidu.com/view/56517.htm)，一定的测试[软件](http://baike.baidu.com/view/37.htm)。

[手机测试](http://baike.baidu.com/view/107025.htm)，一般是指[**软件**](http://baike.baidu.com/view/37.htm)**测试**。

[**软件**](http://baike.baidu.com/view/37.htm)**测试有：**

集成版上的IT测试，对日的事业者测试，ST（系统）测试，还有协议测试（一般针对3G手机的多媒体画质和视频电话），底层的基站相关的测试，各种外联设备（电压，电流，模拟器）相关的测试。还就是FT，FT大多数翻译成场测。

每一种测试的流程都是不同的。

但是**基本的流程**如下：

1.测试需求分析

2.测试计划/测试用例书写

3.测试环境搭建，测试数据准备以及测试执行

4.测试反馈。一般指测试过程中发现异常，进行缺陷分析定位，然后和相关的开发人员进行对应，最后促使问题得到解决的过程。（测试faxianbug->评估提交bug->开发人员bug定位解决，与开发人员针对bug进行沟通，再现->bug修复之后的再测试，确定bug修复。)

5.测试项目结束后的总结分析（有的是没有这个环节的）

6.回归测试，所有测试结束之后，在版本提交之前必须进行的。

**测试过程中很注重的几点是**：

第一，要理解需求，不理解需求，不明白功能，测试无从谈起。

第二，设计测试用例要用心到位，测试是设计出来的，不是凭空臆测的。

第三，测试过程中的沟通和报告。包括自己不懂得业务，如何去向成手或者Leader学习业务，包括测试过程中对于不明白的功能的理解，不正常的现象的分析，bug提交前后和开发人员之间的交流等。

**78.Http中get和post的区别?**

1. get是从服务器上获取数据，post是向服务器传送数据。  
2. get是把参数数据[队列](http://zhidao.baidu.com/search?word=队列&fr=qb_search_exp&ie=utf8)加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。  
3. 对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。  
4. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。  
5. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。

**79.侧滑怎么实现的?**

侧滑菜单实现方式有多种 具体自己项目里面用到的方法

思路：a.自定义ViewGroup ，重写构造函数，重写onLayout（）方法 对三个菜单页面重新排布

b.通过重写onTouchEvent方法，通过onSrcoll 使三个菜单页面左右切换

答：实现侧滑的步骤：

　　要实现ViewPager+Fragment的滑动效果，必须继承FragmentActivity，同时要

给ViewPager设置Adapter。由于ViewPager上面要加载Fragment（需要继承Fragment

类），所以不是重载PagerAdapter,而是重写FragmentPagerAdapter类

　　写好了adapter，在主函数的onCreate（）函数里面初始化ViewPager,同时给

它配置adapter和设置滑动事件监听（这需要实现OnPageChangeListener接口）

　　接下来要做的就是在onCreate()里面初始化RadioButton，并给它设置

OnClickListener监听点击事件

最后就是实现侧滑出左边页面的效果，这里使用了开源组件SlidingMenu，

SlidingMenu组件重写了Activity、FragmentActivity、RelativeLayout等，要实现侧滑

效果，可以把主函数继承的FragmentActivity改成SlidingFragmentActivity，也可以

去在xml布局文件里面直接设置

**80.UDP和TCP的区别?**

TCP是基于连接的协议，也就是说，在正式收发数据前，必须和对方建立可靠的连接。

UDP它是面向非连接的协议，它不与对方建立连接，而是直接就把数据包发送过去！ 用于传输少量数据(数据包模式).

TCP可靠性高更安全，UDP不安全但速度快

**81.双缓存怎么实现的?**

答：1、在内存中建立一块“虚拟画布”：  
Bitmap bmp = new Bitmap(600, 600);  
2、获取这块内存画布的Graphics引用：  
Graphics g = Graphics.FromImage(bmp);  
3、在这块内存画布上绘图：  
g.FillEllipse(brush, i \* 10, j \* 10, 10, 10);  
4、将内存画布画到窗口中  
this.CreateGraphics().DrawImage(bmp, 0, 0);

**82.垃圾收集算法的核心思想**

答：Java语言建立了垃圾收集机制，用以跟踪正在使用的对象和发现并回收不再使用(引用)的对象。该机制可以有效防范动态内存分配中可能发生的两个危险：因内存垃圾过多而引发的内存耗尽，以及不恰当的内存释放所造成的内存非法引用。

　　垃圾收集算法的核心思想是：对虚拟机可用内存空间，即堆空间中的对象进行识别，如果对象正在被引用，那么称其为存活对象，反之，如果对象不再被引用，则为垃圾对象，可以回收其占据的空间，用于再分配。垃圾收集算法的选择和垃圾收集系统参数的合理调节直接影响着系统性能，因此需要开发人员做比较深入的了解。

**83.触发主GC(Garbage Collector)的条件**

答：①当应用程序空闲时,即没有应用线程在运行时,GC会被调用。因为GC在优先级最低的线程中进行,所以当应用忙时,GC线程就不会被调用,但以下条件除外。

　　②Java堆内存不足时,GC会被调用。当应用线程在运行,并在运行过程中创建新对象,若这时内存空间不足,JVM就会强制地调用GC线程,以便回收内存用于新的分配。若GC一次之后仍不能满足内存分配的要求,JVM会再进行两次GC作进一步的尝试,若仍无法满足要求,则 JVM将报“out of memory”的错误,Java应用将停止。

**84.减少GC开销的措施**

答：

(1)不要显式调用System.gc()

(2)尽量减少临时对象的使用

(3)对象不用时最好显式置为Null

(4)尽量使用StringBuffer,而不用String来累加字符串

(5)能用基本类型如Int,Long,就不用Integer,Long对象

(6)尽量少用静态对象变量

(7)分散对象创建或删除的时间

**85.gc与finalize方法**

答：finalize()，其工作原理：一旦[垃圾回收](http://zhidao.baidu.com/search?word=%E5%9E%83%E5%9C%BE%E5%9B%9E%E6%94%B6&fr=qb_search_exp&ie=utf8)器准备好释放对象占用的存储空间，将首先调用其finalize()方法，并且在下一次垃圾回收动作发生时，才会真正回收对象占用的内存。所以如果用finalize()就能在垃圾回收时刻做一些重要的清理工作。

gc 只能清除在堆上分配的内存(纯java语言的所有对象都在堆上使用new分配内存),而不能清除栈上分配的内存（当使用JNI技术时,可能会在栈上分配内 存,例如java调用[c程序](http://zhidao.baidu.com/search?word=c%E7%A8%8B%E5%BA%8F&fr=qb_search_exp&ie=utf8)，而该[c程序](http://zhidao.baidu.com/search?word=c%E7%A8%8B%E5%BA%8F&fr=qb_search_exp&ie=utf8)使用malloc分配内存时）.因此,如果某些对象被分配了栈上的内存区域,那gc就[管不着](http://zhidao.baidu.com/search?word=%E7%AE%A1%E4%B8%8D%E7%9D%80&fr=qb_search_exp&ie=utf8)了,对这样的对象进行 内存回收就要靠finalize().

**86.ViewPager如何实现？**

**答：**a.Viewpager是一个控件，用的时候直接在布局中定义引用（Android.support.v4.view.ViewPager）即可。 b.ViewPager里面显示的是多个View，用LayoutInflater的对象调用inFlater()方法定义布局，并把View对象添加到List<View>集合中。

c.定义适配器继承PagerAdapter，设置给ViewPager。

**87.请说出使用equal和==比较对象时的区别？**

**答：**“==”比较的是2个对象的地址，而equals比较的是2个对象的内容。例如两个String类型的对象：String s1 = new String(“str”);

String s2 = new String(“str”);

用“==”比较的时候返回的是false，因为创建了两个对象，在内存中的地址不一样，用equals则返回true。

**88.什么是Java序列化和反序列话，如何实现Java序列化？**

**答：1）**序列化是将对象状态转化成可保持或传输的格式的过程。与序列化相反的是反序列化，它是将流转化为对象。

**2）**实现序列化必须要实现Serializable接口，目的是声明此类是可以被序列化的，而且继承此类的子类也可以自动的被序列化。

**89.请解释下android程序运行时权限与文件系统权限的区别？**

**答：**apk程序是运行在虚拟机上的，对应的是Android中独有的权限机制，是由Dalvik授权的，而文件系统权限是由linux内核授权的，只有体现到文件系统上时才使用linux的权限设置。

**90.讲一讲overload和override的区别，overloaded的方法是否可以改变返回值的类型？**

**答：**overload是方法的重载，Java中的编译时多态，重载的方法具有相同的方法名，但是参数列表不同（参数的类型和数量），对返回值类型没有要求，可以改变；

override是方法得重写，Java中的运行时多态，为子类在继承父类是重写父类中的方法，方法名，返回值类型和参数列表都相同。

**91.如何使用socket实现TCP点对点通信？**

**答：**服务器端监听：首先实例化ServerSocket，并传入端口号（1001-65535），然后用ServerSocket对象调用accept()方法监听，一旦取得连接则获得socket连接对象的客户端。

客户端：实例化Socket，传入服务器的ip和port，一旦连接成功后，Socket对象就可以获得输入、输出流，从而进行通信。当通信完成后，用Socket对象调用close()方法关闭连接，完成一次完整的 socket连接。

**92.请简述service可能被kill的场景，kill之后如何自启？**

在运行onStartCommand后service进程被kill后，系统将会再次启动service，并传入最后一个intent给onstartCommand。直到调用stopSelf(int)才停止传递intent。如果在被kill后还有未处理好的intent，那被kill后服务还是会自动启动。因此onstartCommand不会接收到任何null的intent

**93.请使用递归方式来遍历盘下的所有文件，并计算出所有图片文件的数量？**

public class F3 {

public static void main(String[] args) {

File f1=new File("D://");

m1(f1);

}

static void m1(File f1) {

// TODO Auto-generated method stub

if(f1.exists()){

File file[]=f1.listFiles();

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

if(file[i].isFile()){

System.out.println(file[i]);

}else{

m1(file[i]);

}

}

}else{

System.out.println("不存在！！！！");

}

}

}

**94.现有两个单向链表，我想知道这两个链表的相交情况。（可以不写实现，但思路必须写清）？**

答：方法一：直接法

直接判断第一个链表的每个结点是否在第二个链表中，时间复杂度为O(len1\*len2)，耗时很大

方法二：利用计数

如果两个链表相交，则两个链表就会有共同的结点；而结点地址又是结点唯一标识。因而判断两个链表中是否存在地址一致的节点，就可以知道是否相交了。可以对第一 个链表的节点地址进行hash排序，建立hash表，然后针对第二个链表的每个节点的地址查询hash表，如果它在hash表中出现，则说明两个链表有共 同的结点。这个方法的时间复杂度为：O(max(len1+len2)；但同时还得增加O(len1)的存储空间存储哈希表。这样减少了时间复杂度，增加 了存储空间。

以链表节点地址为值，遍历第一个链表，使用Hash保存所有节点地址值，结束条件为到最后一个节点（无环）或Hash中该地址值已经存在（有环）。

再遍历第二个链表，判断节点地址值是否已经存在于上面创建的Hash表中。

这个方面可以解决题目中的所有情况，时间复杂度为O(m+n)，m和n分别是两个链表中节点数量。由于节点地址指针就是一个整型，假设链表都是在堆中动态创建的，可以使用堆的起始地址作为偏移量，以地址减去这个偏移量作为Hash函数

方法三

两个没有环的链表相交于一节点，则在这个节点之后的所有结点都是两个链表所共有的。如果它们相交，则最后一个结点一定是共有的，则只需要判断最后一个结点是否相同即可。时间复杂度为O(len1+len2)。对于相交的第一个结点，则可求出两个链表的长度，然后用长的减去短的得到一个差值 K，然后让长的链表先遍历K个结点，然后两个链表再开始比较。还可以这样：其中一个链表首尾相连，检测另外一个链表是否存在环，如果存在，则两个链表相交，而检测出来的依赖环入口即为相交的第一个

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\* 两个不含环的单链表的相交  
相交指的是结点的地址相同，而不是结点的值相同 \*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**95.使用javac/c++写一段程序，找出数组中出现次数最多数字，并输出出现次数，请标明算法的时间复杂值？**

import java.util.HashMap;

import java.util.Iterator;

public class Test

{

public static void main(String[] args)

{

int a [] = {6,6,6,1,2,8,8,5,5,5,12};

int maxnum = 0;

int maxvalue = 0;

HashMap<Integer,Integer> map = new HashMap<Integer,Integer>();

HashMap<Integer,Integer> map2 = new HashMap<Integer,Integer>();

for(int i=0;i<a.length;i++)

{

int count = 0;

for(int j=0;j<a.length;j++)

{

if(a[i]==a[j])

{

count++;

}

}

map.put(a[i], count);//map會把重複的數據過濾掉

}

Iterator it = map.keySet().iterator();

Iterator it2 = map.keySet().iterator();

while(it.hasNext())

{

Object key = it.next();

if(map.get(key)>maxnum)

{

maxnum = map.get(key);

maxvalue = (Integer)key;

}

}

map2.put(maxvalue, maxnum);

while(it2.hasNext())

{

Object key = it2.next();

if(map.get(key)==maxnum)

{

map2.put((Integer)key, maxnum);

}

}

Iterator it3 = map2.keySet().iterator();

while(it3.hasNext())

{

Object key = it3.next();

System.out.print("出現次數最多的數為:"+key+" ");

System.out.println("出現次數:"+map2.get(key));

}

}}

**96.什么是OAuth，Oauth，的角色，Qauth验证流程？**

答：<1>OAuth(开放授权)是一个开放标准，允许用户让第三方应用访问该用户在某一网站上存储的私密的资源(如照片，视频，联系人列表)，而无需将用户名和密码提供给第三方应用。

OAuth是个安全相关的协议，作用在于，使用户授权第三方的[应用程序](http://baike.so.com/doc/3417785.html)访问用户的web资源，并且不需要向第三方应用程序透露自己的密码。

OAuth允许用户提供一个令牌，而不是用户名和密码来访问他们存放在特定服务提供者的数据。每一个令牌授权一个特定的网站(例如，视频编辑网站)在特定的时段(例如，接下来的2小时内)内访问特定的资源(例如仅仅是某一相册中的视频)。这样，OAuth允许用户授权第三方网站访问他们存储在另外的服务提供者上的信息，而不需要分享他们的访问许可或他们数据的所有内容。

OAuth是OpenID的一个补充，但是完全不同的服务

<2>OAuth协议中包含了三个角色：  
**Service Provdier**，即服务提供者，如新浪微博；  
**User**，即普通用户，如新浪微博用户；  
**Consumer**，即第三方应用，如本人开发的应用

<3>我在第三方应用A有需要帐号、密码登录后才能访问的一些数据，我想让用户登录腾讯的AlloyOS就可以访问这些数据，但是我不想把密码告诉腾讯，这时候就可以通过腾讯的QAuth来解决，给AlloyOS一个token，当用户确认绑定账号之后，用户只要登录的AlloyOS，打开第三方应用A时就通过token直接登录的第三方应用。

**97.使用javac/c++写一段程序，找出数组中第K大小的数，输出数所在的位置，请标明算法的时间复杂度？**

答：import java.util.ArrayList;

public class Biaoshi{

public static void main(String args[]){

ArrayList<String> al=new ArrayList<String>();

//向Java动态数组中添加数据

al.add("a");

al.add("b");

al.add("c");

//输出Java动态数组

int j=0;

for(int i=0;i<al.size();i++)

{

j++;

String k=(String)al.get(i);

System.out.println(k+"--------"+j);

}

//删除数组中的某个元素,删除第二个元素

al.remove(1);

//修改Java动态数组，把新的元素放到第二个位置

al.add(1,"2");

////输出Java动态数组

for(int i=0;i<al.size();i++)

{

String k=(String)al.get(i);

System.out.println(k);

}

}

}

**98.进程通信和线程同步的方法？**

答：进程通信方法：

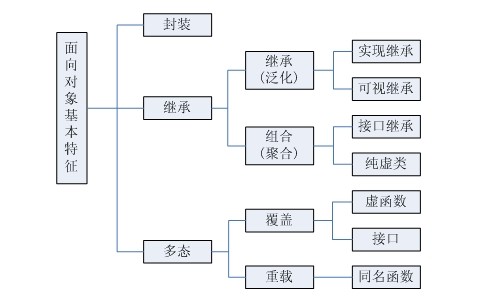
1. 信号(signal)  
   2.管道(pipe):父子进程  
   3.命名管道:mkfifo  
   4.socket  
   5.消息队列:msgget/msgctl/msgrcv/msgsnd  
   6.共享内存:shmget/shmat/shmdt/shmctl  
   7.信号灯:semget/semctl/semop/  
   8.锁文件(flock)

9. DDE是一种动态数据交换机制(Dynamic Data Exchange，DDE)。使用DDE通讯需要两个Windows应用程序，其中一个作为服务器处理信息，另外一个作为客户机从服务器获得信息。客户机应用程序向当前所激活的服务器应用程序发送一条消息请求信息，服务器应用程序根据该信息作出应答，从而实现两个程序之间的数据交换。

1. 线程同步方法：  
   1.互斥量Mutex  
   2.信号灯Semophore  
   3.条件变量Conditions

**99.面向对象的特征有哪些方面？**

面向对象的三个基本特征是：封装、继承、多态。



封装：封装最好理解了。封装是面向对象的特征之一，是对象和类概念的主要特性。封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

继承：面向对象编程 (OOP) 语言的一个主要功能就是“继承”。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

通过继承创建的新类称为“子类”或“派生类”。

被继承的类称为“基类”、“父类”或“超类”。

继承的过程，就是从一般到特殊的过程。

要实现继承，可以通过“继承”（Inheritance）和“组合”（Composition）来实现。

在某些 OOP 语言中，一个子类可以继承多个基类。但是一般情况下，一个子类只能有一个基类，要实现多重继承，可以通过多级继承来实现。

多态：多态性（polymorphisn）是允许你将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。

实现多态，有二种方式，覆盖，重载。

覆盖，是指子类重新定义父类的虚函数的做法。

重载，是指允许存在多个同名函数，而这些函数的参数表不同（或许参数个数不同，或许参数类型不同，或许两者都不同）。

那么，多态的作用是什么呢？我们知道，封装可以隐藏实现细节，使得代码模块化；继承可以扩展已存在的代码模块（类）；它们的目的都是为了——代码重用。而多态则是为了实现另一个目的——接口重用！多态的作用，就是为了类在继承和派生的时候，保证使用“家谱”中任一类的实例的某一属性时的正确调用

**100.Context里面主要包括什么具体的东西？**

答：Context提供了关于应用环境全局信息的接口。它是一个抽象类，它的执行被Android系统所提供。它允许获取以应用为特征的资源和类型。同时启动应用级的操作，如启动Activity，broadcasting和接收intents。

**101.简述条形码的扫描设计思路?**

生成步骤:

1、获取需要生成图形的数据

2、根据Zxing提供的图形算法进行图片的生成

3、展示给用户看

扫描步骤：

1、开启扫描，通过startActivityforResult

2、初始化surfaceView、Camera、自定义View

3、开启DecodeThread去获取并解析自定义View里的图片，通过CaptureActivityHandler来处理解析结果

4、当解析成功，handler发送回到主线程，执行返回上一个页面将解析出来的值返回，通过setResult方法返回

**102. ListView异步加载图片实现思路(优化篇)**

异步加载图片基本思想：

1.先从内存缓存中获取图片显示（内存缓冲）

2.获取不到的话从SD卡里获取（SD卡缓冲）

3.都获取不到的话从网络下载图片并保存到SD卡同时加入内存并显示（视情况看是否要显示）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | |  | |  | |  | |
| 面试题\_Android  1. Overload和Override的区别   Overloaded的方法是否可以改变返回值的类型?  方法的重写Overriding和重载Overloading是Java多态性的不同表现。重写Overriding是父类与子类之间多态性的一种表现，重载Overloading是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被"屏蔽"了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。Overloaded的方法是可以改变返回值的类型。   1. String 和StringBuffer的区别   答: String的长度是不可变的，STRINGBUFFER的长度是可变的。如果你对字符串中的内容经常进行操作，特别是内容要修改时，那么使用StringBuffer，如果最后需要String，那么使用StringBuffer（缓存）的toString()方法。  StringBuilder：节约内存，创建字符串；   1. 实现一个字符串倒序   字符串“abcde”通过写一个函数不让调用第三方的字符串，实现一个字符串倒序，比如字符串“abcde” 变成“edcba”  答： String src = "ABCDEF "; String dst = new StringBuffer(src).reverse().toString();   1. 抽象类与接口的区别（abstract与interface的区别）   答：abstract可以修饰抽象方法，而一个类只要有一个抽象方法，就必须用abstract定义该类，即抽象类。抽象类，被继承，实现它里面的方法，可以只有普通方法  用interface修饰的类，里面的方法都是抽象方法（不能实例化），因此在定义接口的时候，可以直接不加那些修饰，系统会默认的添上去。接口里面的字段都是公有常量，即public static final修饰的字段。可以没有函数原型，里面可以是空的   1. 集合的实现类与区别 collections集合工具类，提供了集合一系列的排序方法   答:Collection接口，集合结构总的父接口，有两个子接口list和set  List接口 元素有序可重复.  实现类有：ArrayList 数组实现轻量级，运行快，线程不安全。JDK1.2 查询快  Vector 数组实现重量级，运行慢，线程安全。JDK1.0  LinkedList链表实现 常用语堆栈与队列的实现 增删操作快  Set 接口 元素无序不可重复  实现类有：HashSet，底层用hashCode()算法实现，保证元素的无序唯一，自定义对象存进HashSet为了保证元素内容不重复需要覆盖hashCode()与equals()方法。散列表（Hash table，也叫哈希表），是根据关键码值(Key value)而直接进行访问的数据结构  SortedSet(不重要) 元素有序（Unicode升序）唯一  TreeSet要求元素有序，自定义的对象需要实现Comparable接口的 compareTo（object o）方法  Map(接口): 与Collection接口无关,有一个子接口SortedMap特点: 元素是key-value, key  唯一,无序; value可重复  实现类: HashMap 轻量级 线程不安全的,允许key或value为null JDK1.2  HashTable 重量级 线程安全的 不允许key或value为null JDK1.0  Properties是HashTable的子类,主键和值都是字符串  SortedMap:(不重要)  特点: key唯一,有序(Unicode升序)  实现类:TreeMap   1. 线程有几种状态,分别是哪些（调用run()和调用start()的区别）   答：1)新建状态(New)：新创建了一个线程对象。  　　2)就绪状态(Runnable)：线程对象创建后，其他线程调用了该对象的start()方法。该状态的线程位于可运行线程池中，变得可运行，等待获取[CPU](http://product.it168.com/list/b/0217_1.shtml)的使用权。  　　3)运行状态(Running)：就绪状态的线程获取了CPU，执行run()方法。  　　4)阻塞状态(Blocked)：阻塞状态是线程因为某种原因放弃CPU使用权，暂时停止运行。直到线程进入就绪状态，才有机会转到运行状态。阻塞的情况分三种：  ①等待阻塞：运行的线程执行wait()方法，JVM会把该线程放入等待池中。  ②同步阻塞：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池中。  ③其他阻塞：运行的线程执行sleep()或join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪状态。  5)死亡状态(Dead)：线程执行完了或者因异常退出了run()方法，该线程结束生命周期。  当调用start方法的时候，该线程就进入就绪状态。等待CPU进行调度执行，此时还没有真正执行线程。  当调用run方法的时候，是已经被CP U进行调度，执行线程的主要任务。   1. 线程的实现方式   答：线程的实现有两种方式，一是继承Thread类，二是实现Runnable接口   1. sleep() 与 wait()的区别   答：①这两个方法来自不同的类分别是，sleep来自Thread类，和wait来自Object  类。  ②最主要是sleep方法没有释放锁，而wait方法释放了锁，使得其他线程可  以使用同步控制块或者方法。sleep不出让系统资源；wait是进入线程等待池  等待，出让系统资源，其他线程可以占用CPU。一般wait不会加时间限制，  因为如果wait线程的运行资源不够，再出来也没用，要等待其他线程调用  notify/notifyAll唤醒等待池中的所有线程，才会进入就绪队列等待OS分配系  统资源。sleep(milliseconds)可以用时间指定使它自动唤醒过来，如果时间不到  只能调用interrupt()强行打断。  ③.wait，notify和notifyAll只能在同步控制方法或者同步控制块里面使用，而  sleep可以在任何地方使用  ④. Sleep需要捕获异常,而wait不需要   1. 线程中wait，join，sleep，yield, notify，notifyall，synchronized，区别及联系   答：1).sleep()方法  在指定时间内让当前正在执行的线程暂停执行，但不会释放“锁标志”。不推荐使用。sleep()使当前线程进入阻塞状态，在指定时间内不会执行。  2).wait()方法  在其他线程调用对象的notify或notifyAll方法前，导致当前线程等待。线程会释放掉它所占有的“锁标志”，从而使别的线程有机会抢占该锁。  唤醒当前对象锁的等待线程使用notify或notifyAll方法,waite() 和notify()必须在synchronized函数或synchronized　block中进行调用。3.yield方法  暂停当前正在执行的线程对象。yield()只是使当前线程重新回到可执行状态，所以执行3)yield()的线程有可能在进入到可执行状态后马上又被执行。yield()只能使同优先级或更高优先级的线程有执行的机会。  4).join方法  等待该线程终止。等待调用join方法的线程结束，再继续执行。如：t.join();//主要用于等待t线程运行结束，若无此句，main则会执行完毕，导致结果不可预测。   1. Final、finally、finanlize()的区别   答：final用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。  finally是异常处理语句结构的一部分，表示总是执行。  finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。   1. J2EE是什么？请对以下在J2EE中常用的名词进行解释(或简单描述)   J2EE是Sun公司提出的多层(multi-diered),分布式(distributed),基于组件(component-base)的企业级应用模型(enterprieseapplicationmodel).在这样的一个应用系统中，可按照功能划分为不同的组件，这些组件又可在不同计算机上，并且处于相应的层次(tier)中。所属层次包括客户层(clietntier)组件,web层和组件,Business层和组件,企业信息系统(EIS)层。   1. java常用的设计模式？说明工厂模式   Java中的23种设计模式：  Factory（工厂模式），Builder（建造模式），FactoryMethod（工厂方法模式），  Prototype（原始模型模式），Singleton（单例模式），Facade（门面模式），  Adapter（适配器模式），Bridge（桥梁模式），Composite（合成模式），  Decorator（装饰模式），Flyweight（享元模式），Proxy（代理模式），  Command（命令模式），Interpreter（解释器模式），Visitor（访问者模式），  Iterator（迭代子模式），Mediator（调停者模式），Memento（备忘录模式），  Observer（观察者模式），State（状态模式），Strategy（策略模式），  TemplateMethod（模板方法模式），ChainOfResponsibleity（责任链模式）  工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某一个类的实例，通常这一组类有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后需要定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。   1. 开发中都用到了那些设计模式?用在什么场合   每个模式都描述了一个在我们的环境中不断出现的问题，然后描述了该问题的解决方案的核心。通过这种方式，你可以无数次地使用那些已有的解决方案，无需在重复相同的工作。主要用到了MVC的设计模式。用来开发JSP/Servlet或者J2EE的相关应用。简单工厂模式等。   1. 常用设计模式及应用场景   常用设计模式及应用场景，用两种方式实现单例模式，要求线程安全  答: 常用设计模式:  单例模式: Calendar实例的获取  适配器模式: Adapter 为ListView GridView等添加数据  工厂模式: Spring IOC 反转控制  代理模式: Spring AOP 面向切面编程  观察者模式: ContentObserver监听内容改变  （懒汉式）程序执行过程中需要这个类的对象时再实例化该类的对象  步骤  1)定义静态私有对象  2)构造方法私有化保证在类的外部无法实例化该类的对象  3)定义对外开放的静态方法在调用方法是判断对象是否为空，为空再创建对象返回  public class Singleton {  private static Singleton singleton;  // 构造方法私有化，保证在类的外部无法实例化该类的对象  private Singleton() {  }  public static synchronized Singleton getSingletonInstance() {  if (singleton == null) {  singleton = new Singleton();  }  return singleton;  }  }  （饿汉式）类加载的时候就实例化该类的对象  public class Singleton {  private static Singleton singleton = new Singleton();  // 构造方法私有化，保证在类的外部无法实例化该类的对象  private Singleton() {  }  public static Singleton getSingletonInstance() {  return singleton;  }  }   1. 广播如何调用，有什么方式，各自的区别，实现广播的意图是什么，哪里用到,有序广播，无序广播?   答：程序中发送广播通过sendBroadcastReceiver（）实现  接收广播通过定义一个类继承BroadcastReceiver并重写onReceive（）方法实现  注册广播有两种方式：  第一种静态方式:在清单文件中通过<receive>标签声明  第二种代码动态方式:  IntentFilter filter = new IntentFilter("android.provider.Telephony.SMS\_RECEIVED");  IncomingSMSReceiver receiver = new IncomgSMSReceiver();  registerReceiver(receiver.filter);  1)第一种不是常驻型广播，也就是说广播跟随activity的生命周期。注意: 在activity结束前，移除广播接收器。  2)第二种是常驻型，也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。  BroadcastReceiver所对应的广播分两类：普通广播和有序广播。  普通广播通过Context.sendBroadcast()方法来发送。它是完全异步的。  所有的receivers接收器的执行顺序不确定。 因此，所有的receivers接收器接收broadcast的顺序不确定。  有序广播，即从优先级别最高的广播接收器开始接收，接收完了如果没有丢弃，就下传给下一个次高优先级别的广播接收器进行处理，依次类推，直到最后。   1. android系统架构   答：  1）应用程序层 java语言 应用程序开发   1. 应用程序框架层 java语言 OS定制 framework层开发 2. 系统运行库层 C C++ 实现 so库 3. Linux内核层 4. Android的四大组件是什么？它们的作用是什么   答:  Android有四大组件：Activity、Service、Broadcast Receiver、Content Provider。  Activity :应用程序中，一个Activity通常就是一个单独的屏幕，它上面可以显示一些控件也可以监听并处理用户的事件做出响应。Activity之间通过Intent进行通信。  Service 服务:一个Service 是一段长生命周期的，没有用户界面的程序，可以用来开发如监控类程序。  BroadcastReceive广播接收器:你的应用可以使用它对外部事件进行过滤只对感兴趣的外部事件(如当电话呼入时，或者数据网络可用时)进行接收并做出响应。广播接收器没有用户界面。然而，它们可以启动一个activity或serice 来响应它们收到的信息。  Content Provider内容提供者 :主要用于多个应用间数据共享。这些数据可以存储在文件系统中或SQLite数据库。   1. Activity生命周期？保存activity的一些信息在哪个生命周期方法中   横竖屏切换activity的生命周期,点击Home生命周期,弹出dialog  答：共有七个周期函数：  void onCreate(Bundle savedInstanceState) 第一次创建时调用  void onStart() 被用户可见时调用  void onResume() 当获得焦点即可与用户交互时调用  void onPause() 当失去焦点时调用  void onStop() 当不可见时调用  void onRestart() 当Activity处于stop状态又被重新启动时调用  void onDestroy() 当销毁时调用   1. Activity的 onSaveInstanceState() 和 onRestoreInstanceState()   答: Activity的 onSaveInstanceState() 和 onRestoreInstanceState()并不是生命周期方法，它们不同于 onCreate()、onPause()等生命周期方法，它们并不一定会被触发。当应用遇到意外情况（如：内存不足、用户直接按Home键）由系统销毁一个Activity时，onSaveInstanceState() 会被调用。但是当用户主动去销毁一个Activity时，例如在应用中按返回键，onSaveInstanceState()就不会被调用。因为在这种情况下，用户的行为决定了不需要保存Activity的状态。通常onSaveInstanceState()只适合用于保存一些临时性的状态，而onPause()适合用于数据的持久化保存。  另外，当屏幕的方向发生了改变， Activity会被摧毁并且被重新创建，如果你想在Activity被摧毁前缓存一些数据，并且在Activity被重新创建后恢复缓存的数据。可以重写Activity的 onSaveInstanceState() 和 onRestoreInstanceState()方法。   1. activity在屏幕旋转时的生命周期   答：不设置Activity的android:configChanges时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次；设置Activity的android:configChanges="orientation"时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次；设置Activity的android:configChanges="orientation|keyboardHidden"时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法   1. Android中asset文件夹和raw文件夹区别   答：res/raw和assets的相同点：  两者目录下的文件在打包后会原封不动的保存在apk包中，不会被编译成二进制。  res/raw和assets的不同点：  1）res/raw中的文件会被映射到R.java文件中，访问的时候直接使用资源ID即 R.raw.filename；assets文件夹下的文件不会被映射到R.java中，访问的时候需要AssetManager类。  2）res/raw不可以有目录结构，而assets则可以有目录结构，也就是assets目录下可以再建立文件夹  3）读取文件资源举例：  读取res/raw下的文件资源，通过以下方式获取输入流来进行写操作  InputStream is = getResources().openRawResource(R.raw.filename);  读取assets下的文件资源，通过以下方式获取输入流来进行写操作  AssetManager am = null;  am = getAssets();  InputStream is = am.open("filename");   1. Android中的五种存储方式及其应用场景   答：1)SharedPreferences  存储路径:(data/data/packagename/shares\_prefs), 轻量级存储,以键值对的形式存储在xml中,一般用来保存应用中的设置属性  2)文件存储 SD卡存储多媒体文件, 文件缓存  3) Sqlite数据库 存储路径:(data/data/packagename/databases), 一种嵌入式数据库,支持sql语言,存储大量结构性数据  4)ContentProvider 进程(应用程序)间数据共享,数据源可以是sqlite,也可以是xml,相关类: ContentResolver(内容解析器), ContentObserver(数据 观察者)  5) 网络存储 天气数据的xml,json格式等等,通过HttpUrlConnection,HttpClient,或者SOAP协议获取数据   1. 什么是ANR 如何避免它？（Android线程间的通信方式 ）   答：ANR：Application Not Responding(应用程序无响应).当出现下列情况时，Android就会显示ANR对话框了： 对输入事件（如按键、触摸屏事件）的响应超过5秒 意向接受器（intentReceiver）超过10秒钟仍未执行完毕Android应用程序完全运行在一个独立的线程中（例如main）。这就意味着，任何在主线程中运行的，需要消耗大量时间的操作都会引发ANR。  解决方案有两种:  1)AsyncTask异步任务中，doInBackground()和onPostExecute(Result)两个方法非常重要  doInBackground() 这个方法运行在后台线程中，主要负责执行那些很耗时的操作，如移动护理系统中的网络连接、解析XML等操作。该方法必须重载。  onPostExecute(Result) 这个方法也运行于UI线程，在doInBackground(Params…)方法执行后调用，该方法用于处理后台任务执行后返回的结果。  2) 子thread + handler   1. listview优化策略   答：1)对convetView进行判空，是当convertView不为空的时候直接重新使用convertView  从而减少了很多不必要的View的创建   1. 定义一个ViewHolder，将convetView的tag设置为ViewHolder,不为空时重新使用即可 2. ListView分页加载实现思路   实现OnScrollListener 接口重写onScrollStateChanged 和onScroll方法，使用onscroll方法实现”滑动“后处理检查是否还有新的记录，如果有，调用 addFooterView，添加记录到adapter, adapter调用 notifyDataSetChanged 更新数据;如果没有记录了，把自定义的mFooterView去掉。使用onScrollStateChanged可以检测是否滚到最后一行且停止滚动然后执行加载   1. ListView图片异步加载实现思路   1、对ImageView通过setTag(传入图片路径)进行绑定,为了避免错位。  2、先去内存缓存中查看是否存在图片，如果存在直接返回并展示，如果不存在先设置默认图片，并且开启子线程去下载图片。  3、当图片下载完成，通过监听器模式，使用接口回调进行图片刷新。（根据父类的findViewWithTag方法找到对应path绑定的ImageView。若没有，则不操作）  imageView.setTag(path);// setTag设置标签，进行绑定  Bitmap image = async.getImage(path, new CallBackImage() {  @Override  public void downloadImage(Bitmap bitmap) {  // TODO Auto-generated method stub  ImageView findViewWithTag = (ImageView) listView  .findViewWithTag(path);  if (findViewWithTag != null) {  findViewWithTag.setImageBitmap(bitmap);  }  }  });  if (image != null) {  imageView.setImageBitmap(image);  } else {  imageView.setImageResource(R.drawable.ic\_launcher);  }   1. Intent的原理、作用、可以传递哪些类型的参数   答：  intent是连接Activity, Service, BroadcastReceiver, ContentProvider四大组件的信使,，可以传递八种基本数据类型以及string, Bundle类型，以及实现了Serializable或者Parcelable的类型。  Intent可以划分成显式意图和隐式意图。  显式意图：调用Intent.setComponent()或Intent.setClass()方法明确指定了组件名的Intent为显式意图，显式意图明确指定了Intent应该传递给哪个组件。  隐式意图：没有明确指定组件名的Intent为隐式意图。 Android系统会根据隐式意图中设置的动作(action)、类别(category)、数据（URI和数据类型）找到最合适的组件来处理这个意图。   1. 如何实现屏幕分辨率的自适应（布局、九线图、目录）   答: 最好可以通过权重(layout\_weight)的方式来分配每个组件的大小，也可以通过具体的像素(dip)来确定大小。  尽量使用Relativelayout 。  已知应用支持平台设备的分辨率,可以提供多个layout\_320\*480 ...  drawable-hdpi,drawable-mdpi,drawable-ldpi分别代表分辨率为480\*800,360\*480,240\*360, 放置图片大小相差1.5倍  最后还需要在AndroidManifest.xml里添加下面一段，没有这一段自适应就不能实现：  <supports-screens  android:largeScreens="true"  android:normalScreens="true"  android:anyDensity = "true"/>  在</application>标签和</manifest> 标签之间添加上面那段代码。即可。  备注：三者的解析度不一样，就像你把电脑的分辨率调低，图片会变大一样，反之分辨率高，图片缩小  还可以通过.9.png实现图片的自适应   1. 简述Android中的IPC机制   IPC（Inter-Process Communication，进程间通信）,aidl是 Android Interface definition language的缩写，它是一种android内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口.编译器可以通过扩展名为aidl的文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的.  BroadcastReceiver也可以实现进程间通信  ContentProvider 提供进程间数据共享   1. Android程序入口如何判断   action节点中的android.intent.action.MAIN表明它所在的Activity是整个应用程序的入口点   1. android哪几种方式访问网络(Http、Https、socket、webService)   http协议：超文本传输协议  (最底层的是Socket,接着是URLConnection，HttpClient)  HttpURLConnection  HttpClient方式（HttpGet和HttpPost类）   1. 说说HttpClient的通信过程   1)生成请求对象（HttpGet get，HttpPost post）  2)生成客户端对象 HttpClient client  3)执行请求接收相应 HttpResponse response = client.execute(post)  HttpEntity entity = response.getEntity()  4)得到数据流  InputStream inputStream = entity.getContent();  5)最后关闭过期连接   1. 移动互联数据交互格式有哪些及其区别，JSON与JavaScript的交互更加方便为什么?   （Json（有数据类型）与xml（没有）的区别？）  移动互联数据交互格式有XML和JSON  1)JSON和XML的数据可读性基本相同  2)JSON和XML同样拥有丰富的解析手段  3)JSON相对于XML来讲，数据的体积小  4)JSON与JavaScript的交互更加方便  5)JSON对数据的描述性比XML较差   1. XML解析有哪几种？各自优缺点，官方推荐使用哪种   基本的解析方式有三种: DOM,SAX,Pull 1)dom解析解析器读入整个文档，然后构建一个驻留内存的树结构，然后代码就可以使用 DOM 接口来操作这个树结构的优点是对文档增删改查比较方便，缺点占用内存比较大。 2)sax解析基于事件驱动型,优点占用内存少，解析速度快，缺点是只适合做文档的读取，不适合做文档的增删改查。  3)pull解析同样基于事件驱动型,android 官方API提供,可随时终止,调用next() 方法提取它们（主动提取事件）   1. sax解析代码   首先SAXParserFactory来创建一个SAXParserFactory实例  SAXParserFactory factory = SAXParserFactory.newInstance();  根据SAXParserFactory实例来创建SAXParser  SAXParser产生SAXReader  XMLReader reader = factory.newSAXParser().getXMLReader();  XMLReader 加载XML，然后解析XML，在解析的过程中触发相对于接口中的事件处理程序   1. 百度地图核心类，及实现的功能   BMapManager:地图引擎管理类,负责初始化，开启地图API，终止百度地图API等工作  MKSearch：搜索服务.用于位置检索、周边检索、范围检索、公交检索、驾乘检索、步行检索  MKSearchListener搜索结果通知接口。该接口返回poi搜索,公交搜索,驾乘路线,步行路线结果  MapView:显示地图的View  MyLocationOverlay:一个负责显示用户当前位置的Overlay。  Overlay:Overlay是一个基类，它表示可以显示在地图上方的覆盖物。   1. GC内存泄露在什么情况下会出现？怎么解决 2. 查询数据库没有关闭游标 3. 构造Adapter时，没有使用缓存的 convertView 4. Bitmap对象不在使用时调用recycle()释放内存 5. 不用的对象没有及时释放对象的引用 6. android内存的优化   答：android内存泄露容易导致内存溢出，又称为OOM。  Android内存优化策略：  1）在循环内尽量不要使用局部变量 2）不用的对象即时释放，即指向NULL  3）数据库的cursor即时关闭。 4）构造adapter时使用缓存contentview 5）调用registerReceiver()后在对应的生命周期方法中调用unregisterReceiver() 6）即时关闭InputStream/OutputStream。 7）android系统给图片分配的内存只有8M, 图片尽量使用软引用, 较大图片可通过BitmapFactory缩放后再使用,并及时recycle  8）尽量避免static成员变量引用资源耗费过多的实例。   1. 加载大图片的时候如何防止内存溢出   答: android系统给图片分配的内存只有8M,当加载大量图片时往往会出现OOM。  Android加载大量图片内存溢出解决方案：  1）尽量不要使用setImageBitmap或setImageResource或BitmapFactory.decodeResource来设置一张大图，因为这些函数在完成decode后，最终都是通过java层的createBitmap来完成的，需要消耗更多内存，可以通过BitmapFactory.decodeStream方法，创建出一个bitmap，再将其设为ImageView的 source  2）使用BitmapFactory.Options对图片进行压缩  InputStream is = this.getResources().openRawResource(R.drawable.pic1);  BitmapFactory.Options options=new BitmapFactory.Options();  options.inJustDecodeBounds = false;  options.inSampleSize = 10; //width，hight设为原来的十分一  Bitmap btp =BitmapFactory.decodeStream(is,null,options);  3）运用Java软引用，进行图片缓存，将需要经常加载的图片放进缓存里，避免反复加载  及时销毁不再使用的Bitmap对象  if(!bmp.isRecycle() ){  bmp.recycle() //回收图片所占的内存  system.gc() //提醒系统及时回收  }  4)使用Lru算法来管理Bitmap，从而实现Bitmap的内存回收，避免出现OOM   1. Android缓存机制（sharePrefrence、Sd卡、数据库SQLite）   答: 客户端缓存机制是android应用开发中非常重要的一项工作,使用缓存机制不仅仅可以为用户节省3G流量,同时在用户体验方面也是非常好的选择，比如有些新闻客户端支持离线模式，也是通过缓存机制实现的.缓存机制分为两部分,一部分是文字缓存,另一部分是多媒体文件缓存.  文字缓存有两种实现：  1）可以将与服务器交互得到的json数据或者xml数据存入sd卡中，并在数据库添加该数据的记录.添加数据库记录时,提供两个关键字段,一个是请求的URL,另一个则是本地保存后的文件地址，每次加载数据之前都会根据URL在数据库中检索  2）将JSON数据解析后装入List<Map>对象中,然后遍历List,将数据统统写入相应的数据库表结构中,以后每次向服务器发起请求之前可以先在数据库中检索,如果有直接返回.  多媒体文件缓存：主要指图片缓存  图片的缓存可以根据当前日期,时间为名字缓存到SD卡中的指定图片缓存目录,同时数据库中做相应记录,记录办法可以采用两个关键字段控制,一个字段是该图片的URL地址,另一个字段是该图片的本机地址.取图片时根据URL在数据中检索,如果没有则连接服务器下载,下载之后再服务器中作出相应记录  缓存文件删除策略:  1) 每一个模块在每次客户端自动或者用户手动更新的时候删除相应模块的缓存文件,并重新下载新的缓存文件.  2)在设置界面中提供删除缓存的功能,点击后删除本机所有缓存.   1. 如何实现消息推送   有哪些方式，各自优缺点，最常使用哪种  答：实现消息推送的方式有五种，分别是轮询，SMS,C2DM,MQTT,XMPP最常使用的是XMPP, 我们做项目时采用的是XMPP协议  1)XMPP协议，它是一种基于XML的传递协议，具有很强的灵活性和可扩展性。它的特点是将复杂性从客户端转移到了服务器端。GTalk、QQ、IM等都用这个协议。  2)轮询:客户端定时去服务端取或者保持一个长Socket，从本质讲这个不叫推送，而是去服务端拽数据。但是实现简单，主要缺点：耗电,浪费用户流量等 3)Google的C2DM，具体不细说，缺点，服务器在国外，不是很稳定。  4)通过短信方式, 但是很难找到免费短信平台  5.)MQTT协议, IBM提供的一种推送服务,不太灵活  6)第三方消息推送，极光、蝴蝶、百度云   1. MVC在Android中的应用   答: Android中界面部分也采用了当前比较流行的MVC框架，在Android中：  1) 视图层（View）：一般采用XML文件进行界面的描述，使用的时候可以非常方便的引入。也可以使用JavaScript+HTML等的方式作为View层，通过WebView组件加载,同时可以实现Java和JavaScript之间的通信。  2) 控制层（Controller）：这句话也就暗含了不要在Acitivity中写代码，要通过Activity交割Model业务逻辑层处理，这样做的另外一个原因是Android中的Acitivity的响应时间是5s，如果耗时的操作放在这里，Android的控制层的重任通常落在了众多的Acitvity的肩上，程序就很容易被回收掉。  3) 模型层（Model）：对数据库的操作、对网络等的操作都应该在Model里面处理，当然对业务计算等操作也是必须放在的该层的。  在Android SDK中的数据绑定，也都是采用了与MVC框架类似的方法来显示数据。在控制层上将数据按照视图模型的要求（也就是Android SDK中的Adapter）封装就可以直接在视图模型上显示了，从而实现了数据绑定。比如显示Cursor中所有数据的ListActivity，其视图层就是一个ListView，将数据封装为ListAdapter，并传递给ListView，数据就在ListView中显示。   1. Android自定义组件实现思路(参考我们的画板)   答: Android自定义组件有三种实现思路:   1. 继承某个现有组件，在其基础上添加额外功能,如继承Gallery实现CoverFlow效果 2. 继承某个Layout，实现复合组件自定义，如TextView和EditText组合实现登录注册组件 3. 继承View，实现onDraw()方法，实现自己绘制组件，如翻页效果组件 4. 版本更新的实现思路   答：在服务器相应URL上有版本文件, 客户端同时存储该应用当前版本号 (SharedPreferences/Sqlite), 每次打开应用,去检测服务器版本号与本地版本号是否一致,如果不一 致,则自定义对话框提示是否下载更新   1. 播放视频有哪些实现方式   答：  1)使用系统自带的播放器来播放，指定Action为ACTION\_VIEW,Data为Uri，Type为其MIME类型。  //调用系统自带的播放器  Intent intent = new Intent(Intent.ACTION\_VIEW);  intent.setDataAndType(uri, "video/mp4");  startActivity(intent);  2)使用VideoView组件来播放, 可以结合MediaController来实现播控, 只是不能随意更改视频的大小及位置。  3)使用MediaPlayer和SurfaceView来实现，这种方式很灵活,可以自定义视频播放的大小和位置。   1. NDK开发流程？（JNI运行原理）   答: NDK应用的开发流程(在应用中定义本地接口(native), 编译成.h头文件,交由C程序员实现,将.c实现通过NDK编译成.so动态链接库,导入项目中libs/armeabi,代码中调用该本地接口)  应用场景: 音频,视频解码,拍摄车牌号,识别车牌号   1. 如何实现一键退出   答：定义一个类继承Application，定义一个集合存放所有的activity，  定义一个添加的方法，再写一个退出的方法，使用for循环全部调用finish方法，然  后在每个Activity的onCreate方法中调用自定义类里的添加方法，然后在需要使用一  键退出的地方调用类中的退出方法即可。   1. AndroidManifest.xml清单文件<Activity>标签中属性的含义   AndroidManifest.xml清单文件<Activity>标签中属性android:excludeFromRecents="true" android:screenOrientation="portrait" android:configChanges="orientation|locale"的含义  答: android:excludeFromRecents表示是否可被显示在最近打开的activity列表里,true表示否,false表示是  android:screenOrientation表示activity显示的模式, 一般用来设置activity横屏显示(horizontal)或竖屏显示(portrait)  android:configChanges=[oneormoreof:"mcc""mnc""locale""touchscreen""keyboard""keyboardHidden""navigation""orientation""fontScale"] 是当所指定属性(Configuration Changes)发生改变时，通知程序调用 onConfigurationChanged()函数,比如orientation屏幕方向发生改变,locale语言环境发生改变时   1. 如何将一个Activity设置成窗口的样式   答：在清单文件AndroidManifest.xml中相应的<activity>标签内设置属性android:theme=”@android:style/Theme.Dialog”   1. 谈谈UI中， Padding和Margin有什么区别,gravity与layout\_gravity的区别   答：Padding 用来指定组件内的内容距离组件边界的距离;  Margin用来指定控件与控件之间的距离  Gravity用来指定组件内的内容相对于组件本身的位置  Layout\_gravity用来指定组件相对于其父组件的位置   1. 实现手风琴效果   哪个组件可以实现手风琴效果，用来实现设置界面的类，实现抽屉效果, 悬浮窗口？  答：实现手风琴效果（ExpandableListView）  设置界面的类（preferenceActivity）保存到sharedpreference中  抽屉效果（slidingDrawer）组件  悬浮窗口: PopWindow,可以实现类似Dialog和菜单的效果   1. Android SDK 3.0(HoneyComb)及4.0(Ice Cream)新特性   答：新版SDK发布的同时也发布了一个扩展包android-support-v4，把部分特性单独的抽出来，使低版本的SDK也可以使用这些特性，主要支持以下特性：  Fragment: 3.0引入,碎片管理,可以局部刷新UI,它设计的功能和Activity一样强大，包括生命周期、导航等，Fragment的每次导航都可以记录下来用于返回。  ViewPager: 提供了多界面切换的新效果  GridLayout: 4.0引入, 网格布局, android第六大布局  Loader: 装载器从android3.0开始引进。它使得在activity或fragment中异步加载数据变得简单   1. android客户端如何实现自动登录   答: 通过SharedPreferences存储用户名,密码,当存储不为空时实现自动登录功能   1. 请介绍下Android中常用的五种布局。   常用五种布局方式，分别是：FrameLayout（框架布局），LinearLayout （线性布局），AbsoluteLayout（绝对布局），RelativeLayout（相对布局），TableLayout（表格布局）。  1)FrameLayout：所有东西依次都放在左上角，会重叠，这个布局比较简单，也只能放一点比较简单的东西。2)LinearLayout：线性布局，每一个LinearLayout里面又可分为垂直布局（android:orientation="vertical"）和水平布局（android:orientation="horizontal" ）。当垂直布局时，每一行就只有一个元素，多个元素依次垂直往下；水平布局时，只有一行，每一个元素依次向右排列。  3)AbsoluteLayout：绝对布局用X,Y坐标来指定元素的位置，这种布局方式也比较简单，但是在屏幕旋转时，往往会出问题，而且多个元素的时候，计算比较麻烦。  4)RelativeLayout：相对布局可以理解为某一个元素为参照物，来定位的布局方式。主要属性有：相对于某一个元素android:layout\_below、 android:layout\_toLeftOf相对于父元素的地方android:layout\_alignParentLeft、android:layout\_alignParentRigh;  5)TableLayout：表格布局，每一个TableLayout里面有表格行TableRow，TableRow里面可以具体定义每一个元素。每一个布局都有自己适合的方式，这五个布局元素可以相互嵌套应用，做出美观的界面。   1. android中的动画有哪几类，它们的特点和区别是什么   答：两种，一种是Tween动画、还有一种是Frame动画。Tween动画，这种实现方式可以使视图组件移动、放大、缩小以及产生透明度的变化;另一种Frame动画，传统的动画方法，通过顺序的播放排列好的图片来实现，类似电影。   1. activity的启动模式有哪些？是什么含义   答：在android里，有4种activity的启动模式，分别为：  “standard” (默认)  “singleTop”  “singleTask”  “singleInstance”  它们主要有如下不同：  1)如何决定所属task  “standard”和”singleTop”的activity的目标task，和收到的Intent的发送者在同一个task内，除非intent包括参数FLAG\_ACTIVITY\_NEW\_TASK。  如果提供了FLAG\_ACTIVITY\_NEW\_TASK参数，会启动到别的task里。  “singleTask”和”singleInstance”总是把activity作为一个task的根元素，他们不会被启动到一个其他task里。  2)是否允许多个实例  “standard”和”singleTop”可以被实例化多次，并且存在于不同的task中，且一个task可以包括一个activity的多个实例；  “singleTask”和”singleInstance”则限制只生成一个实例，并且是task的根元素。 singleTop要求如果创建intent的时候栈顶已经有要创建 的Activity的实例，则将intent发送给该实例，而不发送给新的实例。  3)是否允许其它activity存在于本task内  “singleInstance”独占一个task，其它activity不能存在那个task里；如果它启动了一个新的activity，不管新的activity的launch mode 如何，新的activity都将会到别的task里运行（如同加了FLAG\_ACTIVITY\_NEW\_TASK参数）。  而另外三种模式，则可以和其它activity共存。  4)是否每次都生成新实例  “standard”对于没一个启动Intent都会生成一个activity的新实例；  “singleTop”的activity如果在task的栈顶的话，则不生成新的该activity的实例，直接使用栈顶的实例，否则，生成该activity的实例。  比如现在task栈元素为A-B-C-D（D在栈顶），这时候给D发一个启动intent，如果D是 “standard”的，则生成D的一个新实例，栈变为A－B－C－D－D。  如果D是singleTop的话，则不会生产D的新实例，栈状态仍为A-B-C-D  如果这时候给B发Intent的话，不管B的launchmode是”standard” 还是 “singleTop” ，都会生成B的新实例，栈状态变为A-B-C-D-B。  “singleInstance”是其所在栈的唯一activity，它会每次都被重用。  “singleTask”如果在栈顶，则接受intent，否则，该intent会被丢弃，但是该task仍会回到前台。  当已经存在的activity实例处理新的intent时候，会调用onNewIntent()方法 如果收到intent生成一个activity实例，那么用户可以通过back键回到上一个状态；如果是已经存在的一个activity来处理这个intent的话，用户不能通过按back键返回到这之前的状态。   1. 跟activity和Task 有关的 Intent启动方式有哪些？其含义是   核心的Intent Flag有：  FLAG\_ACTIVITY\_NEW\_TASK  FLAG\_ACTIVITY\_CLEAR\_TOP  FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED  FLAG\_ACTIVITY\_SINGLE\_TOP  FLAG\_ACTIVITY\_NEW\_TASK  如果设置，这个Activity会成为历史stack中一个新Task的开始。一个Task（从启动它的Activity到下一个Task中的 Activity）定义了用户可以迁移的Activity原子组。Task可以移动到前台和后台；在某个特定Task中的所有Activity总是保持相同的次序。  这个标志一般用于呈现“启动”类型的行为：它们提供用户一系列可以单独完成的事情，与启动它们的Activity完全无关。  使用这个标志，如果正在启动的Activity的Task已经在运行的话，那么，新的Activity将不会启动；代替的，当前Task会简单的移入前台。参考FLAG\_ACTIVITY\_MULTIPLE\_TASK标志，可以禁用这一行为。  这个标志不能用于调用方对已经启动的Activity请求结果。  FLAG\_ACTIVITY\_CLEAR\_TOP   如果设置，并且这个Activity已经在当前的Task中运行，因此，不再是重新启动一个这个Activity的实例，而是在这个Activity上方的所有Activity都将关闭，然后这个Intent会作为一个新的Intent投递到老的Activity（现在位于顶端）中。   例如，假设一个Task中包含这些Activity：A，B，C，D。如果D调用了startActivity()，并且包含一个指向Activity B的Intent，那么，C和D都将结束，然后B接收到这个Intent，因此，目前stack的状况是：A，B。   上例中正在运行的Activity B既可以在onNewIntent()中接收到这个新的Intent，也可以把自己关闭然后重新启动来接收这个Intent。如果它的启动模式声明为 “multiple”(默认值)，并且你没有在这个Intent中设置FLAG\_ACTIVITY\_SINGLE\_TOP标志，那么它将关闭然后重新创建；对于其它的启动模式，或者在这个Intent中设置FLAG\_ACTIVITY\_SINGLE\_TOP标志，都将把这个Intent投递到当前这个实例的onNewIntent()中。   这个启动模式还可以与FLAG\_ACTIVITY\_NEW\_TASK结合起来使用：用于启动一个Task中的根Activity，它会把那个Task中任何运行的实例带入前台，然后清除它直到根Activity。这非常有用，例如，当从Notification Manager处启动一个Activity。  FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED  如果设置这个标志，这个activity不管是从一个新的栈启动还是从已有栈推到栈顶，它都将以the front door of the task的方式启动。这就讲导致任何与应用相关的栈都讲重置到正常状态（不管是正在讲activity移入还是移除），如果需要，或者直接重置该栈为初始状态。  FLAG\_ACTIVITY\_SINGLE\_TOP   如果设置，当这个Activity位于历史stack的顶端运行时，不再启动一个新的  FLAG\_ACTIVITY\_BROUGHT\_TO\_FRONT   这个标志一般不是由程序代码设置的，如在launchMode中设置singleTask模式时系统帮你设定。  FLAG\_ACTIVITY\_CLEAR\_WHEN\_TASK\_RESET   如果设置，这将在Task的Activity stack中设置一个还原点，当Task恢复时，需要清理Activity。也就是说，下一次Task带着 FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED标记进入前台时（典型的操作是用户在主画面重启它），这个Activity和它之上的都将关闭，以至于用户不能再返回到它们，但是可以回到之前的Activity。   这在你的程序有分割点的时候很有用。例如，一个e-mail应用程序可能有一个操作是查看一个附件，需要启动图片浏览Activity来显示。这个 Activity应该作为e-mail应用程序Task的一部分，因为这是用户在这个Task中触发的操作。然而，当用户离开这个Task，然后从主画面选择e-mail app，我们可能希望回到查看的会话中，但不是查看图片附件，因为这让人困惑。通过在启动图片浏览时设定这个标志，浏览及其它启动的Activity在下次用户返回到mail程序时都将全部清除。  FLAG\_ACTIVITY\_EXCLUDE\_FROM\_RECENTS   如果设置，新的Activity不会在最近启动的Activity的列表中保存。  FLAG\_ACTIVITY\_FORWARD\_RESULT   如果设置，并且这个Intent用于从一个存在的Activity启动一个新的Activity，那么，这个作为答复目标的Activity将会传到这个新的Activity中。这种方式下，新的Activity可以调用setResult(int)，并且这个结果值将发送给那个作为答复目标的 Activity。  FLAG\_ACTIVITY\_LAUNCHED\_FROM\_HISTORY   这个标志一般不由应用程序代码设置，如果这个Activity是从历史记录里启动的（常按HOME键），那么，系统会帮你设定。  FLAG\_ACTIVITY\_MULTIPLE\_TASK   不要使用这个标志，除非你自己实现了应用程序启动器。与FLAG\_ACTIVITY\_NEW\_TASK结合起来使用，可以禁用把已存的Task送入前台的行为。当设置时，新的Task总是会启动来处理Intent，而不管这是是否已经有一个Task可以处理相同的事情。   由于默认的系统不包含图形Task管理功能，因此，你不应该使用这个标志，除非你提供给用户一种方式可以返回到已经启动的Task。   如果FLAG\_ACTIVITY\_NEW\_TASK标志没有设置，这个标志被忽略。  FLAG\_ACTIVITY\_NO\_ANIMATION   如果在Intent中设置，并传递给Context.startActivity()的话，这个标志将阻止系统进入下一个Activity时应用 Acitivity迁移动画。这并不意味着动画将永不运行——如果另一个Activity在启动显示之前，没有指定这个标志，那么，动画将被应用。这个标志可以很好的用于执行一连串的操作，而动画被看作是更高一级的事件的驱动。  FLAG\_ACTIVITY\_NO\_HISTORY   如果设置，新的Activity将不再历史stack中保留。用户一离开它，这个Activity就关闭了。这也可以通过设置noHistory特性。  FLAG\_ACTIVITY\_NO\_USER\_ACTION   如果设置，作为新启动的Activity进入前台时，这个标志将在Activity暂停之前阻止从最前方的Activity回调的onUserLeaveHint()。   典型的，一个Activity可以依赖这个回调指明显式的用户动作引起的Activity移出后台。这个回调在Activity的生命周期中标记一个合适的点，并关闭一些Notification。   如果一个Activity通过非用户驱动的事件，如来电或闹钟，启动的，这个标志也应该传递给Context.startActivity，保证暂停的Activity不认为用户已经知晓其Notification。  FLAG\_ACTIVITY\_PREVIOUS\_IS\_TOP   If set and this intent is being used to launch a new activity from an existing one, the current activity will not be counted as the top activity for deciding whether the new intent should be delivered to the top instead of starting a new one. The previous activity will be used as the top, with the assumption being that the current activity will finish itself immediately.  FLAG\_ACTIVITY\_REORDER\_TO\_FRONT   如果在Intent中设置，并传递给Context.startActivity()，这个标志将引发已经运行的Activity移动到历史stack的顶端。   例如，假设一个Task由四个Activity组成：A,B,C,D。如果D调用startActivity()来启动Activity B，那么，B会移动到历史stack的顶端，现在的次序变成A,C,D,B。如果FLAG\_ACTIVITY\_CLEAR\_TOP标志也设置的话，那么这个标志将被忽略。   1. 如何启用Service，如何停用Service。   服务的开发比较简单，如下：  第一步：继承Service类  public class SMSService extends Service {}  第二步：在AndroidManifest.xml文件中的<application>节点里对服务进行配置:<service android:name=".SMSService" />  服务不能自己运行，需要通过调用Context.startService()或Context.bindService()方法启动服务。这两个方法都可以启动Service，但是它们的使用场合有所不同。使用startService()方法启用服务，调用者与服务之间没有关连，即使调用者退出了，服务仍然运行。使用bindService()方法启用服务，调用者与服务绑定在了一起，调用者一旦退出，服务也就终止，大有“不求同时生，必须同时死”的特点。  如果打算采用Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，接着调用onStart()方法。如果调用startService()方法前服务已经被创建，多次调用startService()方法并不会导致多次创建服务，但会导致多次调用onStart()方法。采用startService()方法启动的服务，只能调用Context.stopService()方法结束服务，服务结束时会调用onDestroy()方法。  如果打算采用Context.bindService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，接着调用onBind()方法。这个时候调用者和服务绑定在一起，调用者退出了，系统就会先调用服务的onUnbind()方法，接着调用onDestroy()方法。如果调用bindService()方法前服务已经被绑定，多次调用bindService()方法并不会导致多次创建服务及绑定(也就是说onCreate()和onBind()方法并不会被多次调用)。如果调用者希望与正在绑定的服务解除绑定，可以调用unbindService()方法，调用该方法也会导致系统调用服务的onUnbind()-->onDestroy()方法。  服务常用生命周期回调方法如下：  onCreate() 该方法在服务被创建时调用，该方法只会被调用一次，无论调用多少次startService()或bindService()方法，服务也只被创建一次。  onDestroy()该方法在服务被终止时调用。  与采用Context.startService()方法启动服务有关的生命周期方法  onStart() 只有采用Context.startService()方法启动服务时才会回调该方法。该方法在服务开始运行时被调用。多次调用startService()方法尽管不会多次创建服务，但onStart() 方法会被多次调用。  与采用Context.bindService()方法启动服务有关的生命周期方法  onBind()只有采用Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务绑定时被调用，当调用者与服务已经绑定，多次调用Context.bindService()方法并不会导致该方法被多次调用。  onUnbind()只有采用Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务解除绑定时被调用     1. 说明handler机制的原理   一个Handler允许你发送和处理Message和Runable对象，每个线程都有自己的Looper，每个Looper中封装着MessageQueue。Looper负责不断的从自己的消息队列里取出队头的任务或消息执行。每个handler也和线程关联，Handler负责把Message和Runable  对象传递给MessageQueue（用到post，sendMessage等方法），而且在这些对象离开MessageQueue时，Handler负责执行他们（用到handleMessage方法）。  其中Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象   1. 什么情况会导致Force Close ？如何避免？能否捕获导致其的异常   答：程序出现异常，比如nullpointer。  避免：编写程序时逻辑连贯，思维缜密。能捕获异常，在logcat中能看到异常信息   1. 请介绍下ContentProvider是如何实现数据共享的。   一个程序可以通过实现一个Content provider的抽象接口将自己的数据完全暴露出去，而且Content providers是以类似数据库中表的方式将数据暴露。Content providers存储和检索数据，通过它可以让所有的应用程序访问到，这也是应用程序之间唯一共享数据的方法。  要想使应用程序的数据公开化，可通过2种方法：创建一个属于你自己的Content provider或者将你的数据添加到一个已经存在的Content provider中，前提是有相同数据类型并且有写入Content provider的权限。  如何通过一套标准及统一的接口获取其他应用程序暴露的数据  Android提供了ContentResolver，外界的程序可以通过ContentResolver接口访问ContentProvider提供的数据。   1. Service和Thread的区别   答：servie是系统的组件，它由系统进程托管（servicemanager）；它们之间的通信类似于client和server，是一种轻量级的ipc通信，这种通信的载体是binder，它是在linux层交换信息的一种ipc。而thread是由本应用程序托管。1). Thread：Thread 是程序执行的最小单元，它是分配CPU的基本单位。可以用 Thread 来执行一些异步的操作。  2). Service：Service 是android的一种机制，当它运行的时候如果是Local Service，那么对应的 Service 是运行在主进程的 main 线程上的。如：onCreate，onStart 这些函数在被系统调用的时候都是在主进程的 main 线程上运行的。如果是Remote Service，那么对应的 Service 则是运行在独立进程的 main 线程上。  既然这样，那么我们为什么要用 Service 呢？其实这跟 android 的系统机制有关，我们先拿 Thread 来说。Thread 的运行是独立于 Activity 的，也就是说当一个 Activity 被 finish 之后，如果你没有主动停止 Thread 或者 Thread 里的 run 方法没有执行完毕的话，Thread 也会一直执行。因此这里会出现一个问题：当 Activity 被 finish 之后，你不再持有该 Thread 的引用。另一方面，你没有办法在不同的 Activity 中对同一 Thread 进行控制。  举个例子：如果你的 Thread 需要不停地隔一段时间就要连接服务器做某种同步的话，该 Thread 需要在 Activity 没有start的时候也在运行。这个时候当你 start 一个 Activity 就没有办法在该 Activity 里面控制之前创建的 Thread。因此你便需要创建并启动一个 Service ，在 Service 里面创建、运行并控制该 Thread，这样便解决了该问题（因为任何 Activity 都可以控制同一 Service，而系统也只会创建一个对应 Service 的实例）。  因此你可以把 Service 想象成一种消息服务，而你可以在任何有 Context 的地方调用 Context.startService、Context.stopService、Context.bindService，Context.unbindService，来控制它，你也可以在 Service 里注册 BroadcastReceiver，在其他地方通过发送 broadcast 来控制它，当然这些都是 Thread 做不到的。   1. Android异常   Android本身的api并未声明会抛出异常，则其在运行时有无可能抛出runtime异常，你遇到过吗？诺有的话会导致什么问题？如何解决？  答：会，比如nullpointerException。我遇到过，比如textview.setText()时，textview没有初始化。会导致程序无法正常运行出现forceclose。打开控制台查看logcat信息找出异常信息并修改程序。   1. IntentService,IntentPending有何优点   答：Acitivity的进程，当处理Intent的时候，会产生一个对应的Service； Android的进程处理器现在会尽可能的不kill掉你；非常容易使用   1. Activity   如果后台的Activity由于某原因被系统回收了，如何在被系统回收之前保存当前状态？  答：重写onSaveInstanceState()方法，在此方法中保存需要保存的数据，该方法将会在activity被回收之前调用。通过重写onRestoreInstanceState()方法可以从中提取保存好的数据   1. 如何退出Activity？如何安全退出已调用多个Activity的Application   答：对于单一Activity的应用来说，退出很简单，直接finish()即可。当然，也可以用killProcess()和System.exit()这样的方法。  对于多个activity:  1)记录打开的Activity：每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可  2)发送特定广播：在需要结束应用时，发送一个特定的广播，每个Activity收到广播后，关闭即可。  3)递归退出：在打开新的Activity时使用startActivityForResult，然后自己加标志，在onActivityResult中处理，递归关闭。为了编程方便，最好定义一个Activity基类，处理这些共通问题。  在2.1之前，可以使用ActivityManager的restartPackage方法。  它可以直接结束整个应用。在使用时需要权限android.permission.RESTART\_PACKAGES。  注意不要被它的名字迷惑。  可是，在2.2，这个方法失效了。在2.2添加了一个新的方法，killBackground Processes()，需要权限 android.permission.KILL\_BACKGROUND\_PROCESSES。可惜的是，它和2.2的restartPackage一样，根本起不到应有的效果。  另外还有一个方法，就是系统自带的应用程序管理里，强制结束程序的方法，forceStopPackage()。它需要权限android.permission.FORCE\_STOP\_PACKAGES。并且需要添加android:sharedUserId="android.uid.system"属性。同样可惜的是，该方法是非公开的，他只能运行在系统进程，第三方程序无法调用。  因为需要在Android.mk中添加LOCAL\_CERTIFICATE := platform。  而Android.mk是用于在Android源码下编译程序用的。  从以上可以看出，在2.2，没有办法直接结束一个应用，而只能用自己的办法间接办到。  现提供几个方法，供参考：  1)抛异常强制退出：  该方法通过抛异常，使程序Force Close。  验证可以，但是，需要解决的问题是，如何使程序结束掉，而不弹出Force Close的窗口。  2)记录打开的Activity：  每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可。  3)发送特定广播：  在需要结束应用时，发送一个特定的广播，每个Activity收到广播后，关闭即可。  4)递归退出  在打开新的Activity时使用startActivityForResult，然后自己加标志，在onActivityResult中处理，递归关闭。  除了第一个，都是想办法把每一个Activity都结束掉，间接达到目的。但是这样做同样不完美。你会发现，如果自己的应用程序对每一个Activity都设置了nosensor，在两个Activity结束的间隙，sensor可能有效了。但至少，我们的目的达到了，而且没有影响用户使用。为了编程方便，最好定义一个Activity基类，处理这些共通问题。   1. AIDL的全称是什么？如何工作？能处理哪些类型的数据   答：全称是：Android Interface Define Language  在Android中, 每个应用程序都可以有自己的进程. 在写UI应用的时候, 经常要用到Service. 在不同的进程中, 怎样传递对象呢?显然, Java中不允许跨进程内存共享. 因此传递对象, 只能把对象拆分成操作系统能理解的简单形式, 以达到跨界对象访问的目的. 在J2EE中,采用RMI的方式, 可以通过序列化传递对象. 在Android中, 则采用AIDL的方式. 理论上AIDL可以传递Bundle,实际上做起来却比较麻烦。  AIDL(AndRoid接口描述语言)是一种借口描述语言; 编译器可以通过aidl文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的. 如果需要在一个Activity中, 访问另一个Service中的某个对象, 需要先将对象转化成AIDL可识别的参数(可能是多个参数), 然后使用AIDL来传递这些参数, 在消息的接收端, 使用这些参数组装成自己需要的对象.  AIDL的IPC的机制和COM或CORBA类似, 是基于接口的，但它是轻量级的。它使用代理类在客户端和实现层间传递值. 如果要使用AIDL, 需要完成2件事情: 1. 引入AIDL的相关类.; 2. 调用aidl产生的class.  AIDL的创建方法:  AIDL语法很简单,可以用来声明一个带一个或多个方法的接口，也可以传递参数和返回值。 由于远程调用的需要, 这些参数和返回值并不是任何类型.下面是些AIDL支持的数据类型:  1)不需要import声明的简单Java编程语言类型(int,boolean等)  2) String, CharSequence不需要特殊声明  3) List, Map和Parcelables类型, 这些类型内所包含的数据成员也只能是简单数据类型, String等其他比支持的类型.  (另外: 我没尝试Parcelables, 在Eclipse+ADT下编译不过, 或许以后会有所支持)   1. 请解释下Android程序运行时权限与文件系统权限的区别   答：运行时权限Dalvik( android授权)  文件系统 linux 内核授权   1. 系统上安装了多种浏览器，能否指定某浏览器访问指定页面？请说明原由   通过直接发送Uri把参数带过去，或者通过manifest里的intentfilter里的data属性  可以通过指定浏览器，但是必须通过Intent将浏览器相应的Action传递过去   1. android系统的优势和不足   答：Android平台手机 5大优势：  ⑴开放性  在优势方面，Android平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到Android联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。开放性对于Android的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。  ⑵挣脱运营商的束缚  在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年iPhone 上市 ，用户可以更加方便地连接网络，运营商的制约减少。随着EDGE、HSDPA这些2G至3G移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，当你可以通过手机IM软件方便地进行即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样？互联网巨头Google推动的Android终端天生就有网络特色，将让用户离互联网更近。  ⑶丰富的硬件选择  这一点还是与Android平台的开放性相关，由于Android的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。功能上的差异和特色，却不会影响到数据同步、甚至软件的兼容，好比你从诺基亚 Symbian风格手机 一下改用苹果 iPhone ，同时还可将Symbian中优秀的软件带到iPhone上使用、联系人等资料更是可以方便地转移，是不是非常方便呢？  ⑷不受任何限制的开发商  Android平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如可控制正是留给Android难题之一。  ⑸无缝结合的Google应用  如今叱诧互联网的Google已经走过10年度历史，从搜索巨人到全面的互联网渗透，Google服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而Android平台手机将无缝结合这些优秀的Google服务。  再说Android的5大不足：  ⑴安全和隐私  由于手机 与互联网的紧密联系，个人隐私很难得到保守。除了上网过程中经意或不经意留下的个人足迹，Google这个巨人也时时站在你的身后，洞穿一切，因此，互联网的深入将会带来新一轮的隐私危机。  ⑵首先开卖Android手机的不是最大运营商  众所周知，T-Mobile在23日，于美国纽约发布 了Android首款手机G1。但是在北美市场，最大的两家运营商乃AT&T和Verizon，而目前所知取得Android手机销售权的仅有 T-Mobile和Sprint，其中T-Mobile的3G网络相对于其他三家也要逊色不少，因此，用户可以买账购买G1，能否体验到最佳的3G网络服务则要另当别论了！  ⑶运营商仍然能够影响到Android手机  在国内市场，不少用户对购得移动定制机不满，感觉所购的手机被人涂画了广告一般。这样的情况在国外市场同样出现。Android手机的另一发售运营商Sprint就将在其机型中内置其手机商店程序。  ⑷同类机型用户减少  在不少手机论坛都会有针对某一型号的子论坛，对一款手机的使用心得交流，并分享软件资源。而对于Android平台手机，由于厂商丰富，产品类型多样，这样使用同一款机型的用户越来越少，缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，就很少有专门针对某个型号山寨机的讨论和群组，除了哪些功能异常抢眼、颇受追捧的机型以外。  ⑸过分依赖开发商缺少标准配置  在使用PC端的Windows Xp系统的时候，都会内置微软Windows Media Player这样一个浏览器程序，用户可以选择更多样的播放器，如Realplay或暴风影音等。但入手开始使用默认的程序同样可以应付多样的需要。在 Android平台中，由于其开放性，软件更多依赖第三方厂商，比如Android系统的SDK中就没有内置音乐 播放器，全部依赖第三方开发，缺少了产品的统一性。   1. Android dvm的进程和Linux的进程, 应用程序的进程是否为同一个概念   答：DVM指dalivk的虚拟机。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。而每一个DVM都是在Linux 中的一个进程，所以说可以认为是同一个概念。   1. 嵌入式操作系统内存管理有哪几种， 各有何特性   页式，段式，段页，用到了MMU,虚拟空间等技术   1. 什么是嵌入式实时操作系统, Android 操作系统属于实时操作系统吗   嵌入式实时操作系统是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统作出快速响应，并控制所有实时任务协调一致运行的嵌入式操作系统。主要用于工业控制、 军事设备、 航空航天等领域对系统的响应时间有苛刻的要求，这就需要使用实时系统。又可分为软实时和硬实时两种，而android是基于linux内核的，因此属于软实时。   1. 一条最长的短信息约占多少byte   中文70(包括标点)，英文160，160个字节。     1. 用算法调整图片的透明度   有一个一维整型数组int[]data保存的是一张宽为width，高为height的图片像素值信息。请写一个算法，将该图片所有的白色不透明(0xffffffff)像素点的透明度调整为50%。  待定...(这样的问题没有接触过，但是我觉得应该可以解决，因为现在已经得到像素值了，我现在唯一不了解仅仅是如何将单个的像素值透明度进行计算，只需要查找一下相应的算法和原理就OK)   1. 如何将SQLite数据库(dictionary.db文件)与apk文件一起发布   解答：可以将dictionary.db文件复制到Eclipse Android工程中的res aw目录中。所有在res aw目录中的文件不会被压缩，这样可以直接提取该目录中的文件。可以将dictionary.db文件复制到res aw目录中   1. sqlite的用处及用法, 哪里用过,轻量级嵌入型数据库，经常用于嵌入型操作系统，如android中，可以用户对对象性数据的描述并存储 2. sqlite，修改，添加字段，版本更新，排序，默认排序 3. 如何建一个“aaa.db”数据库，如何建一个user表；写出增删改查语句 4. SQLite线程安全吗？sqlite3\_threadsafe()来确实是否线程安全版本   答案：SQLite是线程安全的。  由于很多用户会忽略我们在上一段中给出的建议，我们做出了这种让步。但是，为了达到线程安全，SQLite在编译时必须将SQLITE\_THREADSAFE预处理宏置为1。在Windows和Linux上，已编译的好的二进制发行版中都是这样设置的。如果不确定你所使用的库是否是线程安全的，可以调用sqlite3\_threadsafe()接口找出。   1. 怎么提高sqlite的效率   事务。一致性、原子性、隔离性、持续性   1. 如何将打开res raw目录中的数据库文件   解答：在Android中不能直接打开res aw目录中的数据库文件，而需要在程序第一次启动时将该文件复制到手机内存或SD卡的某个目录中，然后再打开该数据库文件。  复制的基本方法是使用getResources().openRawResource方法获得res aw目录中资源的 InputStream对象，然后将该InputStream对象中的数据写入其他的目录中相应文件中。在Android SDK中可以使用SQLiteDatabase.openOrCreateDatabase方法来打开任意目录中的SQLite数据库文件。   1. DDMS和TraceView的区别   DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器 。   1. java中如何引用本地语言   可以用JNI（java native interface java 本地接口）接口 。   1. EditText单行显示    1. 有属性 android:singleLine   多个输入框,点击Enter,焦点切换到下一个: EditText有属性android:imeOptions  输入法输入文本都有下划线   * 1. 自定义组件,实现onDraw方法  1. oom避免以及产生方式，遇到过几种OOM   图片和数据库，不要过多加载图片，记得回收  五大存储哪个容易报OOM，如何处理  oom情况遇到过么，怎么处理的  图片压缩后会出现什么问题？图片混乱   1. throw与throws区别   throws是用来声明一个方法可能抛出的所有异常信息 throw则是指抛出的一个具体的异常类型。 通常在一个方法（类）的声明处通过throws声明方法（类）可能抛出的异常信息，而在方法（类）内部通过throw声明一个具体的异常信息。 throws通常不用显示的捕获异常，可由系统自动将所有捕获的异常信息抛给上级方法； throw则需要用户自己捕获相关的异常，而后在对其进行相关包装，最后在将包装后的异常信息抛   1. LOST.DIR   LOST.DIR是专门收集Android系统运行时意外丢失的文件而设置的文件夹，其收集的内容包括系统因为意外而没能保存的各类内存、交换、暂存等数据，软件的缓存数据、以及其他各类文件。只要系统发生意外（如软件突然无法正常运行），便会将发生意外的文件移到此目录下，以数字序列名称代替该文件的扩展名。注意：由于LOST.DIR文件夹的特殊作用，里面的文件都是“意外而生”，因此可以删除里面的文件，但请不要删除LOST.DIR文件夹。   1. dp与sp的区别   dp也就是dip。这个和sp基本类似。如果设置表示长度、高度等属性时可以使用dp 或sp。但如果设置字体，需要使用sp。dp是与密度无关，sp除了与密度无关外，还与scale无关。如果屏幕密度为160，这时dp和sp和px是一样的。1dp=1sp=1px，但如果使用px作单位，如果屏幕大小不变（假设还是3.2寸），而屏幕密度变成了320。那么原来TextView的宽度设成160px，在密度为320的3.2寸屏幕里看要比在密度为160的3.2寸屏幕上看短了一半。但如果设置成160dp或160sp的话。系统会自动将width属性值设置成320px的。也就是160 \* 320 / 160。其中320 / 160可称为密度比例因子。也就是说，如果使用dp和sp，系统会根据屏幕密度的变化自动进行转换。   1. java有那些排序，冒泡代码   for(inti=0;i<num.length;i++){  //内循环控制比较后移位  for(intj=num.length-1;j>i;j--){  if(num[j-1]>num[j]){  temp=num[j-1];  num[j-1]=num[j];  num[j]=temp;  }  }   1. Java线程实现方式和区别   继承Thread，实现runable接口。  区别：避免点继承的局限，一个类可以继承多个接口。  ?适合于资源的共享  详细出处参考：http://www.jb51.net/article/37465.htm   1. Android中布局、区别、应用场景   FrameLayout，LinearLayout，TableLayout，RelativeLayout，AbsoluteLayout   1. 线程间通讯   Handler   1. 进程间通讯   Intent 等   1. 软引用怎么用？为什么要用软引用   http://www.cnblogs.com/blogoflee/archive/2012/03/22/2411124.html  1)强引用 平时我们编程的时候例如：Object object=new Object（）；那object就是一个强引用了。如果一个对象具有强引用，那就类似于必不可少的生活用品，垃圾回收器绝不会回收它。当内存空 间不足，Java虚拟机宁愿抛出OutOfMemoryError错误，使程序异常终止，也不会靠随意回收具有强引用的对象来解决内存不足问题。  2)软引用（SoftReference） 如果一个对象只具有软引用，那就类似于可有可物的生活用品。如果内存空间足够，垃圾回收器就不会回收它，如果内存空间不足了，就会回收这些对象的内存。只 要垃圾回收器没有回收它，该对象就可以被程序使用。软引用可用来实现内存敏感的高速缓存。 软引用可以和一个引用队列（ReferenceQueue）联 合使用，如果软引用所引用的对象被垃圾回收，Java虚拟机就会把这个软引用加入到与之关联的引用队列中。 3)弱引用（WeakReference）  如果一个对象只具有弱引用，那就类似于可有可物的生活用品。弱引用与软引用的区别在于：只具有弱引用的对象拥有更短暂的生命周期。在垃圾回收器线程扫描它 所管辖的内存区域的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。不过，由于垃圾回收器是一个优先级很低的线程， 因此不一定会很快发现那些只具有弱引用的对象。 弱引用可以和一个引用队列（ReferenceQueue）联合使用，如果弱引用所引用的对象被垃圾回 收，Java虚拟机就会把这个弱引用加入到与之关联的引用队列中。  4)虚引用（PhantomReference）  "虚引用"顾名思义，就是形同虚设，与其他几种引用都不同，虚引用并不会决定对象的生命周期。如果一个对象仅持有虚引用，那么它就和没有任何引用一样，在 任何时候都可能被垃圾回收。 虚引用主要用来跟踪对象被垃圾回收的活动。虚引用与软引用和弱引用的一个区别在于：虚引用必须和引用队列 （ReferenceQueue）联合使用。当垃圾回收器准备回收一个对象时，如果发现它还有虚引用，就会在回收对象的内存之前，把这个虚引用加入到与之 关联的引用队列中。程序可以通过判断引用队列中是否已经加入了虚引用，来了解被引用的对象是否将要被垃圾回收。程序如果发现某个虚引用已经被加入到引用队 列，那么就可以在所引用的对象的内存被回收之前采取必要的行动。   1. 接口是否可继承接口   抽象类是否可实现(implements)接口? 抽象类是否可继承具体类(concrete class)? 抽象类中是否可以有静态的main方法  接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承具体类。抽象类中可以有静态的main方法。  备注：只要明白了接口和抽象类的本质和作用，这些问题都很好回答，你想想，如果你是java语言的设计者，你是否会提供这样的支持，如果不提供的话，有什么理由吗？如果你没有道理不提供，那答案就是肯定的了。  只有记住抽象类与普通类的唯一区别就是不能创建实例对象和允许有abstract方法。   1. '学java'内存中占多少个字节   10个字节，12   1. Intent可以传递那些数据   基本数据类型，以及序列化的对象。  Serializable  、charsequence  、Parcelable  、Bundle   1. DDMS与TraceView的区别   DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器。   1. 横竖屏切换不设置属性会怎么样   重新执行Activity生命周期   1. 单例类在android中哪种更合适   例如工具类可以封装为单例，用户登录后保存用户信息也可以封装为单例。   1. 如果标题过长超出界面，请写出你的解决方法   跑马灯的效果和HorizontalScrollView   1. 开发android应用怎样减少耗电量   建立连接前按Wifi-3G-2G顺序检查网络不要保持2G/3G连接wifi可用时，尽量用push，不要用定时查询优化应用程序的设计和算法使用简单高效的数据使用gzip压缩数据流>树-节点层次结构Ex,protobuf>json>xml   1. 对于instancevariable和localvarial,JVM在少女情怀内存时有什么区别   实例变量与本地变量在内存分配上的区别:实例变量在堆内存中,本地变量在栈中   1. Java线程都有哪些状态？列举你所知道的线程同步方法   新建状态(New)  就绪状态(Runnable)  运行状态(Running)  阻塞状态(Blocked)  死亡状态(Dead)   1. finish()与System.exit(0)的区别，再谈谈你对退出的理解。   首先一个Activity是有生命周期onCreate,onStart,onResume,onPause,onStop,onDestroy...  finish是Activity的类，仅仅针对Activity，当调用finish()时，只是将活动推向后台，并没有立即释放内存，活动的资源并没有被清理；当调用System.exit(0)时，杀死了整个进程，这时候活动所占的资源也会被释放。  其次，android的机制决定了用户无法完全退出应用，当你的application最长时间没有被用过的时候，android自身会决定将application关闭了。  finish方法可以控制像Activity一样的组件，使其结束生命周期，但应用程序全局的资源并不会立刻释放，像static变量的值仍然会保留，保以再次使用。   1. 反编译和防止反编译   android基于java的，而java反编译工具很强悍，所以对正常apk应用程序基本上可以做到100%反编译还原。  因此开发人员如果不准备开源自己的项目就需要知道怎样防止反编译和反编译他人的项目来学习。  2.3版本以上在eclipse自动生成的default.properties文件中加上一句“proguard.config=proguard.cfg”可以对代码进行混淆，反编译后是很难看懂的。  2.3之前的SDK版本也没关系，把上面的proguard.cfg文件复制一份放到项目中，然后进行相同的操作即可。  Android-proguard混淆器使用，有引用第三方jar包的时候报错：  Youmayneedtospecifyadditionallibraryjars(using'-libraryjars')  处理方法：在proguard.cfg文件顶部加入  -dontwarncom.motorola.\*\*  -keepclasscom.motorola.\*\*{\*;}   * + 1. 面试题\_Java  概念  1. 作用域public,private,protected,以及不写时的区别   AnonymousInnerClass(匿名内部类)是否可以extends(继承)其它类，是否可以implements(实现)interface(接口)  匿名的内部类是没有名字的内部类。不能extends(继承)其它类，但一个内部类可以作为一个接口，由另一个内部类实现。   1. StaticNestedClass和InnerClass的不同   "NestedClass（一般是C++的说法），InnerClass(一般是JAVA的说法)。Java内部类与C++嵌套类最大的不同就在于是否有指向外部的引用上。  静态内部类（InnerClass）意味着1创建一个static内部类的对象，不需要一个外部类对象，2不能从一个static内部类的一个对象访问一个外部类对象   1. &和&&的区别   &和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为true时，整个运算结果才为true，否则，只要有一方为false，则结果为false。  &&还具有短路的功能，即如果第一个表达式为false，则不再计算第二个表达式，例如，对于if(str!=null&&!str.equals(“”))表达式，当str为null时，后面的表达式不会执行，所以不会出现NullPointerException如果将&&改为&，则会抛出NullPointerException异常。If(x==33&++y>0)y会增长，If(x==33&&++y>0)不会增长  &还可以用作位运算符，当&操作符两边的表达式不是boolean类型时，&表示按位与操作，我们通常使用0x0f来与一个整数进行&运算，来获取该整数的最低4个bit位，例如，0x31&0x0f的结果为0x01   1. Collection和Collections的区别   Collection是集合类的上级接口，继承与他的接口主要有Set和List.  Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。   1. 什么时候用assert。   assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。在实现中，assertion就是在程序中的一条语句，它对一个boolean表达式进行检查，一个正确程序必须保证这个boolean表达式的值为true；如果该值为false，说明程序已经处于不正确的状态下，assert将给出警告或退出。一般来说，assertion用于保证程序最基本、关键的正确性。assertion检查通常在开发和测试时开启。为了提高性能，在软件发布后，assertion检查通常是关闭的。  package com.huawei.interview;  public class AssertTest {  /\*\*  \* @param args  \*/  public static void main(String[] args) {  // TODO Auto-generated method stub  int i = 0;  for(i=0;i<5;i++)  {  System.out.println(i);  }  //假设程序不小心多了一句--i;  --i;  assert i==5;  }  }   1. Strings=newString("xyz");创建了几个StringObject   是2个不假。一个是编译时决定的，最后放在常量池中。一个是运行时放在堆里面的。两个都是"xyz"。还有的，s确实只是一个引用，本身不是对象。java不是纯粹的面向对象，也就是说并不是所有的都是对象，如基本类型等   1. Math.round(11.5)等於多少?Math.round(-11.5)等於多少   Math.round(11.5)==12  Math.round(-11.5)==-11  round方法返回与参数最接近的长整数，   1. shorts1=1;s1=s1+1;有什么错?shorts1=1;s1+=1;有什么错   当然有错误，short占2个字节，int占4个字节，short型变量+上int型值计算结果是int型，占4个字节，然后付给一个short型变量s1，就要丢失2个字节的精度，这是不容许的。你非要这么做的话，我给你提供一个办法：  s1=s1+1；改成s1+=1；虽然是一个意思，但是在jvm里面是不同的方法做的，是运算符重载，所以是不一样的。   1. Java有没有goto   保留关键字   1. 数组有没有length()这个方法?String有没有length()这个方法   数组没有length()这个方法，有length的属性。String有length()这个方法   1. 给我一个你最常见到的runtimeexception   常见的运行时异常有如下这些ArithmeticException,ArrayStoreException,BufferOverflowException,BufferUnderflowException,CannotRedoException,CannotUndoException,ClassCastException,CMMException,ConcurrentModificationException,DOMException,EmptyStackException,IllegalArgumentException,IllegalMonitorStateException,IllegalPathStateException,IllegalStateException,ImagingOpException,IndexOutOfBoundsException,MissingResourceException,NegativeArraySizeException,NoSuchElementException,NullPointerException,ProfileDataException,ProviderException,RasterFormatException,SecurityException,SystemException,UndeclaredThrowableException,UnmodifiableSetException,UnsupportedOperationException   1. error和exception有什么区别   error表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。exception表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况   1. abstract的method   是否可同时是static,是否可同时是native，是否可同时是synchronized  都不可以，因为abstract申明的方法是要求子类去实现的，abstract只是告诉你有这样一个接口，你要去实现，至于你的具体实现可以是native和synchronized，也可以不是，抽象方法是不关心这些事的，所以写这两个是没有意义的。然后，static方法是不会被覆盖的，而abstract方法正是要子类去覆盖它，所以也是没有意义的。所以，总的来说，就是java语法不允许你这样做，事实上，也没有意义这样做。   1. 接口是否可继承接口   ?抽象类是否可实现(implements)接口?抽象类是否可继承实体类  (concreteclass)  1)接口可以继承接口..但是要使用extends~而不是用implements如:interfacea{}interfacebextendsa{}  2)抽象类可以实现接口..比如java.util中的AbstractCollection类就是实现的Collection接口  3)抽象类可以继承实体类下面这段执行无误的代码说明的所有的问题   1. 一个类实现接口，里面都变量Stringa;   接口中a=xxxx子类中是a=yyyy在子类中输出a=多少   1. 构造器Constructor是否可被override   构造器Constructor不能被继承，因此不能重写Override，但可以被重载Overload   1. 是否可以继承String类   String类是final类故不可以继承。   1. try{}里有一个return语句   try{}里有一个return语句，那么紧跟在这个try后的finally{}里的code会不会被执行，什么时候被执行，在return前还是后  根据java规范：在try-catch-finally中，如果try-finally或者catch-finally中都有return，则两个return语句都执行并且最终  返回到调用者那里的是finally中return的值；而如果finally中没有return，则理所当然的返回的是try或者catch中return的值，但是  finally中的代码是必须要执行的,而且是在return前执行,除非碰到exit()。   1. 用最有效率的方法算出2乘以8等於几   用移位运算inta=2<<3;  a就是2乘以8最后结果是16这是最省内存最有效率的方法  这个方法确实高效率的。我来解释一下：  2的二进制是10在32位存储器里面是0000000000000010  左移三位后变成0000000000010000也就是16   1. 当一个对象被当作参数传递到一个方法后   当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递  是值传递。Java编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。   1. swtich是否能作用在byte上   ，是否能作用在long上，是否能作用在String上  传递给switch和case语句的参数应该是int、short、char或者byte。long,string都不能作用于swtich   1. List,Set,Map是否继承自Collection接口   List和Set是继承自Collection接口的接口，Set不允许重复的项目，List允许重复项目，Set接口派生的类有TreeSet，HashSet，LinkedHashSet。List接口派生的类有ArrayList，Vector等。Map是独立的接口，不继承Collection接口   1. ArrayList和Vector的区别,HashMap和Hashtable的区别   答：就ArrayList与Vector主要从二方面来说.  1）同步性:Vector是线程安全的，也就是说是同步的，而ArrayList是线程序不安全的，不是同步的  2）数据增长:当需要增长时,Vector默认增长为原来一培，而ArrayList却是原来的一半  就HashMap与HashTable主要从三方面来说。  3）历史原因:Hashtable是基于陈旧的Dictionary类的，HashMap是Java1.2引进的Map接口的一个实现  1）同步性:Hashtable是线程安全的，也就是说是同步的，而HashMap是线程序不安全的，不是同步的  2）值：只有HashMap可以让你将空值作为一个表的条目的key或value   1. 说出ArrayList,Vector, LinkedList的存储性能和特性   ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，  它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，  Vector由于使用了synchronized方法（线程安全），通常性能上较ArrayList差，而LinkedList使用双向链表实现存储，  按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。   1. char型变量中能不能存贮一个中文汉字?为什么   char型变量是用来存储Unicode编码的字符的，unicode编码字符集中包含了汉字，所以，char型变量中当然可以存储汉字啦。不过，如果某个特殊的汉字没有被包含在unicode编码字符集中，那么，这个char型变量中就不能存储这个特殊汉字。补充说明：unicode编码占用两个字节，所以，char类型的变量也是占用两个字节。   1. 垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗   有什么办法主动通知虚拟机进行垃圾回收  对于GC来说，当程序员创建对象时，GC就开始监控这个对象的地址、大小以及使用情况。通常，GC采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对象是"可达的"，哪些对象是"不可达的"。当GC确定一些对象为"不可达"时，GC就有责任回收这些内存空间。可以。程序员可以手动执行System.gc()，通知GC运行，但是Java语言规范并不保证GC一定会执行。   1. 垃圾回收的优点和原理。并考虑2种回收机制。   Java语言中一个显著的特点就是引入了垃圾回收机制，使c++程序员最头疼的内存管理的问题迎刃而解，它使得Java程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java中的对象不再有"作用域"的概念，只有对象的引用才有"作用域"。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清楚和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。   1. float型floatf=3.4是否正确   不正确。精度不准确,应该用强制类型转换，如下所示：floatf=(float)3.4或floatf=3.4f  在java里面，没小数点的默认是int,有小数点的默认是double;   1. 介绍JAVA中的CollectionFrameWork(包括如何写自己的数据结构)   Collection  ├List  │├LinkedList  │├ArrayList  │└Vector  │　└Stack  └Set  Map  ├Hashtable  ├HashMap  └WeakHashMap  Collection是最基本的集合接口，一个Collection代表一组Object，即Collection的元素（Elements）  Map提供key到value的映射  事务属性ACID  答：原子性——一个事务要么完全执行，要么根本不执行。  一致性——一个事务在它完成后不能留下系统的不一致性。  隔离性——所有的事务不许允许在执行的时候不受其他进程或者事务的干涉。  持久性——在事务期间，所有提交的变更数据必须写入持久的数据存储器，并且应该能在硬件和软件失效中保存下来。   1. 实体Bean的三个状态：   答：no-state——Bean实例还没有创建。  pooled——Bean实例被创建，但还没有和一个EJBObject关联。  ready——与EJBObject相关联。若断开关联则回到pooled。   1. JDBC连接数据库6步   LoadtheJDBCDriver  EstablishtheDatabaseConnection  CreateaStatementObject  ExecuteaQuery  ProcesstheResults  ClosetheConnection   1. 事务的4大特性   答：原子性A，一致性C，隔离性I，永久性D   1. 查询语句之间的区别   selectcount（\*）fromstudent和selectcount（id）fromstudent之间的区别。  答案：  selectcount(\*)统计所有学生的记录个数，包括空记录。  Selectcount(Id)统计所有学生的记录个数，不包括null记录。   1. 假设现在有表system.table1   假设现在有表system.table1，表中有三个字段：id(数值型)、name（字符型）、age（数值型）写出SQL语句完成如下功能：在表中查出年龄大于20，且名字以“王”开头的记录，并且按照年龄的倒叙排列出来（年龄大的在前面）。  答案：  Select\*fromsystem.table1whereage>20andnamelike‘王%’orderbyageDESC;   1. 创建CUSTOMERS表   创建CUSTOMERS表，字段为：ID：（非空，主键）bigint，NAME：（非空）varchar，AGE：int类型；创建ORDERS表，字段为：ID：（非空，主键，）bigint，ORDER\_NUMBER：（非空）varchar，PRICE：double，CUSTOMER\_ID：（外键）bigint，设置级连删除；  答案：createtableCUSTOMBERS(  IDbigintnotnull,  NAMEvarchar(15),  AGEint,  primarykey(ID)  );  createtableORDERS(  IDbigintnotnull,  ORDER\_NUMBERvarchar(15)notnulll,  PRICEdoubleprecision,  CUSTOMER\_IDbigint,  primarykey(ID),  );  altertableORDERSaddconstraintFK\_CUSTOMERforeignkey(CUSTOMER\_ID)referencesCUSTOMERS(ID)ondeletecascade;   1. 使用左外连接查询，ORDERS和CUSTOMERS表，   答案：selectc.ID,o.CUSTOMER\_ID,c.NAME,o.IDORDER\_ID,ORDER\_NUMBERfromCUSTOMERScleftouterjoinORDERSonoc.ID=o.CUSTOMER\_ID;   1. 简述数据库事务的生命周期？（可画流程图）   答案：   1. deletefromtablea&truncatetabletablea的区别   　　truncate语句执行速度快,占资源少,并且只记录页删除的日志；  delete对每条记录的删除均需要记录日志   1. 什么是WebService?   　　WebService就是为了使原来各孤立的站点之间的信息能够相互通信、共享而提出的一种接口。  WebService所使用的是Internet上统一、开放的标准，如HTTP、XML、SOAP（简单对象访问协议）、WSDL等，所以WebService可以在任何支持这些标准的环境（Windows,Linux）中使用。  　　注：SOAP协议（SimpleObjectAccessProtocal,简单对象访问协议）,它是一个用于分散和分布式环境下网络信息交换的基于XML的通讯协议。在此协议下，软件组件或应用程序能够通过标准的HTTP协议进行通讯。它的设计目标就是简单性和扩展性，这有助于大量异构程序和平台之间的互操作性，从而使存在的应用程序能够被广泛的用户访问。  　　优势：  　　(1).跨平台。  　　(2).SOAP协议是基于XML和HTTP这些业界的标准的，得到了所有的重要公司的支持。  　　(3).由于使用了SOAP，数据是以ASCII文本的方式而非二进制传输，调试很方便；并且由于这样，它的数据容易通过防火墙，不需要防火墙为了程序而单独开一个“漏洞”。  　　(4).此外，WebService实现的技术难度要比CORBA和DCOM小得多。  　　(5).要实现B2B集成，EDI比较完善与比较复杂；而用WebService则可以低成本的实现，小公司也可以用上。  　　(6).在C/S的程序中，WebService可以实现网页无整体刷新的与服务器打交道并取数。  　　缺点：  　　(1).WebService使用了XML对数据封装，会造成大量的数据要在网络中传输。  　　(2).WebService规范没有规定任何与实现相关的细节，包括对象模型、编程语言，这一点，它不如CORBA。   1. 运行时异常与一般异常有何异同？   异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。   1. 说出Servlet的生命周期，并说出Servlet和CGI的区别。   Servlet被服务器实例化后，容器运行其init方法，请求到达时运行其service方法，service方法自动派遣运行与请求对应的doXXX方法（doGet，doPost）等，当服务器决定将实例销毁的时候调用其destroy方法。  与cgi的区别在于servlet处于服务器进程中，它通过多线程方式运行其service方法，一个实例可以服务于多个请求，并且其实例一般不会销毁，而CGI对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于servlet。   1. 同步和异步有何异同，在什么情况下分别使用他们？举例说明。   如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。  当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。   1. heap和stack有什么区别   栈是一种线形集合，其添加和删除元素的操作应在同一段完成。栈按照后进先出的方式进行处理。  堆是栈的一个组成元素   1. forward和redirect的区别   forward是服务器请求资源，服务器直接访问目标地址的URL，把那个URL的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。  redirect就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session,request参数都可以获取。   1. 说出数据连接池的工作机制是什么   J2EE服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。   1. Set里的元素是不能重复的，那么用什么方法来区分重复与否呢   是用==还是equals()?它们有何区别  1)什么是Set?(what)  Set是Collection容器的一个子接口，它不允许出现重复元素，当然也只允许有一个null对象。  2)如何来区分重复与否呢？(how)  “用iterator()方法来区分重复与否”，这是在网上流传的答案，个人认为这是个错误的答案。JPI中写的很明白：“set不包含满足  e1.equals(e2)的元素对e1和e2”，由此可见回答使用equals()区分更合适。  3)为什么用equals()而不用==来区分？(why)  应该从它俩的区别谈起，==是用来判断两者是否是同一对象（同一事物），而equals是用来判断是否引用同一个对象。再看一下Set里面存的是  对象，还是对象的引用。根据java的存储机制可知，set里面存放的是对象的引用，所以当两个元素只要满足了equals()时就已经指向同一个对象，  也就出现了重复元素。所以应该用equals()来判断。   1. 两个对象值相同(x.equals(y) == true)   ，但却可有不同的hash code，这句话对不对  对。  如果对象要保存在HashSet或HashMap中，它们的equals相等，那么，它们的hashcode值就必须相等。  如果不是要保存在HashSet或HashMap，则与hashcode没有什么关系了，这时候hashcode不等是可以的，例如arrayList存储的对象就不用实现hashcode，当然，我们没有理由不实现，通常都会去实现的。   1. 重写equals()方法的时候，我们还需要重写哪个方法？为什么   每个覆盖了equals方法的类中，也必须覆盖hashCode方法。 如果不这样的话，就会违反Object.hashCode的通用约定，从而导致该类无法结合所有基于散列的集合一起正常运作，这样的集合包括HashMap、HashSet和Hashtable。   1. 当一个对象被当作参数传递到一个方法后   此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递?  是值传递。Java编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。   1. java中有几种方法可以实现一个线程   用什么关键字修饰同步方法?stop()和suspend()方法为何不推荐使用  有两种实现方法，分别是继承Thread类与实现Runnable接口  用synchronized关键字修饰同步方法  反对使用stop()，是因为它不安全。它会解除由线程获取的所有锁定，而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。结果很难检查出真正的问题所在。suspend()方法容易发生死锁。调用suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被"挂起"的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。所以不应该使用suspend()，而应在自己的Thread类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用wait()命其进入等待状态。若标志指出线程应当恢复，则用一个notify()重新启动线程。   1. 简述synchronized和java.util.concurrent.locks.Lock的异同   主要相同点：Lock能完成synchronized所实现的所有功能  主要不同点：Lock有比synchronized更精确的线程语义和更好的性能。synchronized会自动释放锁，而Lock一定要求程序员手工释放，并且必须在finally从句中释放。   1. 当一个线程进入一个对象的一个synchronized方法后   ，其它线程是否可进入此对象的其它方法?  不能，一个对象的一个synchronized方法只能由一个线程访问。   1. Java的接口和C++的虚类的相同和不同处。   由于Java不支持多继承，而有可能某个类或对象要使用分别在几个类或对象里面的方法或属性，现有的单继承机制就不能满足要求。与继承相比，接口有更高的灵活性，因为接口中没有任何实现代码。当一个类实现了接口以后，该类要实现接口里面所有的方法和属性，并且接口里面的属性在默认状态下面都是publicstatic,所有方法默认情况下是public.一个类可以实现多个接口。   1. Java中的异常处理机制的简单原理和应用。   当JAVA程序违反了JAVA的语义规则时，JAVA虚拟机就会将发生的错误表示为一个异常。违反语义规则包括2种情况。一种是JAVA类库内置的语义检查。例如数组下标越界,会引发IndexOutOfBoundsException;访问null的对象时会引发NullPointerException。另一种情况就是JAVA允许程序员扩展这种语义检查，程序员可以创建自己的异常，并自由选择在何时用throw关键字引发异常。所有的异常都是java.lang.Thowable的子类。   1. 描述一下JVM加载class文件的原理机制?   JVM中类的装载是由ClassLoader和它的子类来实现的,JavaClassLoader是一个重要的Java运行时系统组件。它负责在运行时查找和装入类文件的类。   1. 线程的基本概念、线程的基本状态以及状态之间的关系   线程指在程序执行过程中，能够执行程序代码的一个执行单位，每个程序至少都有一个线程，也就是程序本身。  Java中的线程有四种状态分别是：运行、就绪、挂起、结束。   1. 什么情况下调用doGet()和doPost()   Jsp页面中的form标签里的method属性为get时调用doGet()，为post时调用doPost()。  请求中的一个属性   1. 我们在web应用开发过程中经常遇到输出某种编码的字符   ，如iso8859-1等，如何输出一个某种编码的字符串  PublicStringtranslate(Stringstr){  StringtempStr="";  try{  tempStr=newString(str.getBytes("ISO-8859-1"),"GBK");  tempStr=tempStr.trim();  }  catch(Exceptione){  System.err.println(e.getMessage());  }  returntempStr;  }   1. 排序都有哪几种方法？请列举。用JAVA实现一个快速排序。   排序的方法有：插入排序（直接插入排序、希尔排序），交换排序（冒泡排序、快速排序），选择排序（直接选择排序、堆排序），归并排序，分配排序（箱排序、基数排序）  快速排序的伪代码。  //使用快速排序方法对a[0:n-1]排序  从a[0:n-1]中选择一个元素作为middle，该元素为支点  把余下的元素分割为两段left和right，使得left中的元素都小于等于支点，而right中的元素都大于等于支点  递归地使用快速排序方法对left进行排序  递归地使用快速排序方法对right进行排序  所得结果为left+middle+right   1. JAVA语言如何进行异常处理   关键字：throws,throw,try,catch,finally分别代表什么意义？在try块中可以抛出异常吗？  Java通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并提供了良好的接口。在Java中，每个异常都是一个对象，它是Throwable类或其它子类的实例。当一个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个对象的方法可以捕获到这个异常并进行处理。Java的异常处理是通过5个关键词来实现的：try、catch、throw、throws和finally。一般情况下是用try来执行一段程序，如果出现异常，系统会抛出（throws）一个异常，这时候你可以通过它的类型来捕捉（catch）它，或最后（finally）由缺省处理器来处理。  用try来指定一块预防所有"异常"的程序。紧跟在try程序后面，应包含一个catch子句来指定你想要捕捉的"异常"的类型。  throw语句用来明确地抛出一个"异常"。  throws用来标明一个成员函数可能抛出的各种"异常"。  Finally为确保一段代码不管发生什么"异常"都被执行一段代码。  可以在一个成员函数调用的外面写一个try语句，在这个成员函数内部写另一个try语句保护其他代码。每当遇到一个try语句，"异常"的框架就放到堆栈上面，直到所有的try语句都完成。如果下一级的try语句没有对某种"异常"进行处理，堆栈就会展开，直到遇到有处理这种"异常"的try语句。   1. 一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制   可以。必须只有一个类名与文件名相同。   1. java中有几种类型的流   JDK为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？  字节流，字符流。字节流继承于InputStreamOutputStream，字符流继承于InputStreamReaderOutputStreamWriter。在java.io包中还有许多其他的流，主要是为了提高性能和使用方便。   1. 字节流与字符流的区别   要把一片二进制数据数据逐一输出到某个设备中，或者从某个设备中逐一读取一片二进制数据，不管输入输出设备是什么，我们要用统一的方式来完成这些操作，用一种抽象的方式进行描述，这个抽象描述方式起名为IO流，对应的抽象类为OutputStream和InputStream ，不同的实现类就代表不同的输入和输出设备，它们都是针对字节进行操作的。  在应用中，经常要完全是字符的一段文本输出去或读进来，用字节流可以吗？计算机中的一切最终都是二进制的字节形式存在。对于“中国”这些字符，首先要得到其对应的字节，然后将字节写入到输出流。读取时，首先读到的是字节，可是我们要把它显示为字符，我们需要将字节转换成字符。由于这样的需求很广泛，人家专门提供了字符流的包装类。  底层设备永远只接受字节数据，有时候要写字符串到底层设备，需要将字符串转成字节再进行写入。字符流是字节流的包装，字符流则是直接接受字符串，它内部将串转成字节，再写入底层设备，这为我们向IO设别写入或读取字符串提供了一点点方便。  字符向字节转换时，要注意编码的问题，因为字符串转成字节数组，  其实是转成该字符的某种编码的字节形式，读取也是反之的道理。  讲解字节流与字符流关系的代码案例：  import java.io.BufferedReader;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.FileReader;  import java.io.FileWriter;  import java.io.InputStreamReader;  import java.io.PrintWriter;  public class IOTest {  public static void main(String[] args) throws Exception {  String str = "中国人";  /\*FileOutputStream fos = new FileOutputStream("1.txt");    fos.write(str.getBytes("UTF-8"));  fos.close();\*/    /\*FileWriter fw = new FileWriter("1.txt");  fw.write(str);  fw.close();\*/  PrintWriter pw = new PrintWriter("1.txt","utf-8");  pw.write(str);  pw.close();    /\*FileReader fr = new FileReader("1.txt");  char[] buf = new char[1024];  int len = fr.read(buf);  String myStr = new String(buf,0,len);  System.out.println(myStr);\*/  /\*FileInputStream fr = new FileInputStream("1.txt");  byte[] buf = new byte[1024];  int len = fr.read(buf);  String myStr = new String(buf,0,len,"UTF-8");  System.out.println(myStr);\*/  BufferedReader br = new BufferedReader(  new InputStreamReader(  new FileInputStream("1.txt"),"UTF-8"  )  );  String myStr = br.readLine();  br.close();  System.out.println(myStr);  }  }   1. java中会存在内存泄漏吗，请简单描述。   会。如：inti,i2;return(i-i2);//wheni为足够大的正数,i2为足够大的负数。结果会造成溢位，导致错误。   1. 什么是java序列化，如何实现java序列化？   或者请解释Serializable接口的作用。  我们有时候将一个java对象变成字节流的形式传出去或者从一个字节流中恢复成一个java对象，例如，要将java对象存储到硬盘或者传送给网络上的其他计算机，这个过程我们可以自己写代码去把一个java对象变成某个格式的字节流再传输，但是，jre本身就提供了这种支持，我们可以调用OutputStream的writeObject方法来做，如果要让java 帮我们做，要被传输的对象必须实现serializable接口，这样，javac编译时就会进行特殊处理，编译的类才可以被writeObject方法操作，这就是所谓的序列化。需要被序列化的类必须实现Serializable接口，该接口是一个mini接口，其中没有需要实现的方法，implements Serializable只是为了标注该对象是可被序列化的。  例如，在web开发中，如果对象被保存在了Session中，tomcat在重启时要把Session对象序列化到硬盘，这个对象就必须实现Serializable接口。如果对象要经过分布式系统进行网络传输或通过rmi等远程调用，这就需要在网络上传输对象，被传输的对象就必须实现Serializable接口。   1. 是否可以从一个static方法内部发出对非static方法的调用   不可以,如果其中包含对象的method()；不能保证对象初始化.   1. 写clone()方法时，通常都有一行代码，是什么   clone 有缺省行为，super.clone();因为首先要把父类中的成员复制到位，然后才是复制自己的成员。   1. 在JAVA中，如何跳出当前的多重嵌套循环   用break;return方法。   1. 说出一些常用的类，包，接口，请各举5个   常用的类：BufferedReaderBufferedWriterFileReaderFileWirterStringInteger  常用的包：java.langjava.awtjava.iojava.utiljava.sql  常用的接口：RemoteListMapDocumentNodeList   1. 内部类可以引用他包含类的成员吗？有没有什么限制   一个内部类对象可以访问创建它的外部类对象的内容   1. 面向对象的特征有哪些方面   **答：**主要有以下四方面：  1）抽象：  抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。抽象包括两个方面，一是过程抽象，二是数据抽象。  2）继承：  继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。  3）封装：  封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。  4）多态性：  多态性是指允许不同类的对象对同一消息作出响应。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。   1. javac命令和java命令做什么事情呢   要知道java是分两部分的：一个是编译，一个是运行。  javac：负责的是编译的部分，当执行javac时，会启动java的编译器程序。对指定扩展名的.java文件进行编译。 生成了jvm可以识别的字节码文件。也就是class文件，也就是java的运行程序。  java：负责运行的部分.会启动jvm.加载运行时所需的类库,并对class文件进行执行.  一个文件要被执行,必须要有一个执行的起始点,这个起始点就是main函数.   1. Class.forName的作用?为什么要用   答：按参数中指定的字符串形式的类名去搜索并加载相应的类，如果该类字节码已经被加载过，则返回代表该字节码的Class实例对象，否则，按类加载器的委托机制去搜索和加载该类，如果所有的类加载器都无法加载到该类，则抛出ClassNotFoundException。加载完这个Class字节码后，接着就可以使用Class字节码的newInstance方法去创建该类的实例对象了。  有时候，我们程序中所有使用的具体类名在设计时（即开发时）无法确定，只有程序运行时才能确定，这时候就需要使用Class.forName去动态加载该类，这个类名通常是在配置文件中配置的，例如，spring的ioc中每次依赖注入的具体类就是这样配置的，jdbc的驱动类名通常也是通过配置文件来配置的，以便在产品交付使用后不用修改源程序就可以更换驱动类名。   1. 为什么要用 ORM? 和 JDBC 有何不一样?   orm是一种思想，就是把object转变成数据库中的记录，或者把数据库中的记录转变成objecdt，我们可以用jdbc来实现这种思想，其实，如果我们的项目是严格按照oop方式编写的话，我们的jdbc程序不管是有意还是无意，就已经在实现orm的工作了。  现在有许多orm工具，它们底层调用jdbc来实现了orm工作，我们直接使用这些工具，就省去了直接使用jdbc的繁琐细节，提高了开发效率，现在用的较多的orm工具是hibernate。也听说一些其他orm工具，如toplink,ojb等。   1. JDBC中的PreparedStatement相比Statement的好处   答：一个sql命令发给服务器去执行的步骤为：语法检查，语义分析，编译成内部指令，缓存指令，执行指令等过程。  select \* from student where id =3----缓存--xxxxx二进制命令  select \* from student where id =3----直接取-xxxxx二进制命令  select \* from student where id =4--- -会怎么干？  如果当初是select \* from student where id =?--- -又会怎么干？  上面说的是性能提高  可以防止sql注入。   1. 静态变量和实例变量的区别   在语法定义上的区别：静态变量前要加static关键字，而实例变量前则不加。  在程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量。静态变量不属于某个实例对象，而是属于类，所以也称为类变量，只要程序加载了类的字节码，不用创建任何实例对象，静态变量就会被分配空间，静态变量就可以被使用了。总之，实例变量必须创建对象后才可以通过这个对象来使用，静态变量则可以直接使用类名来引用。  例如，对于下面的程序，无论创建多少个实例对象，永远都只分配了一个staticVar变量，并且每创建一个实例对象，这个staticVar就会加1；但是，每创建一个实例对象，就会分配一个instanceVar，即可能分配多个instanceVar，并且每个instanceVar的值都只自加了1次。  public class VariantTest{  public static int staticVar = 0;  public int instanceVar = 0;  public VariantTest(){  staticVar++;  instanceVar++;  System.out.println(“staticVar=” + staticVar + ”,instanceVar=” + instanceVar);  }  }  备注：这个解答除了说清楚两者的区别外，最后还用一个具体的应用例子来说明两者的差异，体现了自己有很好的解说问题和设计案例的能力，思维敏捷，超过一般程序员，有写作能力！   1. 请说出作用域public，private，protected，以及不写时的区别   这四个作用域的可见范围如下表所示。  说明：如果在修饰的元素上面没有写任何访问修饰符，则表示friendly。  作用域 当前类 同一package 子孙类 其他package  public √ √ √ √  protected √ √ √ ×  friendly √ √ × ×  private √ × × ×  备注：只要记住了有4种访问权限，4个访问范围，然后将全选和范围在水平和垂直方向上分别按排从小到大或从大到小的顺序排列，就很容易画出上面的图了。   1. 构造函数与方法的区别   1）构造方法是初始化一个类的对象时调用的，它没有返回值，而且名字必须与类的名字一样， public class wan { public static void main(String args[]) { String str=new String("HelloWorld");//String类，String（）为构造方法，初始化一个对象str System.out.println(str); }  } 2）而成员函数是由类对象主动调用的，使用点操作符，它有返回值 。 举个例子： public class wan  { public static void main(String[] args) { String str=new String("Hello!"); System.out.println(str); System.out.println("将Hello转化成全部大写："+str.toUpperCase()); System.out.println("将Hello转化成全部小写"+str.toLowerCase()); System.out.println("成员方法String（）内的字符长度是："+str.length()); } } 输出结果是： Hello! 将Hello转化成全部大写：HELLO! 将Hello转化成全部小写hello! 成员方法String（）内的字符长度是：6总之，构造函数是赋予对象"生命特征"的,成员方法则是对象所具有的"行为,能力" 构造函数只有在建立对象时由系统调用的,其他任何时候你都别指望用他   1. 对于instance varisble 和 local variable jvm 再申请内存时有什么区别   根据声明方式详细来区分，Java的变量有七种,下面的程序代码展示了这七种变量的声明方式：  Class Myclass {  static int a;  int b;  public static void myMethod(int c) {  try {  int d;  } catch(Exception e) {  }  }  MyClass(int f) {  int[]g = new int[100];  }  }  class variable:声明在class内，method之外，且使用static修饰的变量,例如上面程序代码的a.  instance variable:声明在class内，method之外，且未使用static修饰的变量，例如上面程序的b.  method parameter:声明在method小括号内的变量，例如上面程序代码的c.  狭义的局部变量(local variable):声明在method内的变量，例如上面程序代码的d和g.  exception-handler parameter:声明在catch小括号内的变量，例如上面程序代码的e.  constructor parameter:声明在constructor小括号内的变量，例如上面程序代码的f.  数组元素(array element):数组的元素值没有识别名称，必须透过数组和索引值(index)来识别.例如上面  程序代码的g[0].  根据变量内存来分类  根据变量内存位置来区分，Java的变量有两种，包括了:  heap variable:占用的内存在heap中，这类变量包括了class variable,instance variable,array component,  即前面程序的a,b,g[0].这类变量会自动被JVM初始化默认值.  stack variable:通常广义的局部变量(pan-local variable),其占的内存在stack中，这类变量包括了狭义的  局部变量，method parameter,exception-handler parameter,constructor parameter,即前面程序 的c,d,e,f.狭义 的局部变量不会被JVM初始化成默认值，使用者必须自行初始化该变量,但是parameter类(包括method parameter,exception-handler parameter,constructor parameter)会被JVM初始化成传入值.  根据使用方式来为变量分类  根据使用方式，只要分三类即可，分别是:  class variable:即上例的a.  instance variable:即上例的b.  广义的局部变量:包含上例的c,d,e,f.这四者的差别很小，直接归为一类.  至于"数组元素"(array component)则不在此三类中，但是"数组元素"并不常被注意到，为它多分出一类的用处不大.我通常将数组视为对象,将array component视为对象的instance variable.   1. 如何设置两个应用在一个进程中 2. 在<manifest>里面添加： android:sharedUserId，注意这个属性的取值必须包含点（dot），也就是诸如java packae的形式。比如com.aaa.bbb。没有dot的话，将来adb install xxx.apk就会出错：Failure [INSTALL\_PARSE\_FAILED\_BAD\_SHARED\_USER\_ID]。最晕死的就是这一点在Android文档中没有提到，感谢万能的google赐予了我答案。所有application都要填写的一样。 3. 在<manifest>里面添加：android:sharedUserLabel="@string/shared\_user\_label"，这个label必须是一个string资源，不能是raw string。所有application都要填写的一样。 4. 在<application>里面添加：android:process="xxx.xxx.xxx"，这里所有的application都要填写的一样，内容就是process的名字，一般来说Android中process的名字就是manifest中的package的取值。 5. 所有的application用同样的一个key来sign。如果用Eclipse ADT plugin开发，由于所有的application都使用同一个debug key来sign，所以这一步没有什么额外的工作。更具体的有关sign apk的细节，参考Android文档：http://androidappdocs.appspot.com/guide/publishing/app-signing.html 6. android中常用的权限，最少写三个   读写权限、调用联系人数据的权限、网络权限、调用照相机硬件的权限   1. IO问题   有一个文件夹E://555下面有文件a.txt,b.txt,c.txt通过Java.io.file判断是文件夹还是文件，并获取文件夹以及文件的名字   1. java内存中占多少个字节   6   1. Android开发心得:思想，进程间通信，四大组件 |
|  |
|  |
|  |

2016Android 面试收集

**1、请解释下在单线程模型中Message,Handler,Message Queue,Looper之间的关系。**

拿主线程来说，主线程启动时会调用Looper.prepare()方法，会初始化一个Looper，放入Threadlocal中，接着调用Looper.loop()不断遍历Message Queue，

Handler的创建依赖与当前线程中的Looper，如果当前线程没有Looper则必须调用Looper.prepare()。Handler , sendMessage到MessageQueue，Looper不断

从MessageQueue中取出消息，回调handleMessage方法。

**2、如果有个100M大的文件，需要上传至服务器中，而服务器form表单最大只能上传2M，可以用什么方法。**

这个问题不是很明确我觉得，首先来说使用http协议上传数据，特别在android下，跟form没什么关系。传统的在web中，在form中写文件上传，其实浏览器所做

的就是将我们的数据进行解析组拼成字符串，以流的方式发送到服务器，且上传文件用的都是POST方式，POST方式对大小没什么限制。

回到题目，可以说假设每次真的只能上传2M，那么可能我们只能把文件截断，然后分别上传了。

**3、内存溢出和内存泄漏有什么区别？何时会产生内存泄漏？内存优化有哪些方法？**

内存溢出通俗理解就是软件（应用）运行需要的内存，超出了它可用的最大内存。

内存泄漏就是我们对某一内存空间的使用，使用完成后没有释放。

内存优化：Android中容易内存溢出的部分，就是图片的加载，我们可以使用图片的压缩加上使用LruCache缓存的目的来控制图片所能够使用的内存。

还有对于比较耗资源的对象及时的关闭，例如Database Conn , 各种传感器 ， Service 等等。

**4、AsyncTask使用在哪些场景？它的缺陷是什么？如何解决？**

AsyncTask 运用的场景就是我们需要进行一些耗时的操作，耗时操作完成后更新主线程，或者在操作过程中对主线程的UI进行更新。

缺陷：AsyncTask中维护着一个长度为128的线程池，同时可以执行5个工作线程，还有一个缓冲队列，当线程池中已有128个线程，缓冲队列已满时，如果

此时向线程提交任务，将会抛出RejectedExecutionException。

解决：由一个控制线程来处理AsyncTask的调用判断线程池是否满了，如果满了则线程睡眠否则请求AsyncTask继续处理。

**7、assest文件夹里放文件，对于文件的大小有没有限制？22**

assets目录更像一个附录类型的目录，Android不会为这个目录中的文件生成ID并保存在R类当中，因此它与Android中的一些类和方法兼容度更低。

同时，由于你需要一个字符串路径来获取这个目录下的文件描述符，访问的速度会更慢。但是把一些文件放在这个目录下会使一些操作更加方便，

比方说拷贝一个数据库文件到系统内存中。要注意的是，你无法在Android XML文件中引用到assets目录下的文件，只能通过AssetManager来访问

这些文件。数据库文件和游戏数据等放在这个目录下是比较合适的。另外，网上关于assets和raw的资料都千篇一律了，因此关于这两者中单个文件

大小不能超过1M的\*\*错误\*\*描述也在传播，即如果读取超过1M的文件会报"Data exceeds UNCOMPRESS\_DATA\_MAX (1314625 vs 1048576)"的

IOException，还引申出种种解决方案。个人认为不应该有这样的限制，为了验证这个说法写了个Demo，发现将近5M的压缩包在assets和raw中

都能正常访问，因此在这里纠正一下，理论上只要打包不超过Android APK 50M大小的限制都是没有问题的。当然了，不排除是Android很早期的

时候因为设备硬件原因aapt在编译的时候对这两个文件夹大小做出了限制，如果是这样，较新版的ADT应该不会出现这种情况。

来自：http://my.eoe.cn/futurexiong/archive/5350.html

**8、 启动一个程序，可以主界面点击图标进入，也可以从一个程序中跳转过去，二者有什么区别？**

是因为启动程序（主界面也是一个app），发现了在这个程序中存在一个设置为<category android:name="android.intent.category.LAUNCHER" />的activity,

所以这个launcher会把icon提出来，放在主界面上。当用户点击icon的时候，发出一个Intent：

Intent intent = mActivity.getPackageManager().getLaunchIntentForPackage(packageName);

mActivity.startActivity(intent);

跳过去可以跳到任意允许的页面，如一个程序可以下载，那么真正下载的页面可能不是首页（也有可能是首页），这时还是构造一个Intent，startActivity.

这个intent中的action可能有多种view,download都有可能。系统会根据第三方程序向系统注册的功能，为你的Intent选择可以打开的程序或者页面。所以唯一的一点

不同的是从icon的点击启动的intent的action是相对单一的，从程序中跳转或者启动可能样式更多一些。本质是相同的。

**9、程序之间的亲和性的理解。**

1、默认情况下一个应用的所有Activity都是具有相同的affinity，都是从application中继承，application的affinity默认就是manifest的包名。

2、affinity对Activity来说，就像是身份证一样，可以告诉所在的Task，自己属于其中的一员。

3、应用场合：

a:根据affinity重新为Activity选择合适的宿主Task;

b:与allowTaskReparenting属性配合;

c:启动Activity使用Intent设置了FLAG\_ACTIVITY\_NEW\_TASK标记。

**10、同一个程序，但不同的Activity是否可以放在不同的Task任务栈中？**

可以放在不同的Task中。需要为不同的activity设置不同的affinity属性，启动activity的Intent需要包含FLAG\_ACTIVITY\_NEW\_TASK标记。

**11、横竖屏切换时候Activity的生命周期。**

1、不设置Activity的android:configChanges时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次

2、设置Activity的android:configChanges="orientation"时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次

3、设置Activity的android:configChanges="orientation|keyboardHidden"时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法

**12、AIDL的全称是什么？如何工作？**

全称是：Android Interface Define Language

在Android中, 每个应用程序都可以有自己的进程. 在写UI应用的时候, 经常要用到Service. 在不同的进程中, 怎样传递对象呢? 显然, Java中不允许跨进程内存共享.

 因此传递对象, 只能把对象拆分成操作系统能理解的简单形式, 以达到跨界对象访问的目的. 在J2EE中,采用RMI的方式, 可以通过序列化传递对象. 在Android中, 则

采用AIDL的方式. 理论上AIDL可以传递Bundle,实际上做起来却比较麻烦。

AIDL(AndRoid接口描述语言)是一种借口描述语言; 编译器可以通过aidl文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的. 如果需要

在一个Activity中, 访问另一个Service中的某个对象, 需要先将对象转化成AIDL可识别的参数(可能是多个参数), 然后使用AIDL来传递这些参数, 在消息的接收端, 使用

这些参数组装成自己需要的对象.AIDL的IPC的机制和COM或CORBA类似, 是基于接口的，但它是轻量级的。它使用代理类在客户端和实现层间传递值. 如果要使用AIDL,

需要完成2件事情: 1. 引入AIDL的相关类.; 2. 调用aidl产生的class.

AIDL的创建方法:

AIDL语法很简单,可以用来声明一个带一个或多个方法的接口，也可以传递参数和返回值。 由于远程调用的需要, 这些参数和返回值并不是任何类型.

下面是些AIDL支持的数据类型:

1. 不需要import声明的简单Java编程语言类型(int,boolean等)

2. String, CharSequence不需要特殊声明

3. List, Map和Parcelables类型, 这些类型内所包含的数据成员也只能是简单数据类型, String等其他比支持的类型.

(另外: 我没尝试Parcelables, 在Eclipse+ADT下编译不过, 或许以后会有所支持

**13、dvm的进程和Linux的进程, 应用程序的进程是否为同一个概念**

      Dvm的进程是dalivk虚拟机进程,每个android程序都运行在自己的进程里面,每个android程序系统都会给他分配一个单独的liunx uid(user id),  
每个dvm都是linux里面的一个进程.所以说这两个进程是一个进程.

1. 什么是Activity?

四大组件之一,一般的,一个用户交互界面对应一个activity, activity 是Context的子类,同时实现了window.callback和keyevent.callback, 可以处理与窗体用户交互的事件. 我开发常用的的有ListActivity , PreferenceActivity 等…如果界面有共同的特点或者功能的时候,还会自己定义一个BaseActivity.

2. 请描述一下Activity生命周期。

生命周期描述的是一个类 从创建(new出来)到死亡(垃圾回收)的过程中会执行的方法..

在这个过程中 会针对不同的生命阶段会调用不同的方法

Activity从创建到销毁有多种状态，从一种状态到另一种状态时会激发相应的回调方法，这些回调方法包括：oncreate ondestroy onstop onstart onresume onpause

其实这些方法都是两两对应的，onCreate创建与onDestroy销毁；

onStart可见与onStop不可见；onResume可编辑（即焦点）与onPause；

这6个方法是相对应的，那么就只剩下一个onRestart方法了，这个方法在什么时候调用呢？

答案就是：在Activity被onStop后，但是没有被onDestroy，在再次启动此Activity时就调用onRestart（而不再调用onCreate）方法；

如果被onDestroy了，则是调用onCreate方法。

最后讲自己项目中的经验,比如说豆瓣客户端每次进入某个界面的时候要刷新列表,这个刷新列表的操作 就放在onStart()的方法里面.这样保证每次用户看到的数据都是最新的.

多媒体播放, 播放来电话. onStop() 视频, 视频声音设置为0 , 记录视频播放的位子

onStart() 根据保存的状态恢复现场.

在读文档的时候 还发现 activity还有两个方法 onPostResume() 和 OnPostCreate()这两个生命周期的方法,不过开发的时候没有用到过.

3. 两个Activity之间跳转时必然会执行的是哪几个方法。

一般情况比如说有两个activity,分别叫A,B ,当在A里面激活B组件的时候, A 会调用 onPause()方法,然后B 调用onCreate() ,onStart(), OnResume() , 这个时候B覆盖了窗体, A会调用onStop()方法. 如果B呢 是个透明的,或者是对话框的样式, 就不会调用onStop()方法

4. 横竖屏切换时候Activity的生命周期。

这个生命周期跟清单文件里的配置有关系

1、不设置Activity的android:configChanges时，切屏会重新调用各个生命周期

默认首先销毁当前activity,然后重新加载

2、设置Activity的android:configChanges="orientation|keyboardHidden"时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法

5. 如何将一个Activity设置成窗口的样式。

可以自定义一个activity的样式,详细见手机卫士的程序详细信息

android:theme="@style/FloatActivity"

E:\day9\mobilesafe\res\values\style

6. 你后台的Activity被系统 回收怎么办？如果后台的Activity由于某原因被系统回收可了，如何在被系统回收之前保存当前状态？

除了在栈顶的activity,其他的activity都有可能在内存不足的时候被系统回收,一个activity越处于栈底,被回收的可能性越大.

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

outState.putLong("id", 1234567890);

}

public void onCreate(Bundle savedInstanceState) {

//判断 savedInstanceState是不是空.

//如果不为空就取出来

super.onCreate(savedInstanceState);

}

7. 如何退出Activity？如何安全退出已调用多个Activity的Application？

1、抛异常强制退出：

该方法通过抛异常，使程序Force Close。

验证可以，但是，需要解决的问题是，如何使程序结束掉，而不弹出Force Close的窗口。

2、记录打开的Activity：

每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可。

3、发送特定广播：

在需要结束应用时，发送一个特定的广播，每个Activity收到广播后，关闭即可。

4、递归退出

在打开新的Activity时使用startActivityForResult，然后自己加标志，在onActivityResult中处理，递归关闭。

上面是网上的一些做法.

其实 可以通过 intent的flag 来实现.. intent.setFlag(FLAG\_ACTIVITY\_CLEAR\_TOP)激活一个新的activity,然后在新的activity的oncreate方法里面 finish掉.

讲一讲你对activity的理解

把上面的几点用自己的心得写出来

8. service是否在main thread中执行, service里面是否能执行耗时的操作?

默认情况,如果没有显示的指定service所运行的进程, Service和activity是运行在当前app所在进程的main thread(UI主线程)里面

9. 两个Activity之间怎么传递数据？

基本数据类型可以通过. Intent 传递数据

extras.putDouble(key, value)

intent.putExtras(extras)

Application 全局里面存放 对象 ,自己去实现自己的application的这个类,基础系统的application , 每个activity都可以取到

让对象实现 implements Serializable 接口把对象存放到文件上.

让类实现Serializable 接口,然后可以通过 ObjectOutputStream ObjectInputStream

//ObjectInputStream

//ObjectOutputStream

FileOutputStream fos = new FileOutputStream(new File("/sdcard/studnet.obj"));

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(new Student());

10. 怎么让在启动一个Activity是就启动一个service？

在activity的onCreate()方法里面 startService();

11. 同一个程序，但不同的Activity是否可以放在不同的Task任务栈中？

比方说在激活一个新的activity时候, 给intent设置flag

Intent的flag添加FLAG\_ACTIVITY\_NEW\_TASK

这个被激活的activity就会在新的task栈里面…

12. Activity怎么和service绑定，怎么在activity中启动自己对应的service？

bindService(). 让activity能够访问到 service里面的方法

构建一个intent对象,

Intent service = new Intent(this,MyService.class);

通过bindService的方法去启动一个服务,

ServiceConnection 对象(重写onServiceConnected和OnServiceDisconnected方法) 和BIND\_AUTO\_CREATE.

private class myconn implements ServiceConnection

{

public void onServiceConnected(ComponentName name, IBinder service) {

// TODO Auto-generated method stub

//可以通过IBinder的对象 去使用service里面的方法

}

public void onServiceDisconnected(ComponentName name) {

// TODO Auto-generated method stub

}

}

13. 14 .什么是Service以及描述下它的生命周期。Service有哪些启动方法，有什么区别，怎样停用Service？

在Service的生命周期中，被回调的方法比Activity少一些，只有onCreate, onStart, onDestroy,

onBind和onUnbind。

通常有两种方式启动一个Service,他们对Service生命周期的影响是不一样的。

1 通过startService

Service会经历 onCreate 到onStart，然后处于运行状态，stopService的时候调用onDestroy方法。

如果是调用者自己直接退出而没有调用stopService的话，Service会一直在后台运行。

2 通过bindService

Service会运行onCreate，然后是调用onBind， 这个时候调用者和Service绑定在一起。调用者退出了，Srevice就会调用onUnbind->onDestroyed方法。

所谓绑定在一起就共存亡了。调用者也可以通过调用unbindService方法来停止服务，这时候Srevice就会调用onUnbind->onDestroyed方法。

需要注意的是如果这几个方法交织在一起的话，会出现什么情况呢？

一个原则是Service的onCreate的方法只会被调用一次，就是你无论多少次的startService又bindService，Service只被创建一次。

如果先是bind了，那么start的时候就直接运行Service的onStart方法，如果先是start，那么bind的时候就直接运行onBind方法。

如果service运行期间调用了bindService，这时候再调用stopService的话，service是不会调用onDestroy方法的，service就stop不掉了，只能先UnbindService, 再StopService。

如果一个service通过startService 被start之后，多次调用startService 的话，service会多次调用onStart方法。多次调用stopService的话，service只会调用一次onDestroyed方法。

如果一个service通过bindService被start之后，多次调用bindService的话，service只会调用一次onBind方法。

多次调用unbindService的话会抛出异常。

15. 不用service，B页面为音乐播放，从A跳转到B，再返回，如何使音乐继续播放？

这个问题问的很山寨.默认不做任何处理,B里面的音乐都能播放.

遇到问题, 可以随机应变,灵活发挥,多考虑些细节,比如说这个题就可以这样说,说说你对startActivityForResult的理解()

A开启B的时候,用startActivityForResult()方法, B返回的时候把播放的状态信息返回给A ,A继续播放音乐.

16. 什么是IntentService？有何优点？

普通的service ,默认运行在ui main 主线程

Sdk给我们提供的方便的,带有异步处理的service类,

OnHandleIntent() 处理耗时的操作

17. 什么时候使用Service？

拥有service的进程具有较高的优先级

官方文档告诉我们，Android系统会尽量保持拥有service的进程运行，只要在该service已经被启动(start)或者客户端连接(bindService)到它。当内存不足时，需要保持，拥有service的进程具有较高的优先级。

1． 如果service正在调用onCreate, onStartCommand或者onDestory方法，那么用于当前service的进程相当于前台进程以避免被killed。

2． 如果当前service已经被启动(start)，拥有它的进程则比那些用户可见的进程优先级低一些，但是比那些不可见的进程更重要，这就意味着service一般不会被killed.

3． 如果客户端已经连接到service (bindService),那么拥有Service的进程则拥有最高的优先级，可以认为service是可见的。

4． 如果service可以使用startForeground(int, Notification)方法来将service设置为前台状态，那么系统就认为是对用户可见的，并不会在内存不足时killed。

如果有其他的应用组件作为Service,Activity等运行在相同的进程中，那么将会增加该进程的重要性。

1.Service的特点可以让他在后台一直运行,可以在service里面创建线程去完成耗时的操作.

2.Broadcast receiver捕获到一个事件之后,可以起一个service来完成一个耗时的操作.

3.远程的service如果被启动起来,可以被多次bind, 但不会重新create. 索爱手机X10i的人脸识别的service可以被图库使用,可以被摄像机,照相机等程序使用.

18. 请描述一下Intent 和 Intent Filter。

Android 中通过 Intent 对象来表示一条消息，一个 Intent 对象不仅包含有这个消息的目的地，还可以包含消息的内容，这好比一封 Email，其中不仅应该包含收件地址，还可以包含具体的内容。对于一个 Intent 对象，消息“目的地”是必须的，而内容则是可选项。

通过Intent 可以实现各种系统组件的调用与激活.

Intent filter: 可以理解为邮局或者是一个信笺的分拣系统…

这个分拣系统通过3个参数来识别

Action: 动作

Data: 数据uri

Category : 而外的附加信息

Action 匹配

Action 是一个用户定义的字符串，用于描述一个 Android 应用程序组件，一个 Intent Filter 可以包含多个 Action。在 AndroidManifest.xml 的 Activity 定义时可以在其 <intent-filter >节点指定一个 Action 列表用于标示 Activity 所能接受的“动作”，例如：

<intent-filter >

<action android:name="android.intent.action.MAIN" />

<action android:name="cn.itcast.action" />

……

</intent-filter>

如果我们在启动一个 Activity 时使用这样的 Intent 对象：

Intent intent =new Intent();

intent.setAction("cn.itcast");

那么所有的 Action 列表中包含了“cn.itcast”的 Activity 都将会匹配成功。

Android 预定义了一系列的 Action 分别表示特定的系统动作。这些 Action 通过常量的方式定义在 android.content. Intent中，以“ACTION\_”开头。我们可以在 Android 提供的文档中找到它们的详细说明。

URI 数据匹配

一个 Intent 可以通过 URI 携带外部数据给目标组件。在 <intent-filter >节点中，通过 <data/>节点匹配外部数据。

mimeType 属性指定携带外部数据的数据类型，scheme 指定协议，host、port、path 指定数据的位置、端口、和路径。如下：

<data android:mimeType="mimeType" android:scheme="scheme"

android:host="host" android:port="port" android:path="path"/>

电话的uri tel://12345

http://www.baidu.com

自己定义的uri itcast://cn.itcast/person/10

如果在 Intent Filter 中指定了这些属性，那么只有所有的属性都匹配成功时 URI 数据匹配才会成功。

Category 类别匹配

<intent-filter >节点中可以为组件定义一个 Category 类别列表，当 Intent 中包含这个列表的所有项目时 Category 类别匹配才会成功。

默认是DEFAULT

19. Intent传递数据时，可以传递哪些类型数据？

1.一般的基本数据类型 Intent .putextra() intent.getextra();

2.数据的uri, intent.setData() intent.getData();

20. 说说Activity，Intent，Service是什么关系 。

麦当劳和麦当娜的关系是什么关系?

这种问题,就讲下activity,讲一下service,说一下 通过intent去激活组件,传递数据.

说自己项目中有这样一个网络更新的功能,显示界面就用的activity, 后台有个service每隔半小时都去访问下服务器获取更新的数据…开启服务用的是intent来开启

21. 请描述一下Broadcast Receiver。

用于接收系统的广播通知, 系统会有很多sd卡挂载,手机重启,广播通知,低电量,来电,来短信等….

手机卫士中自定义一个broadcast receiver来获取短信到来的广播, 根据黑名单来判断是否拦截该短信.

画画板生成图片后,发送一个sd挂载的通知,通知系统的gallery去获取到新的图片.

22. 在manifest和代码中如何注册和使 用 broadcast receiver 。

设置广播接收者的优先级,设置广播接受者的action名字 等…

详细见工程代码.

23. 请介绍下ContentProvider是如何实现数据共享的。

把自己的数据通过uri的形式共享出去

需要去实现一个类去继承ContentProvider

public class PersonContentProvider extends ContentProvider{

public boolean onCreate(){

//..

}

query(Uri, String[], String, String[], String)

insert(Uri, ContentValues)

update(Uri, ContentValues, String, String[])

delete(Uri, String, String[])

}

24. 请介绍下Android的数据存储方式。

文件 访问权限.

数据库 sqlite

SharedPreference

网络

25. 为什么要用ContentProvider？它和sql的实现上有什么差别？

屏蔽数据存储的细节,对用户透明,用户只需要关心操作数据的uri就可以了

不同app之间共享,操作数据

Sql也有增删改查的方法.

但是contentprovider 还可以去增删改查本地文件.

26. 请介绍下Android中常用的五种布局。

FrameLayout（框架布局），LinearLayout （线性布局），AbsoluteLayout（绝对布局），RelativeLayout（相对布局），TableLayout（表格布局）

FrameLayout

从屏幕的左上角开始布局,叠加显示, 实际应用 播放器的暂停按钮.

LinearLayout

线性布局，这个东西，从外框上可以理解为一个div，他首先是一个一个从上往下罗列在屏幕上。每一个LinearLayout里面又可分为垂直布局

（android:orientation="vertical"）和水平布局（android:orientation="horizontal"

）。当垂直布局时，每一行就只有一个元素，多个元素依次垂直往下；水平布局时，只有一行，每一个元素依次向右排列。

AbsoluteLayout

绝对布局犹如div指定了absolute属性，用X,Y坐标来指定元素的位置android:layout\_x="20px"

android:layout\_y="12px"

指定平板机型的游戏开发中经常用到绝对布局

指定机型的平板游戏开发.

RelativeLayout

相对布局可以理解为某一个元素为参照物，来定位的布局方式。主要属性有：

相对于某一个元素

android:layout\_below="@id/aaa" 该元素在 id为aaa的下面

android:layout\_toLeftOf="@id/bbb" 改元素的左边是bbb

相对于父元素的地方

android:layout\_alignParentLeft="true" 在父元素左对齐

android:layout\_alignParentRight="true" 在父元素右对齐

TableLayout

表格布局类似Html里面的Table。每一个TableLayout里面有表格行TableRow，TableRow里面可以具体定义每一个元素，设定他的对齐方式 android:gravity="" 。

每一个布局都有自己适合的方式，另外，这五个布局元素可以相互嵌套应用，做出美观的界面。

webview

27. 谈谈UI中， Padding和Margin有什么区别？

Padding 文字对边框, margin是控件对父窗体.

28. widget相对位置的完成在activity的哪个生命周期阶段实现。

这个题没看懂… widget可以理解成桌面小控件, 也可以理解成 某个button, imageview这样的控件…

29. 请解释下在单线程模型中Message、Handler、Message Queue、Looper之间的关系。

30. AIDL的全称是什么？如何工作？

Android interface definition language (android接口定义语言) , 用来跨进程的访问方法,

访问远程的服务的方法. 如何工作 day7 queryStudent .

手机卫士 Itelephony 接口挂断电话.

31. 请解释下Android程序运行时权限与文件系统权限的区别。

Android程序执行需要读取到安全敏感项必需在androidmanifest.xml中声明相关权限请求, 打电话,访问网络,获取坐标,读写sd卡,读写联系人等..安装的时候会提示用户…

文件系统的权限是linux权限. 比如说sharedpreference里面的Context.Mode.private Context.Mode.world\_read\_able Context.Mode\_world\_writeable

32. 系统上安装了多种浏览器，能否指定某浏览器访问指定页面？

找到对应的浏览器的意图,传递数据URI , 激活这个意图

33. 对android主线程的运用和理解。

主ui线程不能执行耗时的操作,

34. 对android虚拟机的理解，包括内存管理机制垃圾回收机制。

虚拟机很小,空间很小,谈谈移动设备的虚拟机的大小限制 16M , 谈谈加载图片的时候怎么处理大图片的,

垃圾回收,没有引用的对象,在某个时刻会被系统gc掉.

35. Framework工作方式及原理，Activity是如何生成一个view的，机制是什么。

可以讲下activity的源码,比如说 每个activity里面都有window.callback和keyevent.callback,一些回调的接口或者函数吧. 框架把activity创建出来就会调用里面的这些回调方法,会调用activity生命周期相关的方法.

**Android面试题汇总A**

**双缓冲技术原理以及优缺点**：

创建一幅后台图像，将每一帧画入图像，然后调用drawImage()方法将整个后台图像一次画到屏幕上去。

优点：双缓冲技术的优点在于大部分绘制是离屏的。

将离屏图像一次绘至屏幕上，比直接在屏幕上绘制要有效得多。

双缓冲技术可以使动画平滑。

缺点：要分配一个后台图像的缓冲，如果图像相当大，这将占用很大一块内存。

**AsyncTask:**

android 提供了一个工具类：AsyncTask ，它使创建需要与用户界面交互的长时间运行的任务变得更简单。

**Socket:**

客户端编程步骤：

1、 创建客户端套接字(指定服务器端IP地址与端口号)

2、 连接(Android 创建Socket时会自动连接)

3、 与服务器端进行通信

4、 关闭套接字

服务器端:

1.创建一个ServerSocket，用于监听客户端Socket的连接请求

2.采用循环不断接受来自客户端的请求

3.每当接受到客户端Socket的请求，服务器端也对应产生一个Socket

**Activity状态**

如图3-7所示，当Activity被创建或销毁时，它们进入或退出Activity栈。当它们做这些动作时，它们就会在四种可能的状态间迁移：

? Active

当Activity在栈的顶端时，它是可见的，有焦点的前台Activity，用来响应用户的输入。Android会不惜一切代价来尝试保证它的活跃性，需要的话它会杀死栈中更靠下的Activity来保证Active Activity需要的资源。当另一个Activity变成Active状态时，这个就会变成paused。

? Paused

在一些情况下，你的Activity可见但不拥有焦点；在这个时刻，它就是暂停的。当最前面的Activity是全透明或非全屏的Activity时，下面的Activity就会到达这个状态。当暂停时，这个Activity还是被看作是active的，但不接受用户的输入事件。在极端的情况下，Android会杀死一个paused的Activity来恢复资源给active Activity。当一个Activity完全不可见时，它就变成stopped。

? Stopped

当一个Activity不可见，它就“停止”了。这个Activity仍然留在内存里来保存所有的状态和成员信息；但是，在什么地方当系统需要内存时，它就是“罪犯”拉出去枪毙了。当一个Activity停止时，保存数据和当前UI状态是很重要的。一旦Activity退出或关闭，它就变成inactive。

? Inactive

当一个曾经被启动过的Activity被杀死时，它就变成inactive。Inactive Activity会从Activity栈中移除，当它重新显示和使用时需要再次启动。

**Android实现下载图片并保存到SD卡中**

思路如下：

权限

首先判断SD卡是否插入-->

public String getSDPath(){

File SDdir=null;

boolean sdCardExist=

Environment.getExternalStorageState().equals(android.os.Environment.MEDIA\_MOUNTED);

if(sdCardExist){

SDdir=Environment.getExternalStorageDirectory();

}

if(SDdir!=null){

return SDdir.toString();

}

else{

return null;

}

}

然后创建文件夹-->

public void createSDCardDir(){

if(getSDPath()==null){

Toast.makeText(PicSharesActivity.this, "未找到SD卡", 1000).show();

}else{

if(Environment.MEDIA\_MOUNTED.equals(Environment.getExternalStorageState())){

// 创建一个文件夹对象，赋值为外部存储器的目录

File sdcardDir =Environment.getExternalStorageDirectory();

//得到一个路径，内容是sdcard的文件夹路径和名字

newPath=sdcardDir.getPath()+"/\*\*\*app/tempImages/";//newPath在程序中要声明

File path1 = new File(newPath);

if (!path1.exists()) {

//若不存在，创建目录，可以在应用启动的时候创建

path1.mkdirs();

System.out.println("paht ok,path:"+newPath);

}

}

else{

System.out.println("false");

}

}

}

创建好文件夹之后就可以保存图片了-->

public void saveMyBitmap(String bitName,int percent) throws IOException {

Bitmap bmp = drawable2Bitmap(picView.getDrawable());//这里的drawable2Bitmap方法是我把ImageView中 的drawable转化成bitmap，当然实验的时候可以自己创建bitmap

File f = new File(newPath + bitName + ".jpg");

f.createNewFile();

FileOutputStream fOut = null;

try {

fOut = new FileOutputStream(f);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

bmp.compress(Bitmap.CompressFormat.JPEG, percent, fOut);

try {

fOut.flush();

} catch (IOException e) {

e.printStackTrace();

}

try {

fOut.close();

} catch (IOException e) {

e.printStackTrace();

}

}

//附加drawable2Bitmap方法

static Bitmap drawable2Bitmap(Drawable d){

int width=d.getIntrinsicWidth();

int height=d.getIntrinsicHeight();

Bitmap.Config config=d.getOpacity()!=PixelFormat.OPAQUE ? Bitmap.Config.ARGB\_8888: Bitmap.Config.RGB\_565;

Bitmap bitmap=Bitmap.createBitmap(width,height,config);

Canvas canvas=new Canvas(bitmap);

d.setBounds(0, 0, width, height);

d.draw(canvas);

return bitmap;

}

**Android面试题汇总B**

　　1. 下列哪些语句关于内存回收的说明是正确的? (b ) A、 程序员必须创建一个线程来释放内存

　　B、 内存回收程序负责释放无用内存

　　C、 内存回收程序允许程序员直接释放内存

　　D、 内存回收程序可以在指定的时间释放内存对象

　　2. 下面异常是属于Runtime Exception 的是(abcd)(多选) A、ArithmeticException

　　B、IllegalArgumentException

　　C、NullPointerException

　　D、BufferUnderflowException

　　3. Math.round(11.5)等于多少(). Math.round(-11.5)等于多少(c). c A、11 ,-11 B、11 ,-12 C、12 ,-11 D、12 ,-12

　　4. 下列程序段的输出结果是：(b )

　　void complicatedexpression\_r(){

　　int x=20, y=30;

　　boolean b;

　　b=x>50&&y>60||x>50&&y<-60||x<-50&&y>60||x<-50&&y<-60;

　　System.out.println(b);

　　}

　　A、true B、false C、1 D、011.activity

　　5. 对一些资源以及状态的操作保存，最好是保存在生命周期的哪个函数中进行(d) A、onPause() B、onCreate() C、 onResume() D、onStart()

　　6. Intent传递数据时，下列的数据类型哪些可以被传递(abcd)(多选) A、Serializable B、charsequence C、Parcelable D、Bundle

　　7. android 中下列属于Intent的作用的是(c) A、实现应用程序间的数据共享

　　B、是一段长的生命周期，没有用户界面的程序，可以保持应用在后台运行，而不会因为切换页面而消失

　　C、可以实现界面间的切换，可以包含动作和动作数据，连接四大组件的纽带

　　D、处理一个应用程序整体性的工作

　　8. 下列属于SAX解析xml文件的优点的是(b) A、将整个文档树在内存中，便于操作，支持删除，修改，重新排列等多种功能

　　B、不用事先调入整个文档，占用资源少

　　C、整个文档调入内存，浪费时间和空间

　　D、不是长久驻留在内存，数据不是持久的，事件过后，若没有保存数据，数据就会

　　消失

　　9. 下面的对自定style的方式正确的是

A、 <resources>

<style name="myStyle">

<itemname="android:layout\_width">fill\_parent</item>

</style>

</resources>

B、 <style name="myStyle">

<itemname="android:layout\_width">fill\_parent</item>

</style>

C、 <resources>

<itemname="android:layout\_width">fill\_parent</item>

</resources>

D、 <resources>

<stylename="android:layout\_width">fill\_parent</style>

</resources>

　　10. 在android中使用Menu时可能需要重写的方法有(ac)。(多选) A、onCreateOptionsMenu()

　　B、onCreateMenu()

　　C、onOptionsItemSelected()

　　D、onItemSelected()

　　11. 在SQL Server Management Studio 中运行下列T-SQL语句，其输出值(c)。 SELECT @@IDENTITY

　　A、 可能为0.1

　　B、 可能为3

　　C、 不可能为-100

　　D、 肯定为0

　　12. 在SQL Server 2005中运行如下T-SQL语句，假定SALES表中有多行数据，执行查询之 后的结果是(d)。 BEGIN TRANSACTION A

　　Update SALES Set qty=30 WHERE qty<30

　　BEGIN TRANSACTION B

　　Update SALES Set qty=40 WHEREqty<40

　　Update SALES Set qty=50 WHEREqty<50

　　Update SALES Set qty=60 WHEREqty<60

　　COMMIT　TRANSACTION B

　　COMMIT TRANSACTION A

　　A、SALES表中qty列最小值大于等于30

　　B、SALES表中qty列最小值大于等于40

　　C、SALES表中qty列的数据全部为50

　　D、SALES表中qty列最小值大于等于60

　　13. 在android中使用SQLiteOpenHelper这个辅助类时，可以生成一个数据库，并可以对数据库版本进行管理的方法可以是(ab) A、getWriteableDatabase()

　　B、getReadableDatabase()

　　C、getDatabase()

　　D、getAbleDatabase()

　　14. android 关于service生命周期的onCreate()和onStart()说法正确的是(ad)(多选题) A、当第一次启动的时候先后调用onCreate()和onStart()方法

　　B、当第一次启动的时候只会调用onCreate()方法

　　C、如果service已经启动，将先后调用onCreate()和onStart()方法

　　D、如果service已经启动，只会执行onStart()方法，不在执行onCreate()方法

　　15. 下面是属于GLSurFaceView特性的是(abc)(多选) A、管理一个surface，这个surface就是一块特殊的内存，能直接排版到android的视图

　　view上。

　　B、管理一个EGL display，它能让opengl把内容渲染到上述的surface上。

　　C、让渲染器在独立的线程里运作，和UI线程分离。

　　D、可以直接从内存或者DMA等硬件接口取得图像数据

　　16. 下面在AndroidManifest.xml文件中注册BroadcastReceiver方式正确的

A、<receiver android:name="NewBroad">

<intent-filter>

<action

android:name="android.provider.action.NewBroad"/>

<action>

</intent-filter>

</receiver>

B、<receiver android:name="NewBroad">

<intent-filter>

android:name="android.provider.action.NewBroad"/>

</intent-filter>

</receiver>

C、<receiver android:name="NewBroad">

<action

android:name="android.provider.action.NewBroad"/>

<action>

</receiver>

D、<intent-filter>

<receiver android:name="NewBroad">

<action>

android:name="android.provider.action.NewBroad"/>

<action>

</receiver>

</intent-filter>

　　17. 关于ContenValues类说法正确的是(a) A、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

　　名是String类型，而值都是基本类型

　　B、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

　　名是任意类型，而值都是基本类型

　　C、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

　　名，可以为空，而值都是String类型

　　D、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中

　　的名是String类型，而值也是String类型

　　18. 我们都知道Hanlder是线程与Activity通信的桥梁,如果线程处理不当，你的机器就会变得越慢，那么线程销毁的方法是(a) A、onDestroy()

　　B、onClear()

　　C、onFinish()

　　D、onStop()

　　19. 下面退出Activity错误的方法是(c) A、finish()

　　B、抛异常强制退出

　　C、System.exit()

　　D、onStop()

　　20. 下面属于android的动画分类的有(ab)(多项) A、Tween B、Frame C、Draw D、Animation

　　21. 下面关于Android dvm的进程和Linux的进程,应用程序的进程说法正确的是(d) A、DVM指dalivk的虚拟机.每一个Android应用程序都在它自己的进程中运行,不一定拥有一个独立的Dalvik虚拟机实例.而每一个DVM都是在Linux中的一个进程,所以说可以认为是同一个概念.

　　B、DVM指dalivk的虚拟机.每一个Android应用程序都在它自己的进程中运行,不一定拥有一个独立的Dalvik虚拟机实例.而每一个DVM不一定都是在Linux中的一个进程,所以说不是一个概念.

　　C、DVM指dalivk的虚拟机.每一个Android应用程序都在它自己的进程中运行,都拥有一个独立的Dalvik虚拟机实例.而每一个DVM不一定都是在Linux中的一个进程,所以说不是一个概念.

　　D、DVM指dalivk的虚拟机.每一个Android应用程序都在它自己的进程中运行,都拥有一个独立的 Dalvik虚拟机实例.而每一个DVM都是在Linux中的一个进程,所以说可以认为是同一个概念.

　　22. Android项目工程下面的assets目录的作用是什么bA、放置应用到的图片资源。

　　B、主要放置多媒体等数据文件

　　C、放置字符串，颜色，数组等常量数据

　　D、放置一些与UI相应的布局文件，都是xml文件

　　23. 关于res/raw目录说法正确的是(a)A、 这里的文件是原封不动的存储到设备上不会转换为二进制的格式

　　B、这里的文件是原封不动的存储到设备上会转换为二进制的格式

　　C、 这里的文件最终以二进制的格式存储到指定的包中

　　D、这里的文件最终不会以二进制的格式存储到指定的包中

　　24. 下列对android NDK的理解正确的是(abcd )A、 NDK是一系列工具的集合

　　B、 NDK 提供了一份稳定、功能有限的 API 头文件声明。

　　C、 使 “Java+C” 的开发方式终于转正，成为官方支持的开发方式

　　D、 NDK 将是 Android 平台支持 C 开发的开端

　　二.文件存储方式

　　三.SQLite数据库方式

　　四.内容提供器(Content provider)方式

　　二、Android面试填空题

　　25. android中常用的四个布局是framlayout，linenarlayout，relativelayout和tablelayout。26. android 的四大组件是activiey，service，broadcast和contentprovide。27. java.io包中的objectinputstream和objectoutputstream类主要用于对对象(Object)的读写。28. android 中service的实现方法是：startservice和bindservice。29. activity一般会重载7个方法用来维护其生命周期，除了onCreate(),onStart(),onDestory() 外还有onrestart,onresume,onpause,onstop。30. android的数据存储的方式sharedpreference,文件,SQlite,contentprovider,网络。31. 当启动一个Activity并且新的Activity执行完后需要返回到启动它的Activity来执行 的回调函数是startActivityResult()。32. 请使用命令行的方式创建一个名字为myAvd,sdk版本为2.2,sd卡是在d盘的根目录下,名字为scard.img， 并指定屏幕大小HVGA.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。33. 程序运行的结果是：\_\_\_\_\_good and gbc\_\_\_\_\_\_\_\_\_\_。 public classExample{

　　String str=new String("good");

　　char[]ch={'a','b','c'};

　　public static void main(String args[]){

　　Example ex=new Example();

　　ex.change(ex.str,ex.ch);

　　System.out.print(ex.str+" and ");

　　Sytem.out.print(ex.ch);

　　}

　　public void change(String str,char ch[]){

　　str="test ok";

　　ch[0]='g';

　　}

　　}

　　34. 在android中，请简述jni的调用过程。(8分)1)安装和下载Cygwin，下载 Android NDK

　　2)在ndk项目中JNI接口的设计

　　3)使用C/C++实现本地方法

　　4)JNI生成动态链接库.so文件

　　5)将动态链接库复制到java工程，在java工程中调用，运行java工程即可

　　35. 简述Android应用程序结构是哪些?(7分)Android应用程序结构是：

　　Linux Kernel(Linux内核)、Libraries(系统运行库或者是c/c++核心库)、Application

　　Framework(开发框架包)、Applications (核心应用程序)

　　36. 请继承SQLiteOpenHelper实现：(10分) 1).创建一个版本为1的“diaryOpenHelper.db”的数据库，

　　2).同时创建一个 “diary” 表(包含一个\_id主键并自增长，topic字符型100

　　长度， content字符型1000长度)

　　3).在数据库版本变化时请删除diary表，并重新创建出diary表。

　　publicclass DBHelper extends SQLiteOpenHelper{

　　public final static String DATABASENAME ="diaryOpenHelper.db";

　　public final static int DATABASEVERSION =1;

　　//创建数据库

　　public DBHelper(Context context,Stringname,CursorFactory factory,int version)

　　{

　　super(context, name, factory,version);

　　}

　　//创建表等机构性文件

　　public void onCreate(SQLiteDatabase db)

　　{

　　String sql ="create tablediary"+

　　"("+

　　"\_idinteger primary key autoincrement,"+

　　"topicvarchar(100),"+

　　"contentvarchar(1000)"+

　　")";

　　db.execSQL(sql);

　　}

　　//若数据库版本有更新，则调用此方法

　　public void onUpgrade(SQLiteDatabasedb,int oldVersion,int newVersion)

　　{

　　String sql = "drop table ifexists diary";

　　db.execSQL(sql);

　　this.onCreate(db);

　　}

　　}

　　37. 页面上现有ProgressBar控件progressBar，请用书写线程以10秒的的时间完成其进度显示工作。(10分)答案

　　publicclass ProgressBarStu extends Activity {

　　private ProgressBar progressBar = null;

　　protected void onCreate(BundlesavedInstanceState) {

　　super.onCreate(savedInstanceState);

　　setContentView(R.layout.progressbar);

　　//从这到下是关键

　　progressBar = (ProgressBar)findViewById(R.id.progressBar);

　　Thread thread = new Thread(newRunnable() {

　　@Override

　　public void run() {

　　int progressBarMax =progressBar.getMax();

　　try {

　　while(progressBarMax!=progressBar.getProgress())

　　{

　　intstepProgress = progressBarMax/10;

　　intcurrentprogress = progressBar.getProgress();

　　progressBar.setProgress(currentprogress+stepProgress);

　　Thread.sleep(1000);

　　}

　　} catch(InterruptedException e) {

　　// TODO Auto-generatedcatch block

　　e.printStackTrace();

　　}

　　}

　　});

　　thread.start();

　　//关键结束

　　}

　　}

　　38. 请描述下Activity的生命周期。 必调用的三个方法：onCreate() --> onStart() --> onResume()，用AAA表示

　　(1)父Activity启动子Activity，子Actvity退出，父Activity调用顺序如下

　　AAA --> onFreeze() --> onPause() --> onStop() --> onRestart()--> onStart(),onResume() …

　　(2)用户点击Home，Actvity调用顺序如下

　　AAA --> onFreeze() --> onPause() --> onStop() -- Maybe -->onDestroy() – Maybe

　　(3)调用finish()， Activity调用顺序如下

　　AAA --> onPause() --> onStop() --> onDestroy()

　　(4)在Activity上显示dialog，Activity调用顺序如下

　　AAA

　　(5)在父Activity上显示透明的或非全屏的activity，Activity调用顺序如下

　　AAA --> onFreeze() --> onPause()

　　(6)设备进入睡眠状态，Activity调用顺序如下

　　AAA --> onFreeze() --> onPause()

　　39. 如果后台的Activity由于某原因被系统回收了，如何在被系统回收之前保存当前状态? 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。

　　40. 如何将一个Activity设置成窗口的样式。 在AndroidManifest.xml 中定义Activity的地方一句话android:theme="@android:style/Theme.Dialog"或android:theme="@android:style/Theme.Translucent"就变成半透明的

　　41. 如何退出Activity?如何安全退出已调用多个Activity的Application?对于单一Activity的应用来说，退出很简单，直接finish()即可。

　　当然，也可以用killProcess()和System.exit()这样的方法。

　　但是，对于多Activity的应用来说，在打开多个Activity后，如果想在最后打开的Activity直接退出，上边的方法都是没有用的，因为上边的方法都是结束一个Activity而已。

　　当然，网上也有人说可以。

　　就好像有人问，在应用里如何捕获Home键，有人就会说用keyCode比较KEYCODE\_HOME即可，而事实上如果不修改framework，根本不可能做到这一点一样。

　　所以，最好还是自己亲自试一下。

　　那么，有没有办法直接退出整个应用呢?

　　在2.1之前，可以使用ActivityManager的restartPackage方法。

　　它可以直接结束整个应用。在使用时需要权限android.permission.RESTART\_PACKAGES。

　　注意不要被它的名字迷惑。

　　可是，在2.2，这个方法失效了。

　　在2.2添加了一个新的方法，killBackgroundProcesses()，需要权限android.permission.KILL\_BACKGROUND\_PROCESSES。

　　可惜的是，它和2.2的restartPackage一样，根本起不到应有的效果。

　　另外还有一个方法，就是系统自带的应用程序管理里，强制结束程序的方法，forceStopPackage()。

　　它需要权限android.permission.FORCE\_STOP\_PACKAGES。

　　并且需要添加android:sharedUserId="android.uid.system"属性

　　同样可惜的是，该方法是非公开的，他只能运行在系统进程，第三方程序无法调用。

　　因为需要在Android.mk中添加LOCAL\_CERTIFICATE := platform。

　　而Android.mk是用于在Android源码下编译程序用的。

　　从以上可以看出，在2.2，没有办法直接结束一个应用，而只能用自己的办法间接办到。

　　现提供几个方法，供参考：

　　1、抛异常强制退出：

　　该方法通过抛异常，使程序ForceClose。

　　验证可以，但是，需要解决的问题是，如何使程序结束掉，而不弹出Force Close的窗口。

　　2、记录打开的Activity：

　　每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可。

　　3、发送特定广播：

　　在需要结束应用时，发送一个特定的广播，每个Activity收到广播后，关闭即可。

　　4、递归退出

　　在打开新的Activity时使用startActivityForResult，然后自己加标志，在onActivityResult中处理，递归关闭。

　　除了第一个，都是想办法把每一个Activity都结束掉，间接达到目的。

　　但是这样做同样不完美。

　　你会发现，如果自己的应用程序对每一个Activity都设置了nosensor，在两个Activity结束的间隙，sensor可能有效了。

　　但至少，我们的目的达到了，而且没有影响用户使用。

　　为了编程方便，最好定义一个Activity基类，处理这些共通问题。

　　42. 请介绍下Android中常用的五种布局。FrameLayout(框架布局)，LinearLayout (线性布局)，AbsoluteLayout(绝对布局)，RelativeLayout(相对布局)，TableLayout(表格布局)

　　43. 请介绍下Android的数据存储方式。一.SharedPreferences方式

　　五. 网络存储方式

　　44. 请介绍下ContentProvider是如何实现数据共享的。创建一个属于你自己的Content provider或者将你的数据添加到一个已经存在的Contentprovider中，前提是有相同数据类型并且有写入Content provider的权限。

　　45. 如何启用Service，如何停用Service。Android中的service类似于windows中的service，service一般没有用户操作界面，它运行于系统中不容易被用户发觉，

　　可以使用它开发如监控之类的程序。

　　一。步骤

　　第一步：继承Service类

　　public class SMSService extends Service { }

　　第二步：在AndroidManifest.xml文件中的节点里对服务进行配置:

　　二。Context.startService()和Context.bindService

　　服务不能自己运行，需要通过调用Context.startService()或Context.bindService()方法启动服务。这两个方法都可

　　以启动Service，但是它们的使用场合有所不同。

　　1.使用startService()方法启用服务，调用者与服务之间没有关连，即使调用者退出了，服务仍然运行。

　　使用bindService()方法启用服务，调用者与服务绑定在了一起，调用者一旦退出，服务也就终止。

　　2.采用Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，

　　接着调用onStart()方法。如果调用startService()方法前服务已经被创建，多次调用startService()方法并

　　不会导致多次创建服务，但会导致多次调用onStart()方法。

　　采用startService()方法启动的服务，只能调用Context.stopService()方法结束服务，服务结束时会调用

　　onDestroy()方法。

　　3.采用Context.bindService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，

　　接着调用onBind()方法。这个时候调用者和服务绑定在一起，调用者退出了，系统就会先调用服务的onUnbind()方法，

　　。接着调用onDestroy()方法。如果调用bindService()方法前服务已经被绑定，多次调用bindService()方法并不会

　　导致多次创建服务及绑定(也就是说onCreate()和onBind()方法并不会被多次调用)。如果调用者希望与正在绑定的服务

　　解除绑定，可以调用unbindService()方法，调用该方法也会导致系统调用服务的onUnbind()-->onDestroy()方法。

　　三。Service的生命周期

　　1.Service常用生命周期回调方法如下：

　　onCreate() 该方法在服务被创建时调用，该方法只会被调用一次，无论调用多少次startService()或bindService()方法，

　　服务也只被创建一次。 onDestroy()该方法在服务被终止时调用。

　　2. Context.startService()启动Service有关的生命周期方法

　　onStart() 只有采用Context.startService()方法启动服务时才会回调该方法。该方法在服务开始运行时被调用。

　　多次调用startService()方法尽管不会多次创建服务，但onStart()方法会被多次调用。

　　3. Context.bindService()启动Service有关的生命周期方法

　　onBind()只有采用Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务绑定时被调用，

　　当调用者与服务已经绑定，多次调用Context.bindService()方法并不会导致该方法被多次调用。

　　onUnbind()只有采用Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务解除绑定时被调用。

　　备注：

　　1. 采用startService()启动服务

　　Intent intent =new Intent(DemoActivity.this, DemoService.class);

　　startService(intent);

　　2.Context.bindService()启动

　　Intent intent =new Intent(DemoActivity.this, DemoService.class);

　　bindService(intent, conn, Context.BIND\_AUTO\_CREATE);

　　//unbindService(conn);//解除绑定

　　46. 注册广播有几种方式，这些方式有何优缺点?请谈谈Android引入广播机制的用意。 Android广播机制(两种注册方法)

　　在android下，要想接受广播信息，那么这个广播接收器就得我们自己来实现了，我们可以继承BroadcastReceiver，就可以有一个广播接受器了。有个接受器还不够，我们还得重写BroadcastReceiver里面的onReceiver方法，当来广播的时候我们要干什么，这就要我们自己来实现，不过我们可以搞一个信息防火墙。具体的代码：

　　public class SmsBroadCastReceiverextends BroadcastReceiver

　　{

　　@Override

　　public void onReceive(Context context, Intent intent)

　　{

　　Bundle bundle = intent.getExtras();

　　Object[] object = (Object[])bundle.get("pdus");

　　SmsMessage sms[]=new SmsMessage[object.length];

　　for(int i=0;i

　　{

　　sms[0] =SmsMessage.createFromPdu((byte[])object);

　　Toast.makeText(context, "来自"+sms.getDisplayOriginatingAddress()+"的消息是："+sms.getDisplayMessageBody(),Toast.LENGTH\_SHORT).show();

　　}

　　//终止广播，在这里我们可以稍微处理，根据用户输入的号码可以实现短信防火墙。

　　abortBroadcast();

　　}

　　}

　　当实现了广播接收器，还要设置广播接收器接收广播信息的类型，这里是信息：android.provider.Telephony.SMS\_RECEIVED

　　我们就可以把广播接收器注册到系统里面，可以让系统知道我们有个广播接收器。这里有两种，一种是代码动态注册：

　　//生成广播处理

　　smsBroadCastReceiver = newSmsBroadCastReceiver();

　　//实例化过滤器并设置要过滤的广播

　　IntentFilter intentFilter = newIntentFilter("android.provider.Telephony.SMS\_RECEIVED");

　　//注册广播

　　BroadCastReceiverActivity.this.registerReceiver(smsBroadCastReceiver,intentFilter);

　　一种是在AndroidManifest.xml中配置广播

　　package="spl.broadCastReceiver"

　　android:versionCode="1"

　　android:versionName="1.0">

　　android:label="@string/app\_name">

　　两种注册类型的区别是：

　　1)第一种不是常驻型广播，也就是说广播跟随程序的生命周期。

　　2)第二种是常驻型，也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。

　　47. 请解释下在单线程模型中Message、Handler、MessageQueue、Looper之间的关系。Handler简介：

　　一个Handler允许你发送和处理Message和Runable对象，这些对象和一个线程的MessageQueue相关联。每一个线程实例和一个单独的线程以及该线程的MessageQueue相关联。当你创建一个新的Handler时，它就和创建它的线程绑定在一起了。这里，线程我们也可以理解为线程的MessageQueue。从这一点上来看，Handler把Message和Runable对象传递给MessageQueue，而且在这些对象离开MessageQueue时，Handler负责执行他们。

　　Handler有两个主要的用途：(1)确定在将来的某个时间点执行一个或者一些Message和Runnable对象。(2)在其他线程(不是Handler绑定线程)中排入一些要执行的动作。

　　Scheduling Message，即(1)，可以通过以下方法完成：

　　post(Runnable):Runnable在handler绑定的线程上执行，也就是说不创建新线程。

　　postAtTime(Runnable,long):

　　postDelayed(Runnable,long):

　　sendEmptyMessage(int):

　　sendMessage(Message):

　　sendMessageAtTime(Message,long):

　　sendMessageDelayed(Message,long):

　　post这个动作让你把Runnable对象排入MessageQueue,MessageQueue受到这些消息的时候执行他们，当然以一定的排序。sendMessage这个动作允许你把Message对象排成队列，这些Message对象包含一些信息，Handler的hanlerMessage(Message)会处理这些Message.当然，handlerMessage(Message)必须由Handler的子类来重写。这是编程人员需要作的事。

　　当posting或者sending到一个Hanler时，你可以有三种行为：当MessageQueue准备好就处理，定义一个延迟时间，定义一个精确的时间去处理。后两者允许你实现timeout,tick,和基于时间的行为。

　　当你的应用创建一个新的进程时，主线程(也就是UI线程)自带一个MessageQueue，这个MessageQueue管理顶层的应用对象(像activities,broadcast receivers等)和主线程创建的窗体。你可以创建自己的线程，并通过一个Handler和主线程进行通信。这和之前一样，通过post和sendmessage来完成，差别在于在哪一个线程中执行这么方法。在恰当的时候，给定的Runnable和Message将在Handler的MessageQueue中被Scheduled。

　　Message简介：

　　Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象提供额外的两个int域和一个Object域，这可以让你在大多数情况下不用作分配的动作。

　　尽管Message的构造函数是public的，但是获取Message实例的最好方法是调用Message.obtain(),或者Handler.obtainMessage()方法，这些方法会从回收对象池中获取一个。

　　MessageQueue简介：

　　这是一个包含message列表的底层类。Looper负责分发这些message。Messages并不是直接加到一个MessageQueue中，而是通过MessageQueue.IdleHandler关联到Looper。

　　你可以通过Looper.myQueue()从当前线程中获取MessageQueue。

　　Looper简介：

　　Looper类被用来执行一个线程中的message循环。默认情况，没有一个消息循环关联到线程。在线程中调用prepare()创建一个Looper，然后用loop()来处理messages，直到循环终止。

　　大多数和message loop的交互是通过Handler。

　　下面是一个典型的带有Looper的线程实现。

　　class LooperThread extends Thread {

　　public Handler mHandler;

　　public void run() {

　　Looper.prepare();

　　mHandler = new Handler() {

　　public voidhandleMessage(Message msg) {

　　// process incomingmessages here

　　}

　　};

　　Looper.loop();

　　}

　　}

　　48. AIDL的全称是什么?如何工作?能处理哪些类型的数据?AIDL的英文全称是Android Interface Define Language

　　当A进程要去调用B进程中的service时，并实现通信，我们通常都是通过AIDL来操作的

　　A工程：

　　首先我们在net.blogjava.mobile.aidlservice包中创建一个RemoteService.aidl文件，在里面我们自定义一个接口，含有方法get。ADT插件会在gen目录下自动生成一个RemoteService.java文件，该类中含有一个名为RemoteService.stub的内部类，该内部类中含有aidl文件接口的get方法。

　　说明一：aidl文件的位置不固定，可以任意

　　然后定义自己的MyService类，在MyService类中自定义一个内部类去继承RemoteService.stub这个内部类，实现get方法。在onBind方法中返回这个内部类的对象，系统会自动将这个对象封装成IBinder对象，传递给他的调用者。

　　其次需要在AndroidManifest.xml文件中配置MyService类，代码如下：

　　为什么要指定调用AIDL服务的ID,就是要告诉外界MyService这个类能够被别的进程访问，只要别的进程知道这个ID，正是有了这个ID,B工程才能找到A工程实现通信。

　　说明：AIDL并不需要权限

　　B工程：

　　首先我们要将A工程中生成的RemoteService.java文件拷贝到B工程中，在bindService方法中绑定aidl服务

　　绑定AIDL服务就是将RemoteService的ID作为intent的action参数。

　　说明：如果我们单独将RemoteService.aidl文件放在一个包里，那个在我们将gen目录下的该包拷贝到B工程中。如果我们将RemoteService.aidl文件和我们的其他类存放在一起，那么我们在B工程中就要建立相应的包，以保证RmoteService.java文件的报名正确，我们不能修改RemoteService.java文件

　　bindService(newInten("net.blogjava.mobile.aidlservice.RemoteService"),serviceConnection, Context.BIND\_AUTO\_CREATE);

　　ServiceConnection的onServiceConnected(ComponentName name, IBinderservice)方法中的service参数就是A工程中MyService类中继承了RemoteService.stub类的内部类的对象。

　　49. 请解释下Android程序运行时权限与文件系统权限的区别。运行时权限Dalvik( android授权)

　　文件系统 linux 内核授权

　　50. 系统上安装了多种浏览器，能否指定某浏览器访问指定页面?请说明原由。通过直接发送Uri把参数带过去，或者通过manifest里的intentfilter里的data属性

　　51. 你如何评价Android系统?优缺点。答：Android平台手机 5大优势：

　　一、开放性

　　在优势方面，Android平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到Android联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。开放性对于Android的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

　　二、挣脱运营商的束缚

　　在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年iPhone 上市 ，用户可以更加方便地连接网络，运营商的制约减少。随着EDGE、HSDPA这些2G至3G移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，当你可以通过手机IM软件方便地进行即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样?互联网巨头Google推动的Android终端天生就有网络特色，将让用户离互联网更近。

　　三、丰富的硬件选择

　　这一点还是与Android平台的开放性相关，由于Android的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。功能上的差异和特色，却不会影响到数据同步、甚至软件的兼容，好比你从诺基亚 Symbian风格手机 一下改用苹果 iPhone ，同时还可将Symbian中优秀的软件带到iPhone上使用、联系人等资料更是可以方便地转移，是不是非常方便呢?

　　四、不受任何限制的开发商

　　Android平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如可控制正是留给Android难题之一。

　　五、无缝结合的Google应用

　　如今叱诧互联网的Google已经走过10年度历史，从搜索巨人到全面的互联网渗透，Google服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而Android平台手机将无缝结合这些优秀的Google服务。

　　再说Android的5大不足：

　　一、安全和隐私

　　由于手机与互联网的紧密联系，个人隐私很难得到保守。除了上网过程中经意或不经意留下的个人足迹，Google这个巨人也时时站在你的身后，洞穿一切，因此，互联网的深入将会带来新一轮的隐私危机。

　　二、首先开卖Android手机的不是最大运营商

　　众所周知，T-Mobile在23日，于美国纽约发布 了Android首款手机G1。但是在北美市场，最大的两家运营商乃AT&T和Verizon，而目前所知取得Android手机销售权的仅有 T-Mobile和Sprint，其中T-Mobile的3G网络相对于其他三家也要逊色不少，因此，用户可以买账购买G1，能否体验到最佳的3G网络服务则要另当别论了!

　　三、运营商仍然能够影响到Android手机

　　在国内市场，不少用户对购得移动定制机不满，感觉所购的手机被人涂画了广告一般。这样的情况在国外市场同样出现。Android手机的另一发售运营商Sprint就将在其机型中内置其手机商店程序。

　　四、同类机型用户减少

　　在不少手机论坛都会有针对某一型号的子论坛，对一款手机的使用心得交流，并分享软件资源。而对于Android平台手机，由于厂商丰富，产品类型多样，这样使用同一款机型的用户越来越少，缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，就很少有专门针对某个型号山寨机的讨论和群组，除了哪些功能异常抢眼、颇受追捧的机型以外。

　　五、过分依赖开发商缺少标准配置

　　在使用PC端的Windows Xp系统的时候，都会内置微软Windows Media Player这样一个浏览器程序，用户可以选择更多样的播放器，如Realplay或暴风影音等。但入手开始使用默认的程序同样可以应付多样的需要。在Android平台中，由于其开放性，软件更多依赖第三方厂商，比如Android系统的SDK中就没有内置音乐 播放器，全部依赖第三方开发，缺少了产品的统一性。

　　52. 什么是ANR 如何避免它?

　　答：ANR：Application NotResponding，五秒

　　在Android中，活动管理器和窗口管理器这两个系统服务负责监视应用程序的响应。当出现下列情况时，Android就会显示ANR对话框了：

　　对输入事件(如按键、触摸屏事件)的响应超过5秒

　　意向接受器(intentReceiver)超过10秒钟仍未执行完毕

　　Android应用程序完全运行在一个独立的线程中(例如main)。这就意味着，任何在主线程中运行的，需要消耗大量时间的操作都会引发ANR。因为此时，你的应用程序已经没有机会去响应输入事件和意向广播(Intentbroadcast)。

　　因此，任何运行在主线程中的方法，都要尽可能的只做少量的工作。特别是活动生命周期中的重要方法如onCreate()和 onResume()等更应如此。潜在的比较耗时的操作，如访问网络和数据库;或者是开销很大的计算，比如改变位图的大小，需要在一个单独的子线程中完成(或者是使用异步请求，如数据库操作)。但这并不意味着你的主线程需要进入阻塞状态已等待子线程结束 -- 也不需要调用Therad.wait()或者Thread.sleep()方法。取而代之的是，主线程为子线程提供一个句柄(Handler)，让子线程在即将结束的时候调用它(xing:可以参看Snake的例子，这种方法与以前我们所接触的有所不同)。使用这种方法涉及你的应用程序，能够保证你的程序对输入保持良好的响应，从而避免因为输入事件超过5秒钟不被处理而产生的ANR。这种实践需要应用到所有显示用户界面的线程，因为他们都面临着同样的超时问题。

　　53. 什么情况会导致Force Close ?如何避免?能否捕获导致其的异常?

　　答：一般像空指针啊，可以看起logcat，然后对应到程序中 来解决错误

　　54. Android本身的api并未声明会抛出异常，则其在运行时有无可能抛出runtime异常，你遇到过吗?诺有的话会导致什么问题?如何解决?

　　55. 简要解释一下activity、 intent 、intent filter、service、Broadcase、BroadcaseReceiver

　　答：一个activity呈现了一个用户可以操作的可视化用户界面

　　一个service不包含可见的用户界面，而是在后台无限地运行

　　可以连接到一个正在运行的服务中，连接后，可以通过服务中暴露出来的借口与其进行通信

　　一个broadcast receiver是一个接收广播消息并作出回应的component，broadcastreceiver没有界面

　　intent:content provider在接收到ContentResolver的请求时被激活。

　　activity, service和broadcast receiver是被称为intents的异步消息激活的。

　　一个intent是一个Intent对象，它保存了消息的内容。对于activity和service来说，它指定了请求的操作名称和待操作数据的URI

　　Intent对象可以显式的指定一个目标component。如果这样的话，android会找到这个component(基于manifest文件中的声明)并激活它。但如果一个目标不是显式指定的，android必须找到响应intent的最佳component。

　　它是通过将Intent对象和目标的intent filter相比较来完成这一工作的。一个component的intent filter告诉android该component能处理的intent。intent filter也是在manifest文件中声明的。

　　56. 　IntentService有何优点?

　　答：IntentService 的好处

　　\* Acitivity的进程，当处理Intent的时候，会产生一个对应的Service

　　\* Android的进程处理器现在会尽可能的不kill掉你

　　\* 非常容易使用

　　57. 横竖屏切换时候activity的生命周期?

　　1、不设置Activity的android:configChanges时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次

　　2、设置Activity的android:configChanges="orientation"时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次

　　3、设置Activity的android:configChanges="orientation|keyboardHidden"时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法

　　如何将SQLite数据库(dictionary.db文件)与apk文件一起发布?

　　解答：可以将dictionary.db文件复制到Eclipse Android工程中的res aw目录中。所有在res aw目录中的文件不会被压缩，这样可以直接提取该目录中的文件。可以将dictionary.db文件复制到res aw目录中

　　58. 如何将打开res aw目录中的数据库文件?

　　解答：在Android中不能直接打开res aw目录中的数据库文件，而需要在程序第一次启动时将该文件复制到手机内存或SD卡的某个目录中，然后再打开该数据库文件。复制的基本方法是使用getResources().openRawResource方法获得res aw目录中资源的 InputStream对象，然后将该InputStream对象中的数据写入其他的目录中相应文件中。在Android SDK中可以使用SQLiteDatabase.openOrCreateDatabase方法来打开任意目录中的SQLite数据库文件。

　　59. Android引入广播机制的用意?

　　答：a:从MVC的角度考虑(应用程序内)

　　其实回答这个问题的时候还可以这样问，android为什么要有那4大组件，现在的移动开发模型基本上也是照搬的web那一套MVC架构，只不过是改了点嫁妆而已。android的四大组件本质上就是为了实现移动或者说嵌入式设备上的MVC架构，它们之间有时候是一种相互依存的关系，有时候又是一种补充关系，引入广播机制可以方便几大组件的信息和数据交互。

　　b：程序间互通消息(例如在自己的应用程序内监听系统来电)

　　c：效率上(参考UDP的广播协议在局域网的方便性)

　　d：设计模式上(反转控制的一种应用，类似监听者模式)

　　60. Android dvm的进程和Linux的进程, 应用程序的进程是否为同一个概念

　　DVM指dalivk的虚拟机。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。而每一个DVM都是在Linux 中的一个进程，所以说可以认为是同一个概念。

　　61. sim卡的EF 文件有何作用

　　sim卡的文件系统有自己规范，主要是为了和手机通讯，sim本 身可以有自己的操作系统，EF就是作存储并和手机通讯用的

　　62. 嵌入式操作系统内存管理有哪几种， 各有何特性

　　页式，段式，段页，用到了MMU,虚拟空间等技术

　　63. 什么是嵌入式实时操作系统, Android 操作系统属于实时操作系统吗?

　　嵌入式实时操作系统是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统作出快速响应，并控制所有实时任务协调一致运行的嵌入式操作系统。主要用于工业控制、 军事设备、 航空航天等领域对系统的响应时间有苛刻的要求，这就需要使用实时系统。又可分为软实时和硬实时两种，而android是基于linux内核的，因此属于软实时。

　　64. 一条最长的短信息约占多少byte?

　　中文70(包括标点)，英文160，160个字节。

　　65. android中的动画有哪几类，它们的特点和区别是什么?

　　两种，一种是Tween动画、还有一种是Frame动画。Tween动画，这种实现方式可以使视图组件移动、放大、缩小以及产生透明度的变化;另一种Frame动画，传统的动画方法，通过顺序的播放排列好的图片来实现，类似电影。

　　66. handler机制的原理

　　andriod提供了Handler 和 Looper 来满足线程间的通信。Handler先进先出原则。Looper类用来管理特定线程内对象之间的消息交换(MessageExchange)。

　　1)Looper: 一个线程可以产生一个Looper对象，由它来管理此线程里的MessageQueue(消息队列)。

　　2)Handler: 你可以构造Handler对象来与Looper沟通，以便push新消息到MessageQueue里;或者接收Looper从Message Queue取出)所送来的消息。

　　3) Message Queue(消息队列):用来存放线程放入的消息。

　　4)线程：UIthread 通常就是main thread，而Android启动程序时会替它建立一个MessageQueue。

　　67. 说说mvc模式的原理，它在android中的运用

　　MVC(Model\_view\_contraller)”模型\_视图\_控制器”。 MVC应用程序总是由这三个部分组成。Event(事件)导致Controller改变Model或View，或者同时改变两者。只要Controller改变了Models的数据或者属性，所有依赖的View都会自动更新。类似的，只要Contro

　　68. DDMS和TraceView的区别? DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器 。

　　69. java中如何引用本地语言 可以用JNI(java nativeinterface java 本地接口)接口。

　　70. 谈谈Android的IPC(进程间通信)机制 IPC是内部进程通信的简称， 是共享"命名管道"的资源。Android中的IPC机制是为了让Activity和Service之间可以随时的进行交互，故在Android中该机制，只适用于Activity和Service之间的通信，类似于远程方法调用，类似于C/S模式的访问。通过定义AIDL接口文件来定义IPC接口。Servier端实现IPC接口，Client端调用IPC接口本地代理。

　　71. NDK是什么NDK是一些列工具的集合，NDK提供了一系列的工具，帮助开发者迅速的开发C/C++的动态库，并能自动将so和java 应用打成apk包。

　　NDK集成了交叉编译器，并提供了相应的mk文件和隔离cpu、平台等的差异，开发人员只需简单的修改mk文件就可以创建出so

**1.常用的存储方式有哪些？**

（五种，说出哪五种，五种存储方式什么情况下用。）注意sharepreferes对象支持读取不支持写入，写入引用Editor。

SQLite：

SQLite是一个轻量级的数据库，支持基本SQL语法，是常被采用的一种数据存储方式。Android为此数据库提供了一个名为SQLiteDatabase的类，封装了一些操作数据库的API。

SharedPreference：

除SQLite数据库外，另一种常用的数据存储方式，其本质就是一个xml文件，常用于存储较简单的参数设置。

File：

即常说的文件（I/O）存储方法，常用语存储大数量的数据，但是缺点是更新数据将是一件困难的事情。

ContentProvider:

Android 系统中能实现所有应用程序共享的一种数据存储方式，由于数据通常在各应用间的是互相私密的，所以此存储方式较少使用，但是其又是必不可少的一种存储方式。 例如音频，视频，图片和通讯录，一般都可以采用此种方式进行存储。每个ContentProvider都会对外提供一个公共的URI（包装成Uri对 象），如果应用程序有数据需要共享时，就需要使用ContentProvider为这些数据定义一个URI，然后其他的应用程序就通过 Content Provider传入这个URI来对数据进行操作。

网络存储：

从网络读取数据和写入数据。 Android提供了通过网络来实现数据的存储和获取的方法。

 我们可以调用WebService返回的数据或是解析HTTP协议实现网络数据交互。

**2.安卓中动画有几类，他们的特点和区别？**

   两种，一种是补间动画，还有一种是帧动画，帧动画类似于放电影，通过播放已经排列放好的图片来实现。

补间动画的实现定义开始和结束，中间由系统计算得出效果有透明，位移，放大缩小等等。

自定义录音或者进度条过程，会应用到帧动画，补间动画，一般用于应用的欢迎界面。

(菜单弹出，例如赞和评论，是个动画过程。)

**3.handler机制原理**

Handler主要用于线程间的通信。

一个Handler允许发送和处理Message和Runable对象，UI主线程会自动分配一个Looper（消息轮询器），每个Looper中封装着MessageQueue（消息队列），遵循先进先出原则。Looper负责不断的从自己的消息队列里取出队头的任务或消息执行。一般是在子线程执行完耗时操作之后，通过Handler的sendMessage或post方法将Message和Runable对象传递给MessageQueue，而且在这些对象离开MessageQueue时，Handler负责执行他们（用到handleMessage方法，主要执行刷新UI的代码）。   
其中Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象提供额外的两个int域和一个Object域。

字面以外加上关联点。

**4除了handler和子线程，还有一个处理线程的是什么，主要方法是什么？**

AsynTask，doInbackGround+onPostExecute

doInBackground() 这个方法运行在后台线程中，主要负责执行那些很耗时的操作，如访问网络。该方法必须重写。

onPostExecute(Result) 这个方法运行于UI主线程，在doInBackground(Params…)方法执行后调用，该方法用于接收后台任务执行后返回的结果，刷新UI显示

**@5.tcp和udp区别.**

TCP---传输控制协议,提供的是面向连接、可靠的字节流服务,传输数据前经过“三次握手”建立连接，保证数据传输的可靠性，但效率比较低。一般用于对于数据传输安全性较高的场合。

UDP---用户数据报协议，是一个简单的面向数据报的运输层协议，面向无连接。UDP不提供可靠性，数据传输可能发生错序，丢包，但效率较高。一般用于对于实时性要求较高的场合。

**6线程之间的通信方式（如何避免ANR）**

（1）. AsyncTask，其中doInBackground()和onPostExecute(Result)两个方法非常重要

doInBackground() 这个方法运行在后台线程中，主要负责执行那些很耗时的操作，如访问网络。该方法必须重写。

onPostExecute(Result) 这个方法运行于UI主线程，在doInBackground(Params…)方法执行后调用，该方法用于接收后台任务执行后返回的结果，刷新UI显示。

（2.）子线程 + handler

在子线程中执行完耗时操作需要刷新UI时，通过handler.sendMessage()发消息给主线程， 然后在主线程Handler中的handleMessage()方法中执行刷新UI操作

1. **activity的生命周期**

**/**/ 开始 重新启动 停止 销毁 暂停

七个，oncreate，onstart，onrestart，onstop，onresume，ondestroy，onpause；（清楚整个周期的过程）

1）.启动Activity：系统会先调用onCreate方法，然后调用onStart方法，最后调用onResume，Activity进入运行状态。

2）.当前Activity被其他Activity覆盖其上或被锁屏：系统会调用onPause方法，暂停当前Activity的执行。

3）.当前Activity由被覆盖状态回到前台或解锁屏：系统会调用onResume方法，再次进入运行状态。

4）.当前Activity转到新的Activity界面或按Home键回到主屏，自身退居后台：系统会先调用onPause方法，然后调用onStop方法，进入停滞状态。

5）.用户后退回到此Activity：系统会先调用onRestart方法，然后调用onStart方法，最后调用onResume方法，再次进入运行状态。

6）.当前Activity处于被覆盖状态或者后台不可见状态，即第2步和第4步，系统内存不足，杀死当前Activity，而后用户退回当前Activity：再次调用onCreate方法、onStart方法、onResume方法，进入运行状态。

7）.用户退出当前Activity：系统先调用onPause方法，然后调用onStop方法，最后调用onDestory方法，结束当前Activity。

**8.ArrayList和LinkedList区别？**

存数据，ArrayList数组存储数据，索引值以下标来搜索，查询比较方，删除增加比较麻烦，但是linkedList以链表式存储数据，对于增删比较方便。

**9.安卓内存的优化？\***

1）静态变量引起内存泄露

在代码优化的过程中，我们需要对代码中的静态变量特别留意。静态变量是类相关的变量， 它的生命周期是从这个类被声明，到这个类彻底被垃圾回收器回收才会被销毁。所以，一般情况下，静态变量从所在的类被使用开始就要一直占用着内存空间，直到 程序退出。如果不注意，静态变量引用了占用大量内存的资源，造成垃圾回收器无法对内存进行回收，就可能造成内存的浪费

2）使用Application的Context

在Android中，Application Context的生命周期和应用的生命周期一样长，而不是取决于某个Activity的生命周期。如果想保持一个长期生命的对象，并且这个对象需要一个 Context，就可以使用Application对象。可以通过调用Context.getApplicationContext()方法或者 Activity.getApplication()方法来获得Application对象。

3）及时关闭资源

Cursor是Android查询数据后得到的一个管理数据集合的类。正常情况下，如 果我们没有关闭它，系统会在回收它时进行关闭，但是这样的效率特别低。如果查询得到的数据量较小时还好，如果Cursor的数据量非常大，特别是如果里面 有Blob信息时，就可能出现内存问题。所以一定要及时关闭Cursor。

4）使用Bitmap及时调用recycle() // 把 重复循环的方法设置 为 null，释放内存

前面的章节讲过，在不使用Bitmap对象时，需要调用recycle()释放内存，然后将它设置为null。虽然调用recycle()并不能保证立即释放占用的内存，但是可以加速Bitmap的内存的释放。

在代码优化的过程中，如果发现某个Activity用到了Bitmap对象，却没有显式的调用recycle()释放内存，则需要分析代码逻辑，增加相关代码，在不再使用Bitmap以后调用recycle()释放内存。

    5）对Adapter进行优化

下面以构造ListView的BaseAdapter为例说明如何对Adapter进行优化。

@软引用和弱引用。

如果一个对象只具有软引用，那么如果内存空间足够，垃圾回收器就不会回收它；如果内存 空间不足了，就会回收这些对象的内存。只要垃圾回收器没有回收它，该对象就可以被程序使用。软引用可用来实现内存敏感的高速缓存。软引用可以和一个引用队 列（ReferenceQueue）联合使用，如果软引用所引用的对象被垃圾回收，Java虚拟机就会把这个软引用加入到与之关联的引用队列中。

如果一个对象只具有弱引用，那么在垃圾回收器线程扫描的过程中，一旦发现了只具有弱引 用的对象，不管当前内存空间足够与否，都会回收它的内存。不过，由于垃圾回收器是一个优先级很低的线程，因此不一定会很快发现那些只具有弱引用的对象。弱 引用也可以和一个引用队列（ReferenceQueue）联合使用，如果弱引用所引用的对象被垃圾回收，Java虚拟机就会把这个弱引用加入到与之关联 的引用队列中。

弱引用与软引用的根本区别在于：只具有弱引用的对象拥有更短暂的生命周期，可能随时被回收。而只具有软引用的对象只有当内存不够的时候才被回收，在内存足够的时候，通常不被回收。

**UI优化**

在Android应用开发过程中，屏幕上控件的布局代码和程序的逻辑代码通常是分开 的。界面的布局代码是放在一个独立的xml文件中的，这个文件里面是树型组织的，控制着页面的布局。通常，在这个页面中会用到很多控件，控件会用到很多的 资源。Android系统本身有很多的资源，包括各种各样的字符串、图片、动画、样式和布局等等，这些都可以在应用程序中直接使用。这样做的好处很多，既 可以减少内存的使用，又可以减少部分工作量，也可以缩减程序安装包的大小。

**10.framgment生命周期？**

oncreate，onstart，onrestart，onstop，onresume，ondestroy，onpause，onAttach，onCreateView，onDettach，onDestroyView;（和activity的生命周期比较，会更好）

 MyFragment onAttach() 粘贴到activity上  
MyFragment onCreate() fragment创建  
MyFragment onCreateView() fragment创建自己的视图

MainActivity onCreate()  
MyFragment onActivityCreated() 可以处理fragment数据的初始化

MainActivity onStart()  
 MyFragment onStart()

MainActivity onResume()  
MyFragment onResume()

按后退键  
MyFragment onPause()  
MainActivity onPause()

 MyFragment onStop()  
 MainActivity onStop()

MyFragment onDestoryView() 销毁掉自己的视图  
MyFragment onDestory()  
MyFragment onDetach() 解除和activity的关系

MainActivity onDestory()

**@11·图片异步加载怎么做？**

可定义图片异步加载工具类，核心方式实现思路如下：

1.先从内存缓存（Map<String,SoftReference<Bitmap>>中获取图片显示

2.获取不到的话从本地SD卡里获取并显示

3.都获取不到的话通过子线程从网络加载图片并保存到内存及SD卡中并通过handler显示

**@12·service和Thread区别？**

[http://wenku.baidu.com/link?url=p-P0sCqgr7y1w-mkd-B8DIwpIppzEud9MCVtttYWp1AWAmfZEfLyqKYKvAJWtXE2N2X8WS4vNHutJRBaG\_RAC9i1AwqY5Wh7Zh4s-XHixju](http://wenku.baidu.com/link?url=4OAMuQ6CYG6IjdyYH6y5zo9kX222dhotZo6sAmhVH7I-6inE8tZVWfs9US4njk2753sYEAK4zK9na5HxjTyKVzFcMQfl9GF5CLwbCLJUynu（详细介绍的路径）)

1）.Thread：Thread 是程序执行的最小单元，它是分配CPU的基本单位。可以用 Thread 来执行一些异步的操作。

2).Service：Service 是android的一种机制，当它运行的时候如果是Local Service，那么对应的 Service 是运行在主进程的 main 线程上的。如：onCreate，onStart 这些函数在被系统调用的时候都是在主进程的 main 线程上运行的。如果是Remote Service，那么对应的 Service 则是运行在独立进程的 main 线程上

**13·内存泄露如何解决？**

<http://wenku.baidu.com/view/03dc4b46b307e87100f69604.html>

1）、   数据库的cursor没有关闭

2）、 构造adapter没有使用缓存contentview

   衍生的listview优化问题：减少创建View的对象，充分使用contentview，可以使用静态类来处理优化getView的过程 3、Bitmap对象不使用时采用recycle()释放内存 4、Activity中的对象生命周期大于Activity

）Bitmap对象不使用recycle（）释放内存

）Activity中的对象生命周期大于Activity；

内存泄露大多数都是由于相同的错误导致的，即：对Context拥有较长时间的引用。

避免Context相关的内存泄露，记住以下事情：

   不要保留对Context-Activity长时间的引用（对Activity的引用的时候，必须确保拥有和Activity一样的生命周期）

   尝试使用Context-Application来替代Context-Activity

   如果你不想控制内部类的生命周期，应避免在Activity中使用非静态的内部类，而应该使用静态的内部类，并在其中创建一个对Activity的弱引用。这种情况的解决办法是使用一个静态的内部类，其中拥有对外部类的WeakReference，如同ViewRoot和它的Winner类那样

   GC（垃圾回收）不能解决内存泄露问题

**14·MVC的使用？**

mvc是model,view,controller的缩写，mvc包含三个部分：   
  
　　l模型（model）对象：是应用程序的主体部分，所有的业务逻辑都应该写在该层。   
  
　　l视图（view）对象：是应用程序中负责生成用户界面的部分。也是在整个mvc架构中用户唯一可以看到的一层，接收用户的输入，显示处理结果。   
  
　　l控制器（control）对象：是根据用户的输入，控制用户界面数据显示及更新model对象状态的部分，控制器更重要的一种导航功能，想用用户出发的相关事件，交给m哦得了处理。   
  
　　android鼓励弱耦合和组件的重用，在android中mvc的具体体现如下：   
  
　 　1)视图层（view）：一般采用xml文件进行界面的描述，使用的时候可以非常方便的引入，当然，如何你对android了解的比较的多了话，就一定 可以想到在android中也可以使用javascript+html等的方式作为view层，当然这里需要进行java和javascript之间的通 信，幸运的是，android提供了它们之间非常方便的通信实现。   
  
　　2)控制层（controller）：android的控制层的重 任通常落在了众多的acitvity的肩上，这句话也就暗含了不要在acitivity中写代码，要通过activity交割model业务逻辑层处理， 这样做的另外一个原因是android中的acitivity的响应时间是5s，如果耗时的操作放在这里，程序就很容易被回收掉。   
  
　　3)模型层（model）：对数据库的操作、对网络等的操作都应该在model里面处理，当然对业务计算等操作也是必须放在的该层的。

**15·常用的设计模式**

**单例设计模式**

       所谓单例设计模式简单说就是无论程序如何运行，采用单例设计模式的类（Singleton类）永远只会有一个实例化对象产生。具体实现步骤如下：

      (1) 将采用单例设计模式的类的构造方法私有化（采用private修饰）。

      (2) 在其内部产生该类的实例化对象，并将其封装成private static类型。

      (3) 定义一个静态方法返回该类的实例。

**工厂设计模式**

       程序在接口和子类之间加入了一个过渡端，通过此过渡端可以动态取得实现了共同接口的子类实例化对象

**代理设计模式**

       指由一个代理主题来操作真实主题，真实主题执行具体的业务操作，而代理主题负责其他相关业务的处理。比如生活中的通过代理访问网络，客户通过网络代理连接网络（具体业务），由代理服务器完成用户权限和访问限制等与上网相关的其他操作（相关业务）。

**观察者设计模式**

       所谓观察者模式，举个例子现在许多购房者都密切观察者房价的变化，当房价变化时，所有购房者都能观察到，以上的购房者属于观察者，这便是观察者模式。

       java中可以借助Observable类和Observer接口轻松实现以上功能。当然此种模式的实现也不仅仅局限于采用这两个类。

**适配器模式**

        如果一个类要实现一个具有很多抽象方法的接口，但是本身只需要实现接口中的部分方法便可以达成目的，所以此时就需要一个中间的过渡类，但此过渡类又不希望 直接使用，所以将此类定义为抽象类最为合适，再让以后的子类直接继承该抽象类便可选择性的覆写所需要的方法，而此抽象类便是适配器类。

**16·面向对象的特征？**

**封装**

封装最好理解了。封装是面向对象的特征之一，是对象和类概念的主要特性。

封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

**继承**

面向对象编程 (OOP) 语言的一个主要功能就是“继承”。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

**多态**

多态性（polymorphisn）是允许你将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。

实现多态，有二种方式，覆盖，重载。

覆盖，是指子类重新定义父类的虚函数的做法。

重载，是指允许存在多个同名函数，而这些函数的参数表不同（或许参数个数不同，或许参数类型不同，或许两者都不同）。

**17·程序结束的时候，如何处理Activity的？\***

记录打开的Activity：（**http://zdpeng.iteye.com/blog/1576055**）

每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可。

**18·配置文件中存放什么？**

四大组件，意图，权限，第三方key，版本号等

**19·Final ，finally，finalized，区别；**

final用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不

可继承。

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此

方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

**20·NDK是什么？**

NDK 是 Native Development Kit 的简称。它是一个工具集，集成了 Android 的交叉编译环境，并提供了一套比较方便的 Makefile ，可以帮助开发者快速开发 C 或是 C++ 的动态库，并自动的将 so 和 java 程序打包成 apk ，在 Android 上运行。Android 上，应用程序的开发，大部分基于 Java 语言来实现。要使用 c 或是 c++ 的程序或库，就需要使用 NDK 来实现。

**21百度地图和谷歌地图的优缺点？**

**优势：**

1一打开百度地图页面，显示的就是目前所在城市地图，谷歌地图不然。

2百度地图的界面，百度地图很多都是在显眼位置用文字做提示，一目了然，谷歌界面有点炫，很多用图标来表示，但是用起来相对复杂，不熟悉的用户需要确定这个图标是否是所需要选择的。

而百度地图搜索框下就直接有“公交”“驾车”的选项，界面简洁直接很多，谷歌地图需要点击获取路线，再点击小图标来选择“公交”这些选项，而且图标不够引人注目，使用起来相对复杂些，试想一个很少用网络地图的人分别使用二者地图，那个更方便呢？应该是[百度](http://www.zhihujingxuan.com/baidu/)，大多数用户通过地图是查询位置-这个地方在哪，周围有什么地物，查询路线-从一个方到另一个地方如何到达，用百度地图，这些问题很容易解决，而用谷歌地图得摆弄一会儿。

3百度地图的查询路线更好,首先还是界面更简洁直接，然后是数据的问题，先不管为他们提供数据的公司如何，刚刚特意同时查了下，同样地起点和终点， 百度地图给出了5条路线，谷歌地图给出的是4条路线，而且百度地图路线的选择时可以选择“较快捷”“少换乘”“少步行”三种方式，更方便。打车费用也直接 显示出来，而且个人体会10元内的打车费用基本正确。

4百度地图有个视野内搜索功能，可以很方便地查找到目标位置，谷歌地图没有。这个很重要。

5一个重要界面布局问题，百度地图是地图视野在左，而谷歌地图是地图视野在右，我们现在时喜欢从左到右看东西啊，这点百度地图更符合用户习惯。 （当然也可能是我常用百度地图，所以这么觉得）

 PS:刚在知乎上有朋友说到——因“左图右列表”强调的是“地图”，反之强调的是“搜索”；我们发现用户在经过这么些年的教育后，“不在乎”地图在哪边，相对其它体验，这一点影响用户选择服务的权重并不高。

6百度地图已经和都市圈合作推出了部分城市如北京、上海、深圳、广州的三维地图，谷歌地图没有。

7百度地图地图视野上方有个截图功能，可以直接拉框截图，谷歌地图没有。

8以北京实时路况信息为例，直接输入北京查询，百度地图显示的是主城区的实时路况，而谷歌显示的区域很大 ，包括了密云、廊坊市了，但是实时路况信息还是显示的主城区，显示路况的彩色条就全挤在一起了，设计得不如百度好。

8使用百度的占大多数，经常可以听到“百度一下啊”的话语。随之带来百度地图的使用率也相对较高。

9百度地图的地图百宝箱、房产、天气等信息。

**劣势：**

1如果需要从网络地图上获取数据的话，谷歌地图完胜，而且还有谷歌地球在一旁辅助，如虎添翼，从网上各种各样的谷歌地图相关工具软件就可以看出。

2路线查询时输入地点名称却有多个符合条件时，会给出一个下拉菜单给予选择，同一地点谷歌地图提供的数据精确到街道，而百度地图精确到区。

3谷歌地图可以将侧边栏隐藏起来，看到更大的地图视野，而百度地图是点击全屏来显示更大的视野，两种切换方式比起来谷歌的更舒服。

4谷歌地图有个分享链接功能，而且还提供“粘贴 HTML 以便嵌入到网站”这一功能，百度没有。

5界面布局百度地图更简洁，但是色彩搭配上和画面感上谷歌地图更好。

6谷歌的品牌形象占优。

**22·安卓的系统架构**

<http://wenku.baidu.com/view/3ef50eddd15abe23482f4d7e.html>

应用层：使用Java语言进行开发的一些应用程序

应用框架曾：主要是谷歌发布的一些操作支持的类库（API框架），开发人员可以使用这些类库方便的进行程序开发，但是在开发时必须遵守框架的开发原则

系统运行库层：当使用安卓框架层进行开发时，安卓操作系统会自动使用一些c/c++的库文件来支持所使用的各个组件，使其可以更好的为程序服务；

Linux内核层：安卓操作系统主要是基于Linux内核，程序的安全性，驱动程序，进程管理等都由Linux内核提供。

**23.listView优化**

1)、convertView复用，对convetView进行判空，当convertView不为空时重复使用，为空则初始化，从而减少了很多不必要的View的创建

2)定义一个ViewHolder，封装Listview Item条目中所有的组件，将convetView的tag设置为ViewHolder,不为空时通过ViewHolder的属性获取对应组件即可

3)、当ListView加载数据量较大时可以采用分页加载和图片异步加载

**24·安卓上安装了多个浏览器，能否指定页面访问；**

通过直接发送URI把参数带过去，或者通过Manifest里的intentfilter的data属性

**25·简述音乐播放的方式与区别？**

1）：MediaPlayer：主要用于播放音频，可以播放视频，但是一般不用其进行视频播放。

2）：SoundPool： 主要用于播放一些短促的声音片段，主要优势是cpu资源占有量低，反应延迟小。

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

**26·工作中如何利用线程池的？】**

Java通过Executors提供四种线程池，分别为：   
newCachedThreadPool创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。  
newFixedThreadPool 创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。  
newScheduledThreadPool 创建一个定长线程池，支持定时及周期性任务执行。  
newSingleThreadExecutor 创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。

**27·onTouch和onTouchEvent区别？**

1.onTouch方法：  
onTouch方法是View的 OnTouchListener接口中定义的方法。  
当一个View绑定了OnTouchLister后，当有touch事件触发时，就会调用onTouch方法。  
（当把手放到View上后，onTouch方法被一遍一遍地被调用）  
  
2.onTouchEvent方法：  
onTouchEvent方法是override 的Activity的方法。  
重新了Activity的onTouchEvent方法后，当屏幕有touch事件时，此方法就会别调用。

**28·JSON的结构？**

json是一种轻量级的数据交换格式，

json简单说就是对象和数组，所以这两种结构就是对象和数组两种结构，通过这两种结构可以表示各种复杂的结构

1、对象：对象表示为“{}”扩起来的内容，数据结构为 {key：value,key：value,...}的键值对的结构，在面向对象的语言中，key为对象的属性，value为对应的属性值，所以很容易理解，取值方法为 对象.key 获取属性值，这个属性值的类型可以是 数字、字符串、数组、对象几种。

2、数组：数组在json中是中括号“[]”扩起来的内容，数据结构为 ["java","javascript","vb",...]，取值方式和所有语言中一样，使用索引获取，字段值的类型可以是 数字、字符串、数组、对象几种。

经过对象、数组2种结构就可以组合成复杂的数据结构了。

**29·SIM卡的EF文件有何作用？**

### SIM卡的文件系统有自己的规范，主要是为了和手机通讯，SIM本身可以有自己的操作系统，EF就是作 存储并和手机通讯用的。

**30.最长的短信多长，多少个字节？**

70个汉字，140个字节

**31·&和&&的区别**

一个位运算符，一个是逻辑运算符

**32·Java的基本特征**

**封装**

封装最好理解了。封装是面向对象的特征之一，是对象和类概念的主要特性。

封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

**继承**

面向对象编程 (OOP) 语言的一个主要功能就是“继承”。继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。

**多态**

多态性（polymorphisn）是允许你将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。

实现多态，有二种方式，覆盖，重载。

覆盖，是指子类重新定义父类的虚函数的做法。

重载，是指允许存在多个同名函数，而这些函数的参数表不同（或许参数个数不同，或许参数类型不同，或许两者都不同）。

**33·注册广播的方式**

android中，不同进程之间传递信息要用到广播，可以有两种方式来实现。   
第一种方式：在Manifest.xml中注册广播，是一种比较推荐的方法，因为它不需要手动注销广播（如果广播未注销，程序退出时可能会出错）。   
具体实现在Manifest的application中添加：   
<receiver android:name=".mEvtReceiver">   
<intent-filter>   
<action android:name="android.intent.action.BOOT\_COMPLETED" />   
</intent-filter>   
</receiver>   
上面两个android:name分别是广播名和广播的动作（这里的动作是表示系统启动完成），如果要自己发送一个广播，在代码中为：   
Intent i = new Intent("android.intent.action.BOOT\_COMPLETED");   
sendBroadcast(i);   
这样，广播就发出去了，然后是接收。   
接收可以新建一个类，继承至BroadcastReceiver，也可以建一个BroadcastReceiver的实例，然后得写onReceive方法，实现如下：   
protected BroadcastReceiver mEvtReceiver = new BroadcastReceiver() {   
@Override   
public void onReceive(Context context, Intent intent) {   
String action = intent.getAction();   
if (action.equals("android.intent.action.BOOT\_COMPLETED")) {   
//Do something   
}   
}   
};   
  
第二种方式，直接在代码中实现，但需要手动注册注销，实现如下：   
IntentFilter filter = new IntentFilter();   
filter.addAction("android.intent.action.BOOT\_COMPLETED");   
registerReceiver(mEvtReceiver, filter); //这时注册了一个recevier ,名为mEvtReceiver，然后同样用上面的方法以重写onReceiver，   
最后在程序的onDestroy中要注销广播，实现如下：   
@Override   
public void onDestroy() {   
super.onDestroy();   
unregisterReceiver(mPlayerEvtReceiver);   
}   
  
  
  
Android系统中的广播是广泛用于应用程序之间通信的一种手段，它类似于事件处理机制，不同的地方就是广播的处理是系统级别的事件处理过程（一般事件处理是控件级别的）。在此过程中仍然是离不开Intent对象，理解广播事件的处理过程，灵活运用广播处理机制，在关键之处往往能实现特别的效果，   
在Android 中如果要发送一个广播必须使用sendBroadCast 向系统发送对其感兴趣的广播接收器中。   
使用广播必须要有一个intent 对象必设置其action动作对象   
使用广播必须在配置文件中显式的指明该广播对象   
每次接收广播都会重新生成一个接收广播的对象   
在BroadCast 中尽量不要处理太多逻辑问题，建议复杂的逻辑交给Activity 或者 Service 去处理

**34·异步和同步的区别？**

同步：提交请求->等待服务器处理->处理完毕返回 这个期间客户端浏览器不能干任何事

（顺序执行处理事件）

异步: 请求通过事件触发->服务器处理（这是浏览器仍然可以作其他事情）->处理完毕

（可同时执行多个事件的并发）

**35·Afinal和XUtils的区别？**

Xutils是基于Afinal的

xUtils 包含了很多实用的android工具。

xUtils 源于Afinal框架，对Afinal进行了适当的精简，和一些适度的扩展和重构。

xUtils 具有Afinal的一些特性如：无需考虑bitmap在android中加载的时候oom的问题和快速滑动的时候图片加载位置错位等问题； 简洁，约定大于配置...

// oom==内存溢出 ( 释放内存 bitmap )

// 对XUtils 的理解: Dbutils 、ViewUtils、HttpUtils、BitmapUtils

DbUtils模块：android中的orm框架，一行代码就可以进行增删改查。

ViewUtils模块：android中的ioc框架，完全注解方式就可以进行UI绑定和事件绑定。

HttpUtils模块：支持同步，异步方式的请求，支持大文件上传；支持GET,POST,PUT,MOVE,COPY,DELETE,HEAD请求，支持multipart上传设置subtype如related。

BitmapUtils模块：加载bitmap的时候无需考虑bitmap加载过程中出现的oom和android容器快速滑动时候出 现的图片错位等现象；内存管理使用lru算法，更好的管理bitmap内存；可配置线程加载线程数量，缓存大小，缓存路径，加载显示动画等...

**// Android 中ANR异常的处理方法 ：( 意思是: 应用程序没有响应 )**

**有两种情况会报ANR: 自己添加的**

1. 主线程在5秒内没有响应输入事件（事件处理/UI线程）
2. 广播在10秒内没有完成返回( BroadcastReceiver )

**下面是导致android中出现ANR异常的的情况:**

1. **在主线程内进行网络操作**
2. **在主线程内进行一些缓慢的磁盘操作(例如执行没有优化的SQL查询)**

**处理的逻辑是:**

**1.New一个新的线程，进行数据请求**

**2.获取数据后，调用handler.sendmessage方法**

**3.在handler的handle()方法中更新UI**

**// Android 如何指定浏览器访问 指定页面?**

指定action 及 uri 就可以访问制定好的 浏览器了

1>简而言之: 一个程序至少有一个进程，一个进程至少有一个线程

2>线程的划分尺度小于进程，使的多线程程序的并发性高

3>进程在执行的过程中拥有独立的内存单元，而多个线程共享内存，从而提高了程序的运行效率

4>每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制

5>多线程的意义在于一个应用程序中，有多个执行部分可以同时执行， 但是操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理及分配资源，这就是进程和线程的重要区别

**36·ContentProvider的URI的配置？**

清单文件之指定URI或者代码里面指定URI，contentProvider通过URI访问数据

**37·**AIDL的全称是什么，如何工作

AIDL的英文全称是Android Interface Define Language

当A进程要去调用B进程中的service时，并实现通信，我们通常都是通过AIDL来操作的

A工程：

首先我们在net.blogjava.mobile.aidlservice包中创建一个RemoteService.aidl文件，在里面我们自定义一个接口，含有方法get。ADT插件会在gen目录下自动生成一个RemoteService.java文件，该类中含有一个名为RemoteService.stub的内部类，该内部类中含有aidl文件接口的get方法。

说明一：aidl文件的位置不固定，可以任意

然后定义自己的MyService类，在MyService类中自定义一个内部类去继承RemoteService.stub这个内部类，实现get方法。在onBind方法中返回这个内部类的对象，系统会自动将这个对象封装成IBinder对象，传递给他的调用者。

其次需要在AndroidManifest.xml文件中配置MyService类，代码如下：

<!-- 注册服务 -->

<service android:name=".MyService">

<intent-filter>

<!-- 指定调用AIDL服务的ID -->

<action android:name="net.blogjava.mobile.aidlservice.RemoteService" />

</intent-filter>

</service>

为什么要指定调用AIDL服务的ID,就是要告诉外界MyService这个类能够被别的进程访问，只要别的进程知道这个ID，正是有了这个ID,B工程才能找到A工程实现通信。

说明：AIDL并不需要权限

B工程：

首先我们要将A工程中生成的RemoteService.java文件拷贝到B工程中，在bindService方法中绑定aidl服务

绑定AIDL服务就是将RemoteService的ID作为intent的action参数。

说明：如果我们单独将RemoteService.aidl文件放在一个包里，那个在我们将gen目录下的该包拷贝到B工程中。如果我们将RemoteService.aidl文件和我们的其他类存放在一起，那么我们在B工程中就要建立相应的包，以保证RmoteService.java文件的报名正确，我们不能修改RemoteService.java文件

bindService(new Inten("net.blogjava.mobile.aidlservice.RemoteService"), serviceConnection, Context.BIND\_AUTO\_CREATE);

ServiceConnection的onServiceConnected(ComponentName name, IBinder service)方法中的service参数就是A工程中MyService类中继承了RemoteService.stub类的内部类的对象。

Android采用RPC的方式来实现（remote procedure call）远程通信，并且Android通过接口定义语言AIDL来生成两个进程之间互相访问的代码。例如，你在Activity里的代码需要访问Service中的一个方法，那么就可以通过这种方法来实现了。

? 创建一个AIDL文件（参见代码）

? 实现AIDL文件生成的Java接口（参见代码）

? 将你的接口暴露给客户端

? 客户端调用

**38·屏幕适配？**

ه最好可以通过权重(layout\_weight)的方式来分配每个组件的大小，也可以通过具体的像素(dip)来确定大小。  
尽量使用Relativelayout  。  
已知应用支持平台设备的分辨率,可以提供多个layout\_320\*480  ...  
drawable-hdpi,drawable-mdpi,drawable-ldpi分别代表分辨率为480\*800,360\*480,240\*360, 放置图片大小相差1.5倍  
最后还需要在AndroidManifest.xml里添加下面一段，没有这一段自适应就不能实现：  
<supports-screens  
android:largeScreens="true"  
android:normalScreens="true"  
  android:anyDensity = "true"/>  
在</application>标签和</manifest> 标签之间添加上面那段代码。即可。  
备注：三者的解析度不一样，就像你把电脑的分辨率调低，图片会变大一样，反之分辨率高，图片缩小  
还可以通过.9.png实现图片的自适应

**// 应用升级( 版本号升级 ) 或软件升级，数据库就会跟着升级**

应用升级==版本号升级 == 同时数据库也会 跟着升级

软件升级 == 同时数据库也会跟着升级

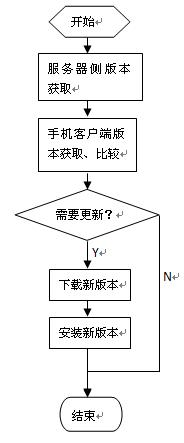
**39·版本更新?数据库的更新？跨版本？**

版本：

在清单文件更改版本号，上传更新的应用到平台，利用平台的更新提醒用户是否更新新版本。

数据库更新：

先更改版本号，在onupgress方法内判断版本号是否一致，不一致则更新数据库。



**40·安卓自定义组件？**

Android自定义组件有三种实现思路:

1) 继承某个现有组件，在其基础上添加额外功能,如继承Gallery实现CoverFlow效果

2) 复合型组件定义: 继承某个Layout，实现复合组件自定义，如TextView和EditText组合实现登录注册组件

3) 继承View，实现onDraw()方法，实现自己绘制组件，如翻页效果组件

**41.listview分页加载的步骤**

通常实现分页加载有两种方式，一种是在ListView底部设置一个按钮，用户点

击即加载。另一种是当用户滑动到底部时自动加载。

在ListView底部设置一个按钮，用户点击即加载实现思路:

// 加上底部View，注意要放在setAdapter方法前

ListView.addFooterView(moreView);

bt.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

pg.setVisibility(View.VISIBLE);// 将进度条可见

bt.setVisibility(View.GONE);// 按钮不可见

handler.postDelayed(new Runnable() {

@Override

public void run() {

loadMoreDate();// 加载更多数据

bt.setVisibility(View.VISIBLE);

pg.setVisibility(View.GONE);

mSimpleAdapter.notifyDataSetChanged();// 通知listView刷新数据

}

}, 2000);

}

});

当用户滑动到底部时自动加载实现思路: 实现OnScrollListener 接口重写

onScrollStateChanged 和onScroll方法，使用onscroll方法实现”滑动“后处理检查是否还有新的记录，如果有,添加记录到adapter, adapter调用 notifyDataSetChanged 更新数据;如果没有记录了，则不再加载数据。使用onScrollStateChanged可以检测是否滚到最后一行且停止滚动然后执行加载.

**42.移动互联数据的交换格式**

移动互联数据交互格式有XML和JSON

1.JSON和XML的数据可读性基本相同

2.JSON和XML同样拥有丰富的解析手段

3.JSON相对于XML来讲，数据的体积小

4.JSON与JavaScript的交互更加方便

5.JSON对数据的描述性比XML较差

6.JSON的速度要远远快于XML

**43.xmpp的底层是怎样实现的？**

socket，双广播，连接广播和通知光爆

**44.Activity的启动模式（四种）**

Standard、singleTop、singleTask、singleInstance

**standard**默认模式，可以不用写配置。在这个模式下，都会默认创建一个新的实例。因此，在这种模式下，可以有多个相同的实例，也允许多个相同Activity叠加。

**singleTop**  
可以有多个实例，但是不允许多个相同Activity叠加。即，如果Activity在栈顶+的时候，启动相同的Activity，不会创建新的实例，而会调用其onNewIntent方法。

**singleTask**  
只有一个实例。在同一个应用程序中启动他的时候，若Activity不存在，则会在当前task创建一个新的实例，若存在，则会把task中在其之上的其它Activity destory掉并调用它的onNewIntent方法。  
如果是在别的应用程序中启动它，则会新建一个task，并在该task中启动这个Activity，singleTask允许别的Activity与其在一个task中共存，也就是说，如果我在这个singleTask的实例中再打开新的Activity，这个新的Activity还是会在singleTask的实例的task中。

**singleInstance**  
只有一个实例，并且这个实例独立运行在一个task中，这个task只有这个实例，不允许有别的Activity存在。

**45.广播机制的作用 ????????**

广播机制，本质上它就是一种组件间的通信方式，如果是两个组件位于不同的进程当中，那么可以用Binder机制来实现，如果两个组件是在同一个进程中，那么它们之间可以用来通信的方式就更多了，这样看来，广播机制似乎是多余的。然而，广播机制却是不可替代的，它和Binder机制不一样的地方在于，广播的发送者和接收者事先是不需要知道对方的存在的，这样带来的好处便是，系统的各个组件可以松耦合地组织在一起，这样系统就具有高度的可扩展性，容易与其它系统进行集成。

实现思路：

 在第1步中，广播的接收者把广播接收器注册到ActivityManagerService中；在第2步中，广播的发送者同样是把广播发送到ActivityManagerService中，由ActivityManagerService去查找注册了这个广播的接收者，然后把广播分发给它们。

        在第2步的分发的过程，其实就是把这个广播转换成一个消息，然后放入到接收器所在的线程消息队列中去，最后就可以在消息循环中调用接收器的onReceive函数了。这里有一个要非常注意的地方是，由于ActivityManagerService把这个广播放进接收器所在的线程消息队列后，就返回了，它不关心这个消息什么时候会被处理，因此，对广播的处理是异步的，即调用sendBroadcast时，这个函数不会等待这个广播被处理完后才返回

实例连接：http://wenku.baidu.com/view/b73dd968af1ffc4ffe47ac0c.html

**46.图片轮播的实现思路**

图片轮播是由viewpage和handler两者结合所实现的，

**47.Sleep和wait的区别**

1.这两个方法来自不同的类分别是，sleep来自Thread类，和wait来自Object

类。

2.最主要是sleep方法没有释放锁，而wait方法释放了锁，使得其他线程可

以使用同步控制块或者方法。sleep不出让系统资源；wait是进入线程等待池

等待，出让系统资源，其他线程可以占用CPU。一般wait不会加时间限制，

因为如果wait线程的运行资源不够，再出来也没用，要等待其他线程调用

notify/notifyAll唤醒等待池中的所有线程，才会进入就绪队列等待OS分配系

统资源。sleep(milliseconds)可以用时间指定使它自动唤醒过来，如果时间不到

只能调用interrupt()强行打断。

3.wait，notify和notifyAll只能在同步控制方法或者同步控制块里面使用，而

sleep可以在任何地方使用

4. Sleep需要捕获异常,而wait不需要

**48.全局异常处理**

[Android使用UncaughtExceptionHandler捕获全局异常](http://blog.csdn.net/hehe9737/article/details/7662123)（未捕获的异常处理程序）

    Android系统的“程序异常退出”，给应用的用户体验造成不良影响。为了捕获应用运行时异常并给出友好提示，便可继承[UncaughtExceptionHandler](http://orgcent.com/tag/UncaughtExceptionHandler/)类来处理。通过Thread.setDefaultUncaughtExceptionHandler()方法将异常处理类设置到线程上即可。  
**Demo下载地址：**<http://code.google.com/p/android-custom-view/downloads/list>

**49.静态变量合实例变量的区别**

语法：静态变量前要加static来修饰，而实例变量前则不加；

程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量；静态变量不属于某个实例对象，而是属于类，所以也称为类变量，静态变量就会被分配空间，静态变量就可以被使用了。

**50.listview失去焦点怎么处理**

在listview子布局里面写，可以解决焦点失去的问题

android:descendantFocusability="blocksDescendants"

**51.什么是匿名内部类，在什么时候调用**

   内部类：内部类可以是static的或者非static的，static内部类只能包含静态方法和静态类变量，只能访问外部类的静态元素，内部类可以实例化，多次使用。

匿名内部类：它只能使用一次，不区分static和非static。如果用到外部类的变量的话，必须是类变量或者实例变量，就是必须是类定义的变量，或者final的局部变量。匿名内部类如果是继承某个类的话是可以重写那个类的方法的，这个和普通内部类一样。

实现事件监听器的时候

用匿名内部类编码非常简洁，也容易读懂，不重复利用时使用。

**52.android中asset和raw的区别；**

\*res/raw和assets的相同点：

1.两者目录下的文件在打包后会原封不动的保存在apk包中，不会被编译成二进制。

\*res/raw和assets的不同点：

1.res/raw中的文件会被映射到R.java文件中，访问的时候直接使用资源ID即R.id.filename；assets文件夹下的文件不会被映射到R.java中，访问的时候需要AssetManager类。  
2.res/raw不可以有目录结构，而assets则可以有目录结构，也就是assets目录下可以再建立文件夹

\*读取文件资源：

1.读取res/raw下的文件资源，通过以下方式获取输入流来进行写操作

InputStream is = getResources().openRawResource(R.id.filename);

//通过 资源 id 直接打开 raw 下的 流文件

2.读取assets下的文件资源，通过以下方式获取输入流来进行写操作

AssetManager am = null;

am = getAssets();

InputStream is = am.open("filename");

// 通过 AssectManager 管理器 来打开 assect 下的流文件

**53.什么是线程**

线程，有时被称为[轻量级进程](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=65949654)(Lightweight Process，LWP)，是程序执行流的最小[单元](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=275431)。一个标准的线程由线程[ID](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=265569)，当前指令指针(PC)，[寄存器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=35071)集合和[堆栈](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=331232)组成。另外，线程是[进程](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=288622)中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有[系统](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=349837)资源，只拥有一点在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以[创建](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=100768)和[撤消](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=126328)另一个线程，同一进程中的多个线程之间可以并发执行。由于线程之间的相互制约，致使线程在运行中呈现出间断性。线程也有[就绪](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=425221)、[阻塞](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=646488)和[运行](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=493544)三种基本状态。每一个程序都至少有一个线程，那就是程序本身。

　　线程是程序中一个单一的顺序控制流程。在单个程序中同时运行多个线程完成不同的工作，称为[多线程](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=452880)。

// 自己找的

**// 线程与进程的区别:**

**定义:**

1. 进程是系统进行资源分配和调度的一个独立单位
2. 线程是进程的一个实体，是CPU调度和分派的基本单位，线程自己基本上不拥有系统资源，只拥有一点在系统运行中必不可少的资源，但是它可与同属一个进程的其他线程共享进程所拥有的全部资源

**关系:**

1. 一个线程可以创建和撤销另一个线程，同一个进程中的多个线程可以同时并发执行，他可以与同进程中的其他线程共享数据，但拥有自己的栈空间，拥有独立的执行顺序

**区别:**

进程和线程区别: 他们是不同的操作系统资源管理方式。

进程有独立的地址空间，一个进程崩溃后，不会对其他进程产生影响

线程只是一个进程中不同的执行路径，线程有自己的堆栈和局部变量，线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉， 所以多线程的操作 要比多进程的程序健壮。 但在进程切换时耗费资源较大，效率要差一些

对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程，（因为进程在切换时耗费的资源要大一些，效率差一些）

**54.线程和进程的区别**

进程是表示资源分配的基本单位，又是调度运行的基本单位。

线程是进程中执行运算的最小单位，亦即执行处理机调度的基本单位。

线程的优点：

（1）易于调度。

（2）提高并发性。通过线程可方便有效地实现并发性。进程可创建多个线程来执行同一程序的不同部分。

（3）开销少。创建线程比创建进程要快，所需开销很少。。

（4）利于充分发挥多处理器的功能。通过创建多线程进程（即一个进程可具有两个或更多个线程），每个线程在一个处理器上运行，从而实现应用程序的并发性，使每个处理器都得到充分运行。

进程和线程的关系：

（1）一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程。  （2）资源分配给进程，同一进程的所有线程共享该进程的所有资源。

（3）处理机分给线程，即真正在处理机上运行的是线程。

（4）线程在执行过程中，需要协作同步。不同进程的线程间要利用消息通信的办法实现同步。

**55.接口和抽象类的区别**

abstract可以修饰抽象方法，而一个类只要有一个抽象方法，就必须用

abstract定义该类，即抽象类。

用interface修饰的类，里面的方法都是抽象方法，因此在定义接口的时候，可以

直接不加那些修饰，系统会默认的添上去。接口里面的字段都是公有常量，即

public static final修饰的字段。

//接口是单继承 、抽象类是多继承

//接口里面只有抽象方法

// 抽象类: 有抽象方法和普通方法 、 抽象类==父类( 抽象类是 顶级父类 )

// 比较两个对象是否相等 == 比较的是地址、 = 比较的是值

//final不能和 static 、actaret 一起使用

**56.xml有几种解析方式、区别**

基本的解析方式有三种: DOM,SAX,Pull

dom解析：解析器读入整个文档，然后构建一个驻留内存的树结构，然后就可以使用 DOM 接口来操作这个树结构。优点是对文档增删改查比较方便，缺点占用内存比较大。

sax解析：基于事件驱动型,优点占用内存少，解析速度快，缺点是只适合做文档的读取，不适合做文档的增删改，不能中途停止。

pull解析：同样基于事件驱动型,android 官方API提供,可随时终止,调用next() 方法提取它们（主动提取事件）

**57.webview的辅助类**

WebSettings、WebViewClient、WebChromeClient

**WebSettings**

设置WebView的一些属性、状态等,例如允许使用javascript，允许使用缓存，允许使用内置的缩放组件

setJavaScriptEnabled(true); 设置支持js

**WebViewClient**

主要帮助WebView处理各种通知、请求事件（例如，点击链接时候如何显示界面，页面开始加载，加载完毕之后有何动作等）

shouldOverrideUrlLoading() onPageStarted() onPageFinished()

**WebChromeClient**

辅助WebView处理Javascript的对话框、网站图标、网站Title、加载进度等

核心方法有onJsAlert() onJsConfirm() onJsPrompt() onProgressChanged() onReceivedIcon()

onReceivedTitle()

**58.webview加载网页的两种方式**

loadUrl(url)  通过url地址加载互联网或本地网页

如：loadUrl("http://www.baidu.com"); // 互联网地址

    loadUrl("file://android\_asset/html/index.html"); /／本地网页

loadData(data,type,encoding) 直接加载网页内容，容易乱码，一般用 [loadDataWithBaseURL](http://localhost/android-new/reference/android/webkit/WebView.html#loadDataWithBaseURL(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String))代替

**59.Intent可以被传递的数据类型**

intent是连接Activity, Service, BroadcastReceiver, ContentProvider四大组件的信使,，可以传递八种基本数据类型以及string, Bundle类型，以及实现了Serializable或者Parcelable的类型。

Intent可以划分成显式意图和隐式意图。

显式意图：调用Intent.setComponent()或Intent.setClass()方法明确指定了组件名的Intent为显式意图，显式意图明确指定了Intent应该传递给哪个组件。

隐式意图：没有明确指定组件名的Intent为隐式意图。 Android系统会根据隐式意图中设置的动作(action)、类别(category)、数据（URI和数据类型）找到最合适的组件来处理这个意图。

**60.拦截短信**

短信其实是一个有序广播

1：创建一个继承BroadcastReceiver类的广播接收器。

2：在清单文件中注册广播并在<intent-filter>中设置优先级 priority（只要比你想拦截程序的短信接收器优先级高就可以了）

3：在onReceiver（）中调用abortBroadcast()就可以了

**61.Throws与Throw的区别**

1.throw则是指抛出的一个具体的异常类型。

2.通常在一个方法（类）的声明处通过throws声明方法（类）可能抛出的异常信息，而在方法（类）内部通过throw声明一个具体的异常信息。

3.throws通常不用显示的捕获异常，可由系统自动将所有捕获的异常信息抛给上级方法；

throw则需要用户自己捕获相关的异常，而后在对其进行相关包装，最后在将包装后的异常信息抛出。

如果在方法中会有异常被抛出而你又不希望在这个方法体内对此异常进行处理，可以使用throws在声明方法的时候同时声明他可能会跑出的异常。

4.两者位置不同.

**62.Service是如何启动**

1通过startService  
　　Service会经历onCreate->onStart

　　stopService的时候直接onDestroy  
　　  
　　如果是调用者(TestServiceHolder)自己直接退出而没有调用stopService的  
　　话，Service会一直在后台运行。  
　　下次TestServiceHolder再起来可以stopService。

2通过bindService  
　　Service只会运行onCreate，这个时候TestServiceHolder和TestService绑定在一起  
　　  
　　TestServiceHolder退出了，Srevice就会调用onUnbind->onDestroyed  
　　所谓绑定在一起就共存亡了。

**63.android虚拟机的进程和linux的关系**

DVM指dalivk的虚拟机。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。而每一个DVM都是在Linux中的一个进程，所以说可以认为是同一个概念。

**64.程序运行的权限和文件系统权限的区别**

运行时权限 Dalvik( android 授权) 文件系统 linux 内核授权

**65.访问网络如何加密**

1：对称加密（ＤＥＳ，ＡＥＳ）和非对称（ＲＳＡ公钥与私钥）。

（支付宝里的商户的公钥和私钥）

2：MD5（算法）

3：Base64

**66.访问网络都有哪两种方式**

get与post是两种不同的提交方式

get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。

对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。

get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。

get安全性非常低，post安全性较高。

HTTP 定义了与服务器交互的不同方法，最基本的方法是 GET 和 POST。事实上 GET 适用于多数请求，而保留 POST 仅用于更新站点。

**67.一键退出的实现**

定义一个类继承Application，定义一个集合存放所有的activity，定义一个添

加的方法，再写一个退出的方法，使用for循环全部调用finish方法，然后在每个

Activity的onCreate方法中调用自定义类里的添加方法，然后在需要使用一键退出

的地方调用类中的退出方法即可。

**68.gc是什么**

GC是垃圾收集的意思（Gabage Collection）,内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显示操作方法。

**69.listview的分页加载**

ListView分页加载通常有两种实现方式：一种是ListView底部设置一个按钮，用户点击即加载；另一种是当用户滑动到底部时自动加载。

在此说一下第一种方式点击底部按钮即加载的实现思路 ：

通过addFooterView()方法为listview底部加入一个“加载更多”按钮和加载进度的视图，默认按钮显示，加载进度为gone。当用户点击“加载更多”按钮时，按钮设置为gone，加载进度显示，并开启子线程从网络上加载下一页数据，加载完毕，发消息给handler，再handler的handleMessage（）方法中通过adapter的notifyDataSetChanged方法通知listview刷新，显示刚加入的数据，并恢复按钮显示，加载进度隐藏。

**70.webservice的UDDI是什么**

UDDI 是一种[目录服务](http://baike.baidu.com/view/1351410.htm)，企业可以使用它对 Web services 进行注册和搜索。UDDI，英文为 "Universal Description, Discovery and Integration"，可译为“通用描述、发现与集成服务”。

**71、webService中uddi是指什么**

UDDI是一种目录服务，企业可以使用它对Web Service进行注册和搜索。UDDI,英文为“Universal Description and Integration”，可译为“通用描述、发现与集成服务”。

**72、说说java的反射机制原理以及应用场合**

JAVA反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有

属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态

获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。

类对象: 封装了类的描述信息的对象,类加载的产物,由jvm创建java.lang.Class

应用场景: Gson 序列化反序列化

获取类对象的方式:1. 类名.class2. 类的对象.getClass()3. Class.forName("

包名.类名")

**73、contentprovider怎么实现数据共享**

一个程序可以通过实现一个Content provider的抽象接口将自己的数据完全暴露出去，而且Content providers是以类似数据库中表的方式将数据暴露。Content providers存储和检索数据，通过它可以让所有的应用程序访问到，这也是应用程序之间唯一共享数据的方法。

要想使应用程序的数据公开化，可通过2种方法：创建一个属于你自己的Content provider或者将你的数据添加到一个已经存在的Content provider中，前提是有相同数据类型并且有写入Content provider的权限。

如何通过一套标准及统一的接口获取其他应用程序暴露的数据？

Android提供了ContentResolver，外界的程序可以通过ContentResolver接口访问ContentProvider提供的数据。

**74、java有几种引用类型**

强引用，软引用、弱引用、幽灵引用、

**75、如何让Activity变成一个窗口？**

答：在清单文件AndroidManifest.xml中相应的<activity>标签内设置属性android:theme=”@android:style/The

**76，启动一个线程时run（）还是start（）**

start（）

**77，ArrayList的便利**

ArrayList<String> al=new ArrayList<String>();

for(int i=0;i<10;i++){

al.add(String.valueOf(i));

}

Iterator<String> it=al.iterator();

while(it.hasNext()){

System.out.println(it.next());

}

**78、String,StringBuffer区别**

JAVA平台提供了两个类：String和StringBuffer，它们可以储存和操作字符串，

即包含多个字符的字符数据。这个String类提供了数值不可改变的字符串。而这个

StringBuffer类提供的字符串进行修改。

StringBuilder与StringBuffer的区别 。

StringBuffer

1，一个类似于 String 的字符串缓冲区，对它的修改不会像String那样重创建对象。

2，使用append()方法修改Stringbuffer的值，使用toString()方法转换为字符串。

Stringbuild是jdk1.5后用来替换stringBuffer的一个类，大多数时候可以替换

StringBuffer。和StringBuffer的区别在于Stringbuild是一个单线程使用的类，不值

执行线程同步所以比StringBuffer的速度快，效率高。

**79、集合的实现类**

HashSet：元素无需、不可重复；ArrayList：元素有序，可重复

HashMap以兼职对的形式伯村数据，key值可重复，value值不可重复

**80、android中常用的布局**

第一种：帧布局（框架布局）FrameLayout，在这个布局中，所有的子元素统统放于这块区域的左上角，并且后面的子元素直接覆盖在前面的子元素之上，将前面的子元素部分和全部遮挡。

第二种：线性布局LinearLayout，最常用的一种布局方式，所有子控件的对齐方式，取决于如何定义 orientation的属性：vertical 垂直方向 ，如果按照这种方向所有的子控件将按照垂直的方式分布在布局上，每行只允许有一个子元素，horizontal水平方向 ，这时子控件将会以水平的方向分布在布局中。

第三种：绝对布局AbsoluteLayout，又可以叫做坐标布局，可以直接指定子元素的绝对位置，这种布局简单直接，直观性强，但是由于手机屏幕尺寸差别比较大，使用绝对定位的适应性会比较差。

第四种：相对布局RelativeLayout，允许子元素指定它们相对于其父元素或兄弟元素的位置，这是实际布局中最常用的布局方式之一。它灵活性大很多，当然属性也多，操作难度也大，属性之间产生冲突的的可能性也大，使用相对布局时要多做些测试。

第五种：表格布局TableLayout，表格布局TableLayout以行列的形式管理子元素，每一行是一个TableRow布局对象，当然也可以是普通的View对象，TableRow里每放一个元素就是一列，总列数由列数最多的那一行决定。

第六种：网格布局 GridLayout，在Android 4.0中，新引入的GridLayout网格布局，GridLayout布局使用虚细线将布局划分为行，列和单元格，也支持一个控件在行，列上都有交错排列。而GridLayout使用的其实是跟LinearLayout类似的API，只不过是修改了一下相关的标签而已，所以对于开发者来说，掌握GridLayout还是很容易的事情。

**81、char，varchar区别**

1、CHAR。CHAR存储定长数据很方便，CHAR字段上的索引效率级高，比如定义char(10)，那么不论你存储的数据是否达到了10个字节，都要占去10个字节的空间。

2、VARCHAR。存储变长数据，但存储效率没有CHAR高。如果一个字段可能的值是不固定长度的，我们只知道它不可能超过10个字符，把它定义为 VARCHAR(10)是最合算的。VARCHAR类型的实际长度是它的值的实际长度+1。为什么“+1”呢？这一个字节用于保存实际使用了多大的长度。

**82.io流读取输出方法**

文件读取：

public static void main(String[] args) throws IOException

{

//创建一个文件读取流对象，并初始化时指定名称的文件相关联。

//要保证此文件是已存在，如不存在会发生异常：FileNotFoundException

FileReader fr = new FileReader("demo.txt");

//调用读取流对象的read方法。

//read方法一次只读一个字符，而且会自动向下读取。当读取完后会返回-1.

int ch = 0;

while((ch = fr.read())!=-1)

System.out.print((char)ch);

fr.close();

}

**83，如何序列化**

1). 实现序列化 Serializable

2). 属性私有化，并提供get，set方法

3). 提供无参构造

4). 属性名必须与json串中属性名保持一致 （因为Gson解析json串底层用到了Java的反射原理）

**84、如何在android中调用js**

loadUrl("javascript:test()");

**85、**Intent的原理

intent是连接Activity, Service, BroadcastReceiver, ContentProvider四大组件的信使,，可以传递八种基本数据类型以及string, Bundle类型，以及实现了Serializable或者Parcelable的类型。

Intent可以划分成显式意图和隐式意图。

显式意图：调用Intent.setComponent()或Intent.setClass()方法明确指定了组件名的Intent为显式意图，显式意图明确指定了Intent应该传递给哪个组件。

隐式意图：没有明确指定组件名的Intent为隐式意图。 Android系统会根据隐式意图中设置的动作(action)、类别(category)、数据（URI和数据类型）找到最合适的组件来处理这个意图。

**86、广播分为几种**

第一种:在清单文件中声明,添加

<receive android:name=".IncomingSMSReceiver ">

<intent-filter>

<action android:name="android.provider.Telephony.SMS\_RECEIVED")

<intent-filter>

<receiver>

第二种使用代码进行注册如:

IntentFilter filter = new IntentFilter("android.provider.Telephony.SMS\_RECEIVED");

IncomingSMSReceiver receiver = new IncomgSMSReceiver();

registerReceiver(receiver.filter);

**87、intent和activity的区别**

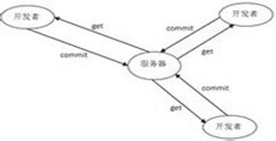
Activity活动，可以理解为窗口，是人机交互的核心，所以又被人们说成是所有程序的根本。Intent意图，负责所有Activity窗口通讯。Service服务是相对Activity来说的，不需要人机交互，但可以为Activity提供交互必需要的一些东西。

Android中，Activity是所有程序的根本，所有程序的流程都运行在Activity之中，Activity具有自己的生命周期. 由系统控制生命周期，程序无法改变，但可以用onSaveInstanceState保存其状态对于Activity，关键是其生命周期的把握（如下图），其次就是状态的保存和恢复（onSaveInstanceState onRestoreInstanceState），以及Activity之间的跳转和数据传输（intent）。

Intent

Android中提供了Intent机制来协助 应用间的交互与通讯，Intent负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述，Android则根据此Intent的描述，负责找到对 应的组件，将 Intent传递给调用的组件，并完成组件的调用。Intent不仅可用于应用程序之间，也可用于应用程序内部的Activity/Service之间的交互。

**88、**解释SVN

Svn是一种集中式文件版本管理系统。集中式管理的工作流程如下图：集中式代码管理的核心是服务器，所有开发者在开始新一天的工作之前必须从服务器获取代码，然后开发，最后解决冲突，提交。所有的版本信息都放在服务器上。如果脱离了服务器，开发者基本上是不可以工作。下面举例说明：开始新一天的工作：1，从服务器下载项目组最新代码。2：进入自己的分支，进行工作，每隔一个小时向服务器自己的分支提交一次代码（很多人都有这个习惯。因为有时候自己对代码改来改去，最后又想还原到前一个小时的版本，或者看看前一个小时自己修改了那些代码，就需要这样做了）。3：下班时间快到了，把自己的分支合并到服务器主分支上，一天的工作完成，并反映给服务器。这就是经典的svn工作流程，从流程上看，有不少缺点，但也有优点。缺点：1、 服务器压力太大，数据库容量暴增。2、 如果不能连接到服务器上，基本上不可以工作，看上面第二步，如果服务器不能连接上，就不能提交，还原，对比等等。3、不适合开源开发（开发人数非常非常多，但是Google app engine就是用svn的）。但是一般集中式管理的有非常明确的权限管理机制（例如分支访问限制），可以实现分层管理，从而很好的解决开发人数众多的问题。优点：1、 管理方便，逻辑明确，符合一般人思维习惯。2、 易于管理，集中式服务器更能保证安全性。3、 代码一致性非常高。4、 适合开发人数不多的项目开发。5、大部分软件配置管理的大学教材都是使用svn和vss。

**89、ipc机制（及原理）**

）IPC（Inter-Process Communication，进程间通信）,

aidl是 Android Interface definition language的缩写，它是一种android内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口.编译器可以通过扩展名为aidl的文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的.

BroadcastReceiver也可以实现进程间通信

ContentProvider 提供进程间数据共享

2).Android的 Service工作流程

1.Android虚拟机启动之前系统会先启动service Manager进程；

2.service Manager打开binder驱动，并通知binder kernel驱动程序这个进程将作为System Service Manager；

3.然后该进程将进入一个循环，等待处理来自其他进程的数据。

4.用户创建一个System service后，通过defaultServiceManager得到一个远程ServiceManager的接口，通过这个接口我们可以调用addService函数将System service添加到Service Manager进程中；

5.然后client可以通过getService获取到需要连接的目的Service的IBinder对象，这个IBinder是Service的BBinder在binder kernel的一个参考，

6.所以service IBinder 在binder kernel中不会存在相同的两个IBinder对象，每一个Client进程同样需要打开Binder驱动程序。对用户程序而言，我们获得这个对象就可以通过binder kernel访问service对象中的方法。

7.Client与Service在不同的进程中，通过这种方式实现了类似线程间的迁移的通信方式，对用户程序而言当调用Service返回的IBinder接口后，访问Service中的方法就如同调用自己的函数。

**90、八种基本数据类型**

byte、short、int、long、float、double、char、boolean

**91、上传图片怎么做**

用图片异步加载通过开启子线程实现

**92、异常**

答：编译时异常： 程序正确，但因为外在的环境条件不满足引发。例如：用户错

误及I/O问题----程序试图打开一个并不存在的远程Socket端口。这不是程序本

身的逻辑错误，而很可能是远程机器名字错误(用户拼写错误)。对商用软件系统，

程序开发者必须考虑并处理这个问题。Java编译器强制要求处理这类异常，如果

不捕获这类异常，程序将不能被编译。

运行期异常： 这意味着程序存在bug，如数组越界，0被除，入参不满足规范.....

这类异常需要更改程序来避免，Java编译器强制要求处理这类异常。

**93、==equals区别**

1、 ==是判断两个变量或实例是不是指向同一个内存空间

equals是判断两个变量或实例所指向的内存空间的值是不是相同

2、==是指对内存地址进行比较

equals()是对字符串的内容进行比较

3、==指引用是否相同

equals（）指的是值是否相同

**94 public，private，protect权限**

public在任何情况下都可用；

protect可在当前类，同一个包中和子孙类中使用；

default：可在当前类和同一个包中使用

private只能在当前类中使用

**95、HttpClient与HttpUrlConnection的区别**

后者更为常用。HttpClient是个开源框架，封装了访问http的请求头，参数，内容体，响应等等， HttpURLConnection是java的标准类，什么都没封装，用起来太原始，不方便HttpClient实际上是对Java提供方法的一些封装，在HttpURLConnection中的输入输出流操作，在这个接口中被统一封装成了HttpPost(HttpGet)和HttpResponse，这样，就减少了操作的繁琐性。

**96，json解析方式的两种区别**

1，SDK提供JSONArray，JSONObject

2，google提供的 Gson

通过fromJson()实现对象的反序列化（即将json串转换为对象类型）

通过toJson()实现对象的序列化 （即将对象类型转换为json串）

通过google提供的Gson解析json时，定义JavaBean的规则是什么？

1). 实现序列化 Serializable

2). 属性私有化，并提供get，set方法

3). 提供无参构造

4). 属性名必须与json串中属性名保持一致 （因为Gson解析json串底层用到了Java的反射原理）

**97、什么时候用递归**

树用递归方便，用其它方法好像跟本不可能．

递归用在特定的函数如 f(x)= f(x-1) + 2

像这种情况，你想算出f(x)就必需算出f(x-1)，而f(x)和f(x-1)实际上都是用共一个方法，只是参数相差一．这种时候用递归就很快．

**98、jdbc中怎么处理事物**

事务，也是数据库事务，指的是作为单个逻辑工作单元执行的一系列操作。正常的情况下，操作应该顺利进行，与操作相关的所有数据库信息也成功地更新；

但是，如果在这一系列过程中任何一个环节出了差错，导致操作失败了，数据库中所有信息都必须保持操作前的状态不变。否则，数据库的信息将会一片混乱而不可预测。

一个逻辑工作单元要称为事务，必须满足ACID（原子性，一致性，隔离性和持久性）

事务的结束只能有两种形式：提交和回滚。操作完全成功则提交，产生永久性的修改；操作不完全成功则回滚，恢复到事务开始前的状态。它们将结束一个事务。

（1）关闭自动提交事务。通过设置连接的自动提交事务属性为false，

**99、android时间分发机制**

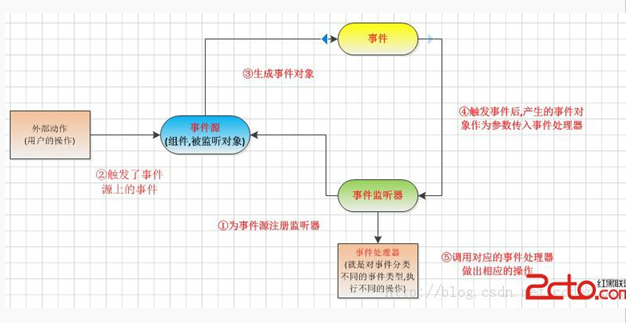
当你点击了某个控件，首先会去调用该控件所在布局的dispatchTouchEvent方法，然后在布局的dispatchTouchEvent方法中找到被点击的相应控件，再去调用该控件的dispatchTouchEvent方法

**100，static内部发出对非static的调用**

不能调用

**基于回调机制的处理以及监听机制的处理**

监听机制的处理模型图：



①直接使用匿名内部类,作为事件监听器

ps:就是我们平时最常用的那种,setXxxListener后就重写里面的对应方法

通常都是临时使用一次,复用性不高

②使用内部类作为事件监听器

使用优点:可以在该类中复用,可直接访问外部类的所有界面组件

③使用外部类作为事件监听器

就是另外创建一个处理事件的[Java](http://www.2cto.com/kf/ware/Java/)文件,该形式比较少见

因为外部类不能直接访问用户界面类中的[组件](http://www.2cto.com/kf/all/zujian/),要通过构造方法将组件传入使用,

这样的结果就是代码不够简洁

④直接使用Activity作为事件监听器

只需要让Activity类实现~Listener事件监听接口,在Activity中定义重写对应的事件处理器方法

<http://www.2cto.com/kf/201403/287434.html>

回调机制的处理：

Android平台中，每个View都有自己的处理事件的回调方法，通过重写View中的这些回调方法来实现需要的响应事件。当某个事件没有被任何一个View处理时，便会调用Activity中相应的回调方法。Android提供了以下回调方法供用户使用：

1. onKeyDown：  
   功能：该方法是接口KeyEvent.Callback中的抽象方法，所有的View全部实现了该接口并重写了该方法，该方法用来捕捉手机键盘被按下的事件。

2. onKeyUp：  
   功能：该方法同样是接口KeyEvent.Callback中的一个抽象方法，并且所有的View同样全部实现了该接口并重写了该方法，onKeyUp方法用来捕捉手机键盘按键抬起的事件。

3. onTouchEvent：

   功能：该方法在View类中的定义，并且所有的View子类全部重写了该方法，应用程序可以通过该方法处理手机屏幕的触摸事件。

４.　onTrackBallEvent：  
　　　功能：　接下来将介绍的是手机中轨迹球的处理方法onTrackBallEvent。所有的View同样全部实现了该方法。

５.　 onFocusChanged：  
　　　功能：　前面介绍的各个方法都可以在View及Activity中重写，接下来介绍的onFocusChanged却只能在View中重写。该方法是焦点改变的回调方法，当某个控件重写了该方法后，当焦点发生变化时，会自动调用该方法来处理焦点改变的事件。

http://blog.csdn.net/liranke/article/details/6855601

**UI中padding和margin的区别**

Padding 用来指定组件内的内容距离组件边界的距离;

Margin用来指定控件与控件之间的距离

**View如何刷新**

实现模式对话框的OnOk()函数，在其中添加下面的代码就可以了：  
（(C\*\*\*View)(AfxGetMainWnd()->GetActiveView())）->Invalidate();  
其中C\*\*\*View替换为你自己的View类的名字。  
当然，调用Invalidate()只是按你的问题要求实现了刷新显示而已，至于如何实现图片大小的更新，仍需要其他代码（比如在View类中添加代表图片大小的成员变量，在OnDraw()中按照成员变量绘制图片，在OnOK()中更新View成员变量的值）

**This和getApplicationContext ()的区别**

getApplicationContext():生命周期是整个应用，应用摧毁，它才摧毁。

this:代表当前,在Activity当中就是代表当前的Activity，换句话说就是Activity.this在Activity当中可以缩写为this.

**横竖屏切换activity周期的变化**

1、新建一个Activity并把各个生命周期打印出来

2、运行Activity，得到如下信息

onCreate-->  
onStart-->  
onResume-->

3、按crtl+f12切换成横屏时

onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->

4、再按crtl+f12切换成竖屏时，打印信息如下

onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->  
onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->

5、修改AndroidManifest.xml，把该Activity添加 android:configChanges="orientation"，执行步骤3

onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->

6、再执行步骤4，发现不会再打印相同信息，但多打印了一行onConfigChanged

onSaveInstanceState-->  
onPause-->  
onStop-->  
onDestroy-->  
onCreate-->  
onStart-->  
onRestoreInstanceState-->  
onResume-->  
onConfigurationChanged-->

7、把步骤5的android:configChanges="orientation" 改成 android:configChanges="orientation|keyboardHidden"，执行步骤3，就只打印onConfigChanged

onConfigurationChanged-->

8、执行步骤4

onConfigurationChanged-->  
onConfigurationChanged-->

总结：

1、不设置Activity的android:configChanges时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次

2、设置Activity的android:configChanges="orientation"时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次

3、设置Activity的android:configChanges="orientation|keyboardHidden"时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法

总结一下整个Activity的生命周期

补充一点，当前Activity产生事件弹出Toast和AlertDialog的时候

ActivityActivity的生命周期不会有改变运行时按下HOME键(跟被完全覆盖是一样的)：onSaveInstanceState --> onPause --> onStop       onRestart -->onStart--->onResume

Activity未被完全覆盖只是失去焦点：onPause--->onResume

**如何实现ViewPager**

先在布局文件中调用v4包下的ViewPager，在java代码中初始化ViewPager，并设置adapter（继承PagerAdapter），再设置监听onPageChangeListener

**Sqlite中的事务如何实现**

开启事务： start transaction   
 sql:…update account set money=money-500 where id=1;--检验隔离性  
 sql…update account set money=money+500 where id=2;   
commit提交-验证原子性、一致性

**Android中常用的菜单以及区别**

1、左右推出的Menu

使用方法：

一.首先导入SlidingMenu这个库文件

二.在你的项目中引入SlidingMenu这个库

eclipse中的流程：右击你的项目 -- Properties -- Android  -- Add  选择SlidingMenu库

三.在你的项目中使用这个SlidingMenu

配置设定方法中 setMode() 是常用属性有：

1）SlidingMenu.LEFT    左测菜单

2）SlidingMenu.RIGHT   右侧菜单

3）SlidingMenu.LEFT\_RIGHT    左右2测菜单

2、Path菜单

3、GridView菜单

4、popuwindow菜单

5、系统菜单

**重载和重写的区别**

方法的重载属于,编译时多态,方法名相同参数列表不同，返回值必须相同或都没有返回值类型。         方法的覆盖属于运行时多态，子类覆盖父类的方法,子类指向父类引用,在调用方法的时候用父类的引用调用。

**Set list map集合的特点**

List接口 元素有序可重复.

实现类有：ArrayList    数组实现轻量级，运行快，线程不安全。JDK1.2 查询快

 Vector   数组实现重量级，运行慢，线程安全。JDK1.0保证元素的无序唯一，自定义对象存进HashSet为了保证元素内容不重复需要覆盖hashCode()与equals()方法。

          SortedSet(不重要) 元素有序（Unicode升序）唯一

  LinkedList  链表实现 常用语堆栈与队列的实现 增删操作快

Set 接口 元素无序不可重复

实现类有：HashSet，底层用hashCode()算法实现，

  TreeSet要求元素有序，自定义的对象需要实

现Comparable接口的 compareTo（object o）方法

 Map(接口): 与Collection接口无关,有一个子接口SortedMap特点: 元素是key-value, key

 唯一,无序; value可重复

实现类: HashMap 轻量级 线程不安全的,允许key或value为null JDK1.2

        HashTable 重量级 线程安全的 不允许key或value为null   JDK1.0

       Properties是HashTable的子类,主键和值都是字符串

SortedMap:(不重要)

特点: key唯一,有序(Unicode升序)

实现类:TreeMap

**gradView上拉加载和下拉刷新怎么实现**

实现OnScrollListener 接口重写onScrollStateChanged 和onScroll方法，使用onscroll方法实现”滑动“后处理检查是否还有新的记录，如果有，调用 addFooterView，添加记录到adapter, adapter调用 notifyDataSetChanged 更新数据;如果没有记录了，把自定义的mFooterView去掉。使用onScrollStateChanged可以检测是否滚到最后一行且停止滚动然后执行加载

**用过哪些框架**

XUtils AFinal 第三方开源框架

**横向scrollView和ViewPager之间的冲突**

重写ScrollView，

publicclassHomeFeatureLayoutextendsHorizontalScrollView {

privateArrayList<ListItem> items = null;

privateGestureDetector gestureDetector;

View.OnTouchListener gestureListener;

privatestaticfinalint SWIPE\_MIN\_DISTANCE = 5;

privatestaticfinalint SWIPE\_THRESHOLD\_VELOCITY = 300;

privateint activeFeature = 0;

publicHomeFeatureLayout(Context context, ArrayList<ListItem> items){

super(context);

setLayoutParams(newLayoutParams(LayoutParams.FILL\_PARENT, LayoutParams.WRAP\_CONTENT));

setFadingEdgeLength(0);

this.setHorizontalScrollBarEnabled(false);

this.setVerticalScrollBarEnabled(false);

LinearLayout internalWrapper = newLinearLayout(context);

internalWrapper.setLayoutParams(newLayoutParams(LayoutParams.FILL\_PARENT, LayoutParams.FILL\_PARENT));

internalWrapper.setOrientation(LinearLayout.HORIZONTAL);

addView(internalWrapper);

this.items = items;

for(int i = 0; i< items.size();i++){

LinearLayout featureLayout = (LinearLayout) View.inflate(this.getContext(),R.layout.homefeature,null);

TextView header = (TextView) featureLayout.findViewById(R.id.featureheader);

ImageView image = (ImageView) featureLayout.findViewById(R.id.featureimage);

TextView title = (TextView) featureLayout.findViewById(R.id.featuretitle);

title.setTag(items.get(i).GetLinkURL());

TextView date = (TextView) featureLayout.findViewById(R.id.featuredate);

header.setText("FEATURED");

Image cachedImage = newImage(this.getContext(), items.get(i).GetImageURL());

image.setImageDrawable(cachedImage.getImage());

title.setText(items.get(i).GetTitle());

date.setText(items.get(i).GetDate());

internalWrapper.addView(featureLayout);

}

gestureDetector = newGestureDetector(newMyGestureDetector());

setOnTouchListener(newView.OnTouchListener() {

@Override

publicboolean onTouch(View v, MotionEventevent) {

if (gestureDetector.onTouchEvent(event)) {

returntrue;

}

elseif(event.getAction() == MotionEvent.ACTION\_UP || event.getAction() == MotionEvent.ACTION\_CANCEL ){

int scrollX = getScrollX();

int featureWidth = getMeasuredWidth();

activeFeature = ((scrollX + (featureWidth/2))/featureWidth);

int scrollTo = activeFeature\*featureWidth;

smoothScrollTo(scrollTo, 0);

returntrue;

}

else{

returnfalse;

}

}

});

}

classMyGestureDetectorextendsSimpleOnGestureListener {

@Override

publicboolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {

try {

//right to left

if(e1.getX() - e2.getX() > SWIPE\_MIN\_DISTANCE &&Math.abs(velocityX) > SWIPE\_THRESHOLD\_VELOCITY) {

activeFeature = (activeFeature < (items.size() - 1))? activeFeature + 1:items.size() -1;

smoothScrollTo(activeFeature\*getMeasuredWidth(), 0);

returntrue;

}

//left to right

elseif (e2.getX() - e1.getX() > SWIPE\_MIN\_DISTANCE &&Math.abs(velocityX) > SWIPE\_THRESHOLD\_VELOCITY) {

activeFeature = (activeFeature >0)? activeFeature - 1:0;

smoothScrollTo(activeFeature\*getMeasuredWidth(), 0);

returntrue;

}

} catch (Exception e) {

// nothing

}

returnfalse;

}

}

**集合和数组的区别**

1）：数组在定义时必须定义长度，而集合在定义时不必定义长度。

2）：数组在定义时必须要声明数组的数据类型，而集合不必。但是一般情况下我们都是存储统一数据类型的数据，我们可以使用泛型的写法来限制集合里面的数据类型。

**什么方法不能被重写**

被final修饰的方法，构造方法，

**编译时异常和运行时异常的区别**

异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

**Gps定位和网络定位的区别**

GPS定位需设置权限android.permission.ACCESS\_LOCATION\_FINE

网络定位需设置权限android.permission.ACCESS\_LOCATION\_COARSE

即GPS定位比网络定位在精确度方面更精准，但在速度方面稍差一些，而且GPS定位适合在室外环境下使用，网络定位在室内，室外都可以。

**Collection 和 Collections的区别**Collection是集合类的上级接口，继承与他的接口主要有Set 和List  
Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

Native是啥

**Service实现的两种方式**

1.Context.startService()方式启动

①Context.startService()方式的生命周期： 启动时，startService –> onCreate() –> onStart()停止时，stopService –> onDestroy()如果调用者直接退出而没有停止Service，则Service 会一直在后台运行 Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，接着调用onStart()方法。如果调用startService()方法前服务已经被创建，多次调用startService()方法并不会导致多次创建服务，但会导致多次调用onStart()方法。采用startService()方法启动的服务，只能调用Context.stopService()方法结束服务，服务结束时会调用onDestroy()方法附代码

2.Context.bindService()方式启动：①Context.bindService()方式的生命周期： 绑定时,bindService -> onCreate() –> onBind()调用者退出了，即解绑定时,Srevice就会unbindService –>onUnbind() –> onDestory()Context.bindService()方式启动 Service的方法：绑定Service需要三个参数：bindService(intent, conn, Service.BIND\_AUTO\_CREATE);第一个：Intent对象第二个：ServiceConnection对象，创建该对象要实现它的onServiceConnected()和 onServiceDisconnected()来判断连接成功或者是断开连接第三个：如何创建Service，一般指定绑定的时候自动创建附代码

**手机上使用的传感器有什么**

#define SENSOR\_TYPE\_ACCELEROMETER       1 //加速度

#define SENSOR\_TYPE\_MAGNETIC\_FIELD      2 //磁力

#define SENSOR\_TYPE\_ORIENTATION         3 //方向

#define SENSOR\_TYPE\_GYROSCOPE           4 //陀螺仪

#define SENSOR\_TYPE\_LIGHT               5 //光线感应

#define SENSOR\_TYPE\_PRESSURE            6 //压力

#define SENSOR\_TYPE\_TEMPERATURE         7 //温度

#define SENSOR\_TYPE\_PROXIMITY           8 //接近

#define SENSOR\_TYPE\_GRAVITY             9 //重力

#define SENSOR\_TYPE\_LINEAR\_ACCELERATION 10//线性加速度

#define SENSOR\_TYPE\_ROTATION\_VECTOR     11//旋转矢量

**SurfaceView实现的原理**

SurfaceView，它拥有独立的绘图表面，即它不与其宿主窗口共享同一个绘图表面。由于拥有独立的绘图表面，因此SurfaceView的UI就可以在一个独立的线程中进行行绘制。又由于不占用主线程资源，SurfaceView一方面可以实现复杂而高效的UI，另一方面又不会导致用户输入得不到及时响应。

**Int Integer的区别**

 int 是基本类型，直接存数值

　　1.Java 中的数据类型分为基本数据类型和复杂数据类型

　　int 是前者>>integer 是后者（也就是一个类）

　　2.初始化时>>

　　int i =1;

　　Integer i= new Integer(1);(要把integer 当做一个类看)

　　int 是基本数据类型（面向过程留下的痕迹，不过是对java的有益补充）

　　Integer 是一个类，是int的扩展，定义了很多的转换方法

类似的还有：float Float;double Double;string String等

**Application类的作用**

Application和Actovotu,Service一样是android框架的一个系统组件，当android程序启动时系统会创建一个 application对象，用来存储系统的一些信息。通常我们是不需要指定一个Application的，这时系统会自动帮我们创建，如果需要创建自己 的Application，也很简单创建一个类继承 Application并在manifest的application标签中进行注册(只需要给Application标签增加个name属性把自己的 Application的名字定入即可)。

android系统会为每个程序运行时创建一个Application类的对象且仅创建一个，所以Application可以说是单例 (singleton)模式的一个类.且application对象的生命周期是整个程序中最长的，它的生命周期就等于这个程序的生命周期。因为它是全局 的单例的，所以在不同的Activity,Service中获得的对象都是同一个对象。所以通过Application来进行一些，数据传递，数据共享 等,数据缓存等操作。

**嵌入式操作系统**

嵌入式实时操作系统是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统作出快速响应，并控制所有实时任务协调一致运行的嵌入式操作系统。主要用于工业控制、 军事设备、 航空航天等领域对系统的响应时间有苛刻的要求，这就需要使用实时系统。又可分为软实时和硬实时两种，而android是基于linux内核的，因此属于软实时。

**ViewFlipper和ViewPager的区别**

1.ViewFilpper类继承于ViewAnimator类。而ViewAnimator类继承于FrameLayout。

查看ViewAnimator类的源码可以看出此类的作用主要是为其中的View切换提供动画效果。该类有如下几个和动画相关的方法。

2.ViewPager用于实现多页面的切换效果，该类存在于Google的兼容包里面，所以在引用时记得在BuilldPath中加入“android-support-v4.jar”

**定义数组的几种方式**

nt a[]; //声明未初始化  
a = new int [10]; // 定义占用空间大小（10个int）  
int a[] = new int [10]; //声明并定义大小（即分配了指定大小的空间）  
int a[] = {1,2,3}; // 声明并初始化，占用空间大小是3个int。

**知道哪几种排序方式**

冒泡

public static int[] doMaoPao(int[] in){

int temp;

int length = in.length;

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

for(int j=0;j<length-1-i;j++){

if(in[j]<in[j+1]){

temp = in[j+1];

in[j+1] = in[j];

in[j] = temp;

}

System.out.print("输出j="+in[j]+"");

System.out.println("");

System.out.print("输出j+1= "+in[j+1]);

}

// System.out.print("i="+in[i]);

}

return in;

}

选择

public static int[] doPaiXv(int[] in){

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

int temp = 0;

for(int j=i+1;j<in.length;j++){

if(in[i]>in[j]){

temp = in[i];

in[i] = in[j];

in[j] = temp;

}

System.out.print("输出j="+in[j]+"");

System.out.println("");

System.out.print("输出i= "+in[i]);

}

}

return in;

}

二分

public static void binary\_sort(int a[]){

int i,j,temp;

int left,right,mid;

for(i = 1; i<a.length; i++){

left = 0;

right = i-1;

temp = a[i];

while(left <= right){

mid = (left + right) /2;

if(a[mid] > temp){

right = mid - 1;

}else{

left = mid + 1;

}

}

for(j=i-1; j>right; j--){

a[j+1] = a[j];

a[right+1] = temp;

}

}

}

public static void main(String[] args) {

int a[] = {2,6,9,8,4,7,3,1,0,5};

binary\_sort(a);

for(int i=0;i<a.length;i++)

System.out.print(a[i]+ "");

}

**Doget和doPost的区别**

通过get方式提交的数据有大小的限制，通常在1024字节左右。也就是说如果提交的数据很大，用get方法就可需要小心；而post方式没有数据大小的限制，理论上传送多少数据都可以。

通过get传递数据，实际上是将传递的数据按照”key，value”的方式跟在URL的后面来达到传送的目的的；而post传递数据是通过http请求的附件进行的，在URL中并没有明文显示

**什么是API**

API（Application Programming Interface,应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节.  
简单来说,就是一套可供调用的系统接口.

**常遇到的异常、如何解决**

异常的继承结构：

基类为Throwable，Error和Exception继承Throwable，RuntimeException和IOException等继承Exception，具体的RuntimeException继承RuntimeException。

Error和RuntimeException及其子类成为未检查异常（unchecked），其它异常成为已检查异常（checked）。

1、 SQLException：操作数据库异常类。

问题：当Java应用程序与数据库进行沟通时如果产生了错误，就会触发这个类。同时会将数据库的错误信息通过这个类显示给用户。当用户插入数据的时候，如果这个字段的值跟现有的纪录重复了，违反了数据库的唯一性约束，此时数据库就会跑出一个异常信息。这个信息一般用户可能看不到，因为其发生在数据库层面的。此时这个操作数据库异常类就会捕捉到数据库的这个异常信息，并将这个异常信息传递到前台。如此的话，前台用户就可以根据这个异常信息来分析发生错误的原因。

解决方案：检查插入的数据是否重复。  
 2、 ClassCastException：数据类型转换异常。

问题：在Java应用程序中，有时候需要对数据类型进行转换。这个转换包括显示的转换与隐式的转换。不过无论怎么转换，都必须要符合一个前提的条件，即数据类型的兼容性。在数据类型进行转换之前，就保证数据类型的兼容性。如此的话，就不容易造成数据类型的转换异常。如在只允许数值类型的字段中，可以设置不允许用户输入数值以外的字符。

解决方案：注意相应的组件在初始化的时候，是否相对应。  
 3、 NumberFormatException：字符串转换为数字类型时抛出的异常。  
  
在数据类型转换过程中，如果是字符型转换为数字型过程中出现的问题，对于这个异常在Java程序中采用了一个独立的异常，即NumberFormatException.如现在讲字符型的数据“123456”转换为数值型数据时，是允许的。但是如果字符型数据中包含了非数字型的字符，如123#56，此时转换为数值型时就会出现异常。系统就会捕捉到这个异常，并进行处理。  
解决方案：检查数据是否正确。

4. java.lang.NullPointerException  
(1).如果使用第三包的jar包，jar包中有错误

(2).没有实例化组件

(3).没有把Adapter和值绑定；

(4)．当页面中某个组件赋值为null时activity会出现错误，程序崩溃后其他页面的数据就会被重新初始化

(5).用Intent传值时，获取的key不存在

(6).没有赋初始值

5. java.lang.ClassNotFoundException　　异常的解释是"指定的类不存在"。  
6. java.lang.ArithmeticException　　这个异常的解释是"数学运算异常"，比如程序中出现了除以零这样的运算就会出这样的异常。  
7. java.lang.ArrayIndexOutOfBoundsException  
　　异常的解释是"数组下标越界"，现在程序中大多都有对数组的操作，因此在调用数组的时候一定要认真检查，看自己调用的下标是不是超出了数组的范围，一般来说，显示（即直接用常数当下标）调用不太容易出这样的错，但隐式（即用变量表示下标）调用就经常出错了，还有一种情况，是程序中定义的数组的长度是通过某些特定方法决定的，不是事先声明的，这个时候，最好先查看一下数组的length，以免出现这个异常。  
8. java.lang.IllegalArgumentException

这个异常的解释是"方法的参数错误"，比如g.setColor(int red,int green,int blue)这个方法中的三个值，如果有超过２５５的也会出现这个异常，因此一旦发现这个异常，我们要做的，就是赶紧去检查一下方法调用中的参数传递是不是出现了错误。  
9. java.lang.IllegalAccessException  
　　这个异常的解释是"没有访问权限"，当应用程序要调用一个类，但当前的方法即没有对该类的访问权限便会出现这个异常。对程序中用了Package的情况下要注意这个异常

10.安全异常：产生的原因是由于当前的设备不支持当前程序，这个是由于机型产生的问题，我们应该换一个设备进行测试，检测当前程序存在什么样的异常；另一种原因是获取激活的网络信息实体类，需要添加权限，否则会出现此异常

解决办法：在配置文件中，添加android\_permission\_Access\_NetWork\_State的权限

11. RejectExcuteException：一个异步任务只能执行一次，否则会报异常

一个线程最多能执行5个线程，超出则等待，否则会报拒绝执行的异常

解决办法：使用子线程+handler来代替AsyncTask，或者一个线程中少添加几个异步操作。

12. Out of memory：内存溢出是存放的数据过多导致内存不足的溢出事件

13. Memory leak：内存泄露是本应该释放的资源没有充分得到释放，就继续添加数据导致内存泄露事件；

Java应用程序中常见的异常类还有很多。如未找到相应类异常、不允许访问某些类异常、文件已经结束异常、文件未找到异常、字段未找到异常等等。一般系统开发人员都可以根据这个异常名来判断当前异常的类型。

ArithmeticException（除数为0的异常）, BufferOverflowException（缓冲区上溢异常）, BufferUnderflowException（缓冲区下溢异常）, IndexOutOfBoundsException（出界异常）, NullPointerException（空指针异常）, EmptyStackException（空栈异常）, IllegalArgumentException（不合法的参数异常）, NegativeArraySizeException, NoSuchElementException, SecurityException, SystemException, UndeclaredThrowableException

**常用的推送，并简单介绍**

并行的都包括：即时信息和空间协议(IMPP)、空间和即时信息协议(PRIM)；

推送的实现原理：

XMPP是一种基于XML的协议，它继承了在XML环境中灵活的发展性。因此，基于XMPP的应用具有超强的可扩展性。XMPP的前身是Jabber，一个开源形式组织产生的网络即时通信协议。所有从一个client到另一个client的jabber消息和数据都要xmpp server，client连接到server，server利用本地目录系统的证书对其认证，client制定目标地址，让server告知目标状态，server查找，连接并进行相互认证，client间进行交互。

XMPP系统特点：

1)客户机/服务器通信模式；(2)分布式网络；(3)简单的客户端；(4)XML的数据格式。

通俗解释：

其实XMPP 是一种很类似于http协议的一种数据传输协议，它的过程就如同“解包装--〉包

装”的过程，用户只需要明白它接受的类型，并理解它返回的类型，就可以很好的利用xmpp来进行数据通讯。

**服务器提供数据前段实现容易出现什么问题**

接口不可用，接口提供的数据格式不是我们想要的。

**133．如何判断那种网络**

1.网络是否连接（包括Wifi和移动网络）

// 是否有可用网络

**private** **boolean** isNetworkConnected() {

        ConnectivityManager cm =

                (ConnectivityManager) mContext.getSystemService(Context.CONNECTIVITY\_SERVICE);

        NetworkInfo network = cm.getActiveNetworkInfo();

**if** (network != **null**) {

**return** network.isAvailable();

        }

**return** **false**;

    }

2.wifi是否可用

// Wifi是否可用

**private** **boolean** isWifiEnable() {

        WifiManager wifiManager = (WifiManager) mContext

                .getSystemService(Context.WIFI\_SERVICE);

**return** wifiManager.isWifiEnabled();

    }

**GPS是否可用**

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/sky837/article/details/7867601)

// Gps是否可用

**private** **boolean** isGpsEnable() {

        LocationManager locationManager =

                ((LocationManager) mContext.getSystemService(Context.LOCATION\_SERVICE));

**return** locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER);

}

**抽象类和接口的区别**

abstract可以修饰抽象方法，而一个类只要有一个抽象方法，就必须用abstract定义该类，即抽象类。抽象类，被继承，实现它里面的方法，可以只有普通方法

用interface修饰的类，里面的方法都是抽象方法（不能实例化），因此在定义接口的时候，可以直接不加那些修饰，系统会默认的添上去。接口里面的字段都是公有常量，即public static final修

饰的字段。可以没有函数原型，里面可以是空的。

**135. Android中版本代号**

Android 1.0（没有开发代号）

Android 1.1 - Petit Four

Android 1.5 - Cupcake

Android 1.6 - Donut

Android 2.0/2.1 - Éclair

Android 2.2 - Froyo

Android 2.3 - Gingerbread

Android 3.0/3.1/3.2 - Honeycomb

Android 4.0 - Ice Cream Sandwich

Android 4.1/4.2/4.3 - Jelly Bean

Android 4.4 - KitKat

Android 5.0/5.1 – Lollipop

**136. Final和Finally的区别**

final用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

**136. 分享平台有哪些**

ShardSdk、友盟等。

**137. 后台Activity被回收如何恢复**

重写onSaveInstanceState()方法，在此方法中保存需要保存的数据，该方法将会在activity被回收之前调用。通过重写onRestoreInstanceState()方法可以从中提取保存好的数据

**138.Android事件分发的理解**

**事件分发：public boolean dispatchTouchEvent(MotionEvent ev)**

Touch 事件发生时 Activity 的 dispatchTouchEvent(MotionEvent ev) 方法会以隧道方式（从根元素依次往下传递直到最内层子元素或在中间某一元素中由于某一条件停止传递）将事件传递给最外层 View 的 dispatchTouchEvent(MotionEvent ev) 方法，并由该 View 的 dispatchTouchEvent(MotionEvent ev) 方法对事件进行分发。dispatchTouchEvent 的事件分发逻辑如下：

如果 **return true**，事件会分发给当前 View 并由 dispatchTouchEvent 方法进行消费，同时事件会停止向下传递；

如果 **return false**，事件分发分为两种情况：

如果当前 View 获取的事件直接来自 Activity，则会将事件返回给 Activity 的 onTouchEvent 进行消费；

如果当前 View 获取的事件来自外层父控件，则会将事件返回给父 View 的  onTouchEvent 进行消费。

如果返回系统默认的 **super**.dispatchTouchEvent(ev)，事件会自动的分发给当前 View 的 onInterceptTouchEvent 方法。

**▐ 事件拦截：public boolean onInterceptTouchEvent(MotionEvent ev)**

在外层 View 的 dispatchTouchEvent(MotionEvent ev) 方法返回系统默认的 super.dispatchTouchEvent(ev) 情况下，事件会自动的分发给当前 View 的 onInterceptTouchEvent 方法。onInterceptTouchEvent 的事件拦截逻辑如下：

如果 onInterceptTouchEvent 返回 **true**，则表示将事件进行拦截，并将拦截到的事件交由当前 View 的 onTouchEvent 进行处理；

如果 onInterceptTouchEvent 返回 **false**，则表示将事件放行，当前 View 上的事件会被传递到子 View 上，再由子 View 的 dispatchTouchEvent 来开始这个事件的分发；

如果 onInterceptTouchEvent 返回 **super**.onInterceptTouchEvent(ev)，事件默认会被拦截，并将拦截到的事件交由当前 View 的 onTouchEvent 进行处理。

**▐ 事件响应：public boolean onTouchEvent(MotionEvent ev)**

在 dispatchTouchEvent 返回 super.dispatchTouchEvent(ev) 并且 onInterceptTouchEvent 返回 true 或返回 super.onInterceptTouchEvent(ev) 的情况下 onTouchEvent 会被调用。onTouchEvent 的事件响应逻辑如下：

如果事件传递到当前 View 的 onTouchEvent 方法，而该方法返回了 false，那么这个事件会从当前 View 向上传递，并且都是由上层 View 的 onTouchEvent 来接收，如果传递到上面的 onTouchEvent 也返回 false，这个事件就会“消失”，而且接收不到下一次事件。

如果返回了 true 则会接收并消费该事件。

如果返回 super.onTouchEvent(ev) 默认处理事件的逻辑和返回 false 时相同。

到这里，与 Touch 事件相关的三个方法就分析完毕了。

**139. 线程的几种状态**

1)新建状态(New)：新创建了一个线程对象。

　2)就绪状态(Runnable)：线程对象创建后，其他线程调用了该对象的start()方法。该状态的线程位于可运行线程池中，变得可运行，等待获取[CPU](http://product.it168.com/list/b/0217_1.shtml)的使用权。

　　3)运行状态(Running)：就绪状态的线程获取了CPU，执行run()方法。

　　4)阻塞状态(Blocked)：阻塞状态是线程因为某种原因放弃CPU使用权，暂时停止运行。直到线程进入就绪状态，才有机会转到运行状态。阻塞的情况分三种：

等待阻塞：运行的线程执行wait()方法，JVM会把该线程放入等待池中。

同步阻塞：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池中。

其他阻塞：运行的线程执行sleep()或join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪状态。

5)死亡状态(Dead)：线程执行完了或者因异常退出了run()方法，该线程结束生命周期。

当调用start方法的时候，该线程就进入就绪状态。等待CPU进行调度执行，此时还没有真正执行线程。

**140. Ddms和Traceview的区别**

DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器。

**150. PathImage是什么**

.9.PNG是安卓开发里面的一种特殊的图片，这种格式的图片通过ADT自带的编辑工具生成，使用九宫格切分的方法，使图片支持在android 环境下的自适应展示。

**151. Android项目的目录结构**

1)src目录：src（即SourceCode)，src目录用来存放应用程序中所有的源代码，代码的源文件一般存放在该目录下的相应的包下。

2)gen目录：gen目录，该目录下一般只有一个文件，即R文件。该文件夹下面有个R.java文件，R.java是在建立项目时自动生成的，定义该项目所有资源的索引。

3)Android4.1目录：该文件夹下包含android.jar文件，这是一个Java归档文件，其中包含构建应用程序所需的所有的AndroidSDK库（如Views、Controls）和APIs。

通过android.jar将自己的应用程序绑定到AndroidSDK和AndroidEmulator，这允许你使用所有Android的库和包，且使你的应用程序在适当的环境中调试。

4)assets目录：该目录存放应用程序中使用的外部资源文件，如声音视频等，在Android程序中可以通过输入/输出流对该目录中的文件进行读写。

5)res目录：该目录用来存放程序这用到的图片、界面布局文件及XML格式的描述文件。该目录下有多个目录。新建一Android项目，在res目录下一般会有drawable,layout,valuse三个目录。drawable用来存放图片资源,目录下一般会有三个目录分别存放高中低三种不同分辨率的图片，layout目录用来存放应用程序的界面布局文件，values目录下存放字符串资源，颜色资源，数组资源等XML文件。

6)AndroidManifest.xml

**152. Intentservice的用法**

androidservicethreadstringclassnullIntentService实现原理及内部代码：

Android中除了Service还有一个IntentService，他们之间到底有哪些区别呢?在继承关系上而言IntentService是Service的子类，内部实现的代码中涉及到一些Android入门开发者不了解的Looper，Android123在早期的文章中已经说明他们的用法，这里不再赘述，有关原理大家可以看源码实现如下:

1. publicabstractclassIntentServiceextendsService{
2. privatevolatileLoopermServiceLooper;
3. privatevolatileServiceHandlermServiceHandler;//一个Handler封装了Looper对象
4. privateStringmName;
5. privatebooleanmRedelivery;
6. privatefinalclassServiceHandlerextendsHandler{
7. publicServiceHandler(Looperlooper){
8. super(looper);
9. }
10. @Override
11. publicvoidhandleMessage(Messagemsg){
12. onHandleIntent((Intent)msg.obj);
13. stopSelf(msg.arg1);
14. }
15. }
16. publicIntentService(Stringname){//构造方法，需要提供一个name作为标识
17. super();
18. mName=name;
19. }
20. 对于下面的setIntentRedelivery的参数，如果为真时
21. onStartCommand(Intent,int,int)}willreturn
22. Service#START\_REDELIVER\_INTENT},soifthisprocessdiesbefore
23. onHandleIntent(Intent)}returns,theprocesswillberestarted
24. 如果为假时
25. onStartCommand(Intent,int,int)}willreturn
26. Service#START\_NOT\_STICKY},andiftheprocessdies
27. publicvoidsetIntentRedelivery(booleanenabled){
28. mRedelivery=enabled;
29. }
30. @Override
31. publicvoidonCreate(){//这里重写父类Service的创建，主要是构造一个线程
32. super.onCreate();
33. HandlerThreadthread=newHandlerThread("IntentService["+mName+"]");
34. thread.start();
35. mServiceLooper=thread.getLooper();
36. mServiceHandler=newServiceHandler(mServiceLooper);
37. }
38. @Override
39. publicvoidonStart(Intentintent,intstartId){//Android2.0以前的Service启动参数控制
40. Messagemsg=mServiceHandler.obtainMessage();
41. msg.arg1=startId;
42. msg.obj=intent;
43. mServiceHandler.sendMessage(msg);
44. }
45. @Override
46. publicintonStartCommand(Intentintent,intflags,intstartId){//Android2.0以后的服务启动参数
47. onStart(intent,startId);
48. returnmRedelivery?START\_REDELIVER\_INTENT:START\_NOT\_STICKY;
49. }
50. @Override
51. publicvoidonDestroy(){//服务摧毁的时候Looper一定要释放掉，这点很重要。
52. mServiceLooper.quit();
53. }
54. @Override
55. publicIBinderonBind(Intentintent){
56. returnnull;
57. }
58. protectedabstractvoidonHandleIntent(Intentintent);
59. }
60. 从上面的代码相信大家可以看出IntentService和Service的不同了，通过Looper和Thread来解决标准Service中处理逻辑的阻塞问题，毕竟Android的Service也是会阻塞的。
61. 关于使用
62. IntentService是Service类的子类，用来处理异步请求。客户端可以通过startService(Intent)方法传递请求给IntentService，IntentService通过workerthread处理每个Intent对象，执行完所有的工作之后自动停止Service。
63. 说明：workerthread处理所有通过传递过来的请求，创建一个workerqueue，一次只传递一个intent到onHandleIntent中，从而不必担心多线程带来的问题。处理完毕之后自动调用stopSelf()方法；默认实现了Onbind（）方法，返回值为null；
64. 模式实现了哦你StartCommand()方法，这个方法会放到workerqueue中，然后在onHandleIntent()中执行0
65. 使用IntentService需要两个步骤：
66. 1、写构造函数
67. 2、复写onHandleIntent()方法
68. 好处：处理异步请求的时候可以减少写代码的工作量，比较轻松地实现项目的需求
69. 用Service来处理后台耗时操作，却很少注意到还有个IntentService
70. 首先IntentService是继承自Service的，那我们先看看Service的官方介绍，这里列出两点比较重要的地方：
71. 1.AServiceisnotaseparateprocess.TheServiceobjectitselfdoesnotimplyitisrunninginitsownprocess;unlessotherwisespecified,itrunsinthesameprocessastheapplicationitispartof.
72. 2.AServiceisnotathread.Itisnotameansitselftodoworkoffofthemainthread(toavoidApplicationNotRespondingerrors).
73. 稍微翻一下(英文水平一般)
74. 1.Service不是一个单独的进程，它和应用程序在同一个进程中。
75. 2.Service不是一个线程，所以我们应该避免在Service里面进行耗时的操作
76. 关于第二点我想说下，不知道很多网上的文章都把耗时的操作直接放在Service的onStart方法中，而且没有强调这样会出现ApplicationNotResponding！希望我的文章能帮大家认清这个误区(Service不是一个线程，不能直接处理耗时的操作)。
77. 有人肯定会问，那么为什么我不直接用Thread而要用Service呢？关于这个，大家可以网上搜搜，这里不过多解释。有一点需要强调，如果有耗时操作在Service里，就必须开启一个单独的线程来处理！！！这点一定要铭记在心。
78. IntentService相对于Service来说，有几个非常有用的优点，首先我们看看官方文档的说明：
79. IntentServiceisabaseclassforServicesthathandleasynchronousrequests(expressedasIntents)ondemand.ClientssendrequeststhroughstartService(Intent)calls;theserviceisstartedasneeded,handleseachIntentinturnusingaworkerthread,andstopsitselfwhenitrunsoutofwork.
80. This"workqueueprocessor"patterniscommonlyusedtooffloadtasksfromanapplication'smainthread.TheIntentServiceclassexiststosimplifythispatternandtakecareofthemechanics.Touseit,extendIntentServiceandimplementonHandleIntent(Intent).IntentServicewillreceivetheIntents,launchaworkerthread,andstoptheserviceasappropriate.
81. Allrequestsarehandledonasingleworkerthread--theymaytakeaslongasnecessary(andwillnotblocktheapplication'smainloop),butonlyonerequestwillbeprocessedatatime.
82. 稍微翻译理一理，这里主要是说IntentService使用队列的方式将请求的Intent加入队列，然后开启一个workerthread(线程)来处理队列中的Intent，对于异步的startService请求，IntentService会处理完成一个之后再处理第二个，每一个请求都会在一个单独的workerthread中处理，不会阻塞应用程序的主线程，这里就给我们提供了一个思路，如果有耗时的操作与其在Service里面开启新线程还不如使用IntentService来处理耗时操作。下面给一个小例子：
83. 1Service：
84. packagecom.zhf.service;
85. importAndroid.app.Service;
86. importAndroid.content.Intent;
87. importAndroid.os.IBinder;
88. publicclassMyServiceextendsService{
89. @Override
90. publicvoidonCreate(){
91. super.onCreate();
92. }
93. @Override
94. publicvoidonStart(Intentintent,intstartId){
95. super.onStart(intent,startId);
96. //经测试，Service里面是不能进行耗时的操作的，必须要手动开启一个工作线程来处理耗时操作
97. System.out.println("onStart");
98. try{
99. Thread.sleep(20000);
100. }catch(InterruptedExceptione){
101. e.printStackTrace();
102. }
103. System.out.println("睡眠结束");
104. }
105. @Override
106. publicIBinderonBind(Intentintent){
107. returnnull;
108. }
109. }
110. 2.IntentService：
111. packagecom.zhf.service;
112. importAndroid.app.IntentService;
113. importAndroid.content.Intent;
114. publicclassMyIntentServiceextendsIntentService{
115. publicMyIntentService(){
116. super("yyyyyyyyyyy");
117. }
118. @Override
119. protectedvoidonHandleIntent(Intentintent){
120. //经测试，IntentService里面是可以进行耗时的操作的
121. //IntentService使用队列的方式将请求的Intent加入队列，然后开启一个workerthread(线程)来处理队列中的Intent
122. //对于异步的startService请求，IntentService会处理完成一个之后再处理第二个
123. System.out.println("onStart");
124. try{
125. Thread.sleep(20000);
126. }catch(InterruptedExceptione){
127. e.printStackTrace();
128. }
129. System.out.println("睡眠结束");
130. }
131. }
132. 测试主程序：
133. packagecom.zhf.service;
134. importAndroid.app.Activity;
135. importAndroid.content.Intent;
136. importAndroid.os.Bundle;
137. publicclassServiceDemoActivityextendsActivity{
138. /\*\*Calledwhentheactivityisfirstcreated.\*/
139. @Override
140. publicvoidonCreate(BundlesavedInstanceState){
141. super.onCreate(savedInstanceState);
142. setContentView(R.layout.main);
143. startService(newIntent(this,MyService.class));//主界面阻塞，最终会出现Applicationnotresponding
144. //连续两次启动IntentService，会发现应用程序不会阻塞，而且最重的是第二次的请求会再第一个请求结束之后运行(这个证实了IntentService采用单独的线程每次只从队列中拿出一个请求进行处理)
145. startService(newIntent(this,MyIntentService.class));
146. startService(newIntent(this,MyIntentService.class));
147. }
148. }

**153. IntentService有何优点?**

答案：IntentService使用队列的方式将请求的Intent加入队列，然后开启一个worker thread(线程)来处理队列中的Intent，对于异步的startService请求，IntentService会处理完成一个之后再处理第二个，每一个请求都会在一个单独的worker thread中处理，不会阻塞应用程序的主线程，这里就给我们提供了一个思路，如果有耗时的操作与其在Service里面开启新线程还不如使用IntentService来处理耗时操作。

**154. 如何启用service，如何停用service？**

答案：①：通过调用Context.startservice()启动，调用Context.stopservice()结束，startservice()可以传递参数给service.

②：通过调用context.bindservice()启动，调用Context.unbindservice()结束，还可以通过serviceConnection访问service。

**155. 解释下单线程中message、handler、Message Queue、Looper之间的关系。**

答案：简单来说，Handler获取当前线程中的looper对象，looper用来从存放Message说的MessageQueue中获取Message，再有Handler进行Message的分发和处理。

**156. AIDL全称是什么？能处理哪些数据？**

答案：全称是：Android Interface Definition Landuage是一种街口描述语言；编译器可以通过aidl文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程跨界对象访问的目的，aidl的IPC的机制和COM或CORBA类似，是基于接口的，是轻量级的。使用代理类在客户端和实现层间接传递值。

**157. Activity的启动模式有哪些？**

答案：①、standard，每次激活activity时，都创建activity实例，并放入任务栈。

②、singleTop，如果activity自己激活自己，即任务栈栈顶就是该activity，则不需要创建，其余情况都要创建activity实例。

③、singleTask，如果要激活的那个activity在任务栈中存在该实例，则不需要创建，只需要把此activity放入栈顶，并把该activity以上的activity实例都pop；

④、singleInstance，如果应用1的任务栈中创建了MainActivity实例，如果应用2也要激活MainActivity，则不需要创建，两应用共享该Activity实例；

**158. 如何评价Android？**

Android平台的五大优势：

开放性、挣脱运营商的束缚、丰富的硬件选择、不受任何限制的开发商、无缝结合的Google应用

Android平台的五大不足：

1. 安全和隐私
2. 首先开卖Android手机的不是最大运营商
3. 运营商仍然能够影响到Android手机
4. 同类机型用户减少
5. 过分依赖开发商缺少标准配置

**159. 断点续传**

断点续传下载，其实就是记录下原来下载的位置，然后从那个位置开始下载就是了。

**160. Dvm是什么**

Android 运行环境主要指的虚拟机技术——Dalvik。Android中的所有Java程序都是运行在Dalvik VM上的。Android上的每个程序都有自己的线程，DVM只执行.dex的Dalvik executable 文件。每个Android应用在底层都对应有一个独立的DVM实例并在其解释下执行。

**170. 图片三级缓存的机制**

实现图片缓存也不难，需要有相应的cache策略。这里我采用 内存-文件-网络 三层cache机制，其中内存缓存包括强引用缓存和软引用缓存（SoftReference），其实网络不算cache，这里姑且也把它划到缓存的层次结构中。当根据url向网络拉取图片的时候，先从内存中找，如果内存中没有，再从缓存文件中查找，如果缓存文件中也没有，再从网络上通过http请求拉取图片。在键值对（key-value）中，这个图片缓存的key是图片url的hash值，value就是bitmap。所以，按照这个逻辑，只要一个url被下载过，其图片就被缓存起来了。

**171. 你看好的行业领域**

穿戴

**172. 如何控制耗电量**

在以下情况下耗电比较多：

1、 大数据量的传输。

2、 不停的在网络间切换。

3、 解析大量的文本数据。

**1、 在需要网络连接的程序中，首先检查网络连接是否正常，如果没有网络连接，那么就不需要执行相应的程序。**

**2、 使用效率高的数据格式和解析方法。(fastJson,LoganSquare-master)**

3、使用GZIP压缩方式下载数据，能减少网络流量

4、 其它一些优化方法：  
回收java对象，特别是较大的java对像

对定位要求不是太高的话尽量不要使用GPS定位，可能使用wifi和移动网络cell定位即可。GPS定位消耗的电量远远高于移动网络定位。

尽量不要使用浮点运算。

获取屏幕尺寸等信息可以使用缓存技术，不需要进行多次请求。

**173. Service用在哪个线程**

默认情况下Service是运行在启动该Service的应用主线程的，如果Service中的操作占用大量的CPU资源或有阻断操作（比如播放MP3或者访问网络）会影响应用主线程的响应性能，甚至会造成“应用程序无响应(ANR)”问题。

**174. ContentProvider和SQL的区别**

Sql只能在该工程的内部共享数据，ContentProvider能在工程之间实现数据共享。

**175.Android如何访问自定义ContentProvider**

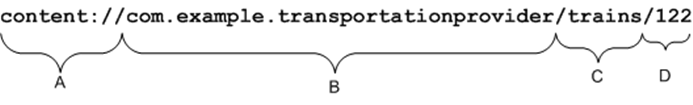
1.第一：得到ContentResolver类对象：ContentResolver cr = getContentResolver（）；

2. 第二：定义要查询的字段String数组。

3. 第三：使用cr.query();返回一个Cursor对象。

4. 第四：使用while循环得到Cursor里面的内容。

**176.Uri有什么组成**

答：URI由是部分组成：

A，标准的前缀: ContentProvider（内容提供者）的scheme已经由Android所规定， scheme为：content://

B，唯一标识整个Content Provider: 主机名（或叫Authority）用于唯一标识这个ContentProvider，外部调用者可以根据这个标识来找到它。

C，描述了数据的路径，确定返回哪类数据: 路径（path）可以用来表示我们要操作的数据，路径的构建应根据业务而定

1. ID唯一标注请求的数据: 我们要请求数据的ID

**177. 对android系统的理解**

5大优势：

1. 开放性
2. 挣脱运营商的束缚
3. 丰富的硬件选择
4. 不受任何限制的开发商
5. 无缝结合的Google应用

Android的5大不足

1. 安全和隐私
2. 首先开卖Android手机的不是最大运营商
3. 运营商仍然能够影响到Android手机
4. 同类机型用户减少
5. 过分依赖开发商缺少标准配置

**178. 简单介绍服务**

服务是没有界面的长生命周期的代码。一个很好的例子是媒体播放器从列表中播放歌曲。在一个媒体播放器程序中，大概要有一个或多个活动（activity）来供用户选择歌曲并播放它。然而，音乐的回放就不能使用活动（activity）了，因为用户希望他导航到其他界面时音乐继续播放。这种情况下，媒体播放器活动（activity）要用Context.startService()启动一个服务来在后台运行保持音乐的播放。系统将保持这个音乐回放服务的运行直到它结束。注意一下，你要用Context.bindService()方法连接服务（如果它没有运行，要先启动它）。当连接到服务后，你可以通过服务暴露的一个接口和它通信。对于音乐服务，它允许你暂停、倒带，等等。

**179. 一条最长短信约占多少byte**

160

**180. 上拉加载下拉刷新如何实现**

答：实现OnScrollListener 接口重写onScrollStateChanged 和onScroll方法，使用onscroll方法实现”滑动“后处理检查是否还有新的记录，如果有，调用 addFooterView，添加记录到adapter, adapter调用 notifyDataSetChanged 更新数据;如果没有记录了，把自定义的mFooterView去掉。使用onScrollStateChanged可以检测是否滚到最后一行且停止滚动然后执行加载

**181.手机适配如何做，自适应**

答:

在布局中设置控件尺寸是尽量使用fill-parent,warp-content,dp等值，文字使用sp

尽量使用Relativelayout 。

对不同的屏幕提供不同大小的图片，

已知应用支持平台设备的分辨率,可以提供多个layout\_320\*480 ...

drawable-hdpi,drawable-mdpi,drawable-ldpi分别代表分辨率为480\*800,360\*480,240\*360, 放置图片大小相差1.5倍

最后还需要在AndroidManifest.xml里添加下面一段，没有这一段自适应就不能实现：

<supports-screens

android:largeScreens="true"

android:normalScreens="true"

android:anyDensity = "true"/>

在</application>标签和</manifest> 标签之间添加上面那段代码。即可。

**182. Manifest中放什么东西**

AndroidManifest.xml是每个android程序中必须的文件。它位于整个项目的根目录，描述了package中暴露的组件（activities,services,等等），他们各自的实现类，各种能被处理的数据和启动位置。

除了能声明程序中的Activities,ContentProviders,Services,和IntentReceivers,还能指定

permissions和instrumentation（安全控制和测试）

**183. 自定义组件的实现思路**

答: Android自定义组件有三种实现思路:

1. 继承某个现有组件，在其基础上添加额外功能,如继承Gallery实现CoverFlow效果
2. 继承某个Layout，实现复合组件自定义，如TextView和EditText组合实现登录注册组件
3. 继承View，实现onDraw()方法，实现自己绘制组件，如翻页效果组件

**184. 项目中的混淆打包**

a在项目的project.properties文件中加入如下代码：

proguard.config=proguard.cfg即可

好处：1.隐藏代码，避免被反编译2.减少代码提交，提高效率4.android中四大组件不能被混淆，所用在清单文件中声明的都是被混淆5.native不能被混淆

**185. 什么是020，b2b，b2c，**

http://blog.sina.com.cn/s/blog\_64e090b001016843.html

**186.如何进行手势判断**

用onTouchEvent(ev)判断手势

**187. Xutil的用法**

http://www.csdn123.com/html/topnews201408/82/6182.htm

**188. ContentProvider Uri 的组成**



**189. 数据库如何写sql语句**

公司框架中中有对数据库的增删改查，有时候自己也写一些复杂的sql语句】

**190. 电商领域的核心**

订单，支付，购物，和信息安全

**191. Asynctask的原理**

AsyncTask的本质是一个线程池，所有提交的异步任务都会在这个线程池中的工作线程内执行，当工作线程需要跟UI线程交互时，工作线程会通过向在UI线程创建的Handler传递消息的方式，调用相关的回调函数，从而实现UI界面的更新。

**192.asset和res的区别**

答：res/raw和assets的相同点：

两者目录下的文件在打包后会原封不动的保存在apk包中，不会被编译成二进制。

res/raw和assets的不同点：

1）res/raw中的文件会被映射到R.java文件中，访问的时候直接使用资源ID即 R.raw.filename；assets文件夹下的文件不会被映射到R.java中，访问的时候需要AssetManager类。

2）res/raw不可以有目录结构，而assets则可以有目录结构，也就是assets目录下可以再建立文件夹

3）读取文件资源举例：

读取res/raw下的文件资源，通过以下方式获取输入流来进行写操作

InputStream is = getResources().openRawResource(R.raw.filename);

读取assets下的文件资源，通过以下方式获取输入流来进行写操作

AssetManager am = null;

am = getAssets();

InputStream is = am.open("filename");

**193. 瀑布流如何实现**

就是异步下载图片的时候把图片添加到scrollview中，

你需要几列就在scrollview中放置几个linearlayout，在scrollview滑动的的时候动态的测量linearlayout随机生成linearlayout的高度

**194. 图片的内存不足**

可能是你加载的当前的图片占用内存大于当前剩余内存，解决方法就是及时释放内存，不显示的图片要释放掉。也可能是你加载的图片是空的，导致内存不足。android系统给图片分配的内存只有8M,当加载大量图片时往往会出现OOM。

Android加载大量图片内存溢出解决方案：

1）尽量不要使用setImageBitmap或setImageResource或BitmapFactory.decodeResource来设置一张大图，因为这些函数在完成decode后，最终都是通过java层的createBitmap来完成的，需要消耗更多内存，可以通过BitmapFactory.decodeStream方法，创建出一个bitmap，再将其设为ImageView的 source

2）使用BitmapFactory.Options对图片进行压缩

InputStream is = this.getResources().openRawResource(R.drawable.pic1);  
 BitmapFactory.Options options=new BitmapFactory.Options();  
 options.inJustDecodeBounds = false;  
 options.inSampleSize = 10; //width，hight设为原来的十分一  
 Bitmap btp =BitmapFactory.decodeStream(is,null,options);   
3）运用Java软引用，进行图片缓存，将需要经常加载的图片放进缓存里，避免反复加载

及时销毁不再使用的Bitmap对象

if(!bmp.isRecycle() ){  
 bmp.recycle() //回收图片所占的内存  
 system.gc() //提醒系统及时回收  
}

**195. Http和Https有什么区别**

http和https使用的是完全不同的连接方式,用的端口也不一样,前者是80,后者是443。http的连接很简单,是无状态的,... HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议 要比http协议安全

**196. 相册缩略图如何实现**

public class Thumbnail extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

try {

saveMyBitmap("ooo");

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void saveMyBitmap(String bitName) throws IOException {

File originalFile = new File("sdcard/pic/ll.jpg");

Bitmap bmp = decodeFile(originalFile);

File f = new File("/sdcard/" + bitName + ".jpg");

f.createNewFile();

FileOutputStream fOut = null;

try {

fOut = new FileOutputStream(f);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

//BitmapFactory.Options options=new BitmapFactory.Options();

        //options.inSampleSize = 10;

//options.inTempStorage = new byte[16\*1024];

//Bitmap bmp = BitmapFactory.decodeFile("/sdcard/pic/sd.jpg");

//Bitmap bmp = BitmapFactory.decodeFile("/sdcard/pic/ll.jpg", options);

//bmp = Bitmap.createScaledBitmap(bmp, 800, 480, true);

bmp.compress(Bitmap.CompressFormat.JPEG, 30, fOut);

try {

fOut.flush();

} catch (IOException e) {

e.printStackTrace();

}

try {

fOut.close();

} catch (IOException e) {

e.printStackTrace();

}

}

//decodes image and scales it to reduce memory consumption

private Bitmap decodeFile(File f){

   try {

       //Decode image size

       BitmapFactory.Options o = new BitmapFactory.Options();

       o.inJustDecodeBounds = true;

       BitmapFactory.decodeStream(new FileInputStream(f),null,o);

       //The new size we want to scale to

       final int REQUIRED\_HEIGHT=800;

       final int REQUIRED\_WIDTH=480;

       //Find the correct scale value. It should be the power of 2.

       int width\_tmp=o.outWidth, height\_tmp=o.outHeight;

       System.out.println(width\_tmp+"  "+height\_tmp);

       Log.w("===", (width\_tmp+"  "+height\_tmp));

       int scale=1;

       while(true){

           if(width\_tmp/2<REQUIRED\_WIDTH && height\_tmp/2<REQUIRED\_HEIGHT)

               break;

           width\_tmp/=2;

           height\_tmp/=2;

           scale++;

           Log.w("===", scale+"''"+width\_tmp+"  "+height\_tmp);

       }

       //Decode with inSampleSize

       BitmapFactory.Options o2 = new BitmapFactory.Options();

       o2.inSampleSize=scale;

       return BitmapFactory.decodeStream(new FileInputStream(f), null, o2);

   } catch (FileNotFoundException e) {}

   return null;

}

}

**197.View更新页面的五种方式**

Activity.runOnUiThread(Runnable)

View.post(Runnable)

View.postDelayed(Runnable,long);

View.postInvalidate();

View.Invalidate();

**198.Android程序的入口**

action节点中的android.intent.action.MAIN表明它所在的Activity是整个应用程序的入口点

**199. 高德地图了解**

地图渲染技术：性能提升10倍，所占空间降低80%，比传统地图软件节省流量超过90%

专业在线导航功能：覆盖全国364个城市、全国道路里程352万公里

在线[导航](http://baike.baidu.com/view/425408.htm)功能：最新[高德](http://baike.baidu.com/view/876937.htm)在线导航引擎，全程语音指引提示，完善偏航判定和偏航重导功能。

AR虚拟实景：AR功能结合手机摄像头和用户位置、方向等信息，将信息点以更直观的方式展现给用户，为发现和目标地点指引

丰富的出行查询功能：地名信息查询、分类信息查询、公交换乘、驾车路线规划、公交线路查询、位置收藏夹等丰富的基础地理信息查询工具。

锁屏语音提示，即使你的iPhone在锁屏状态也能听到高德导航的语音提示，不用担心一直开着iPhone屏幕费电了。

夜间导航HUD抬头提示，打开高德导航并开启[HUD](http://baike.baidu.com/view/1122312.htm)，把iPhone放到汽车挡风玻璃下，高德导航会把路线提示倒映到汽车挡风玻璃上，看起来特别方便，不用担心低头看手机影响驾驶了。

**动态导航**

交通路况实时播报

智能计算到达目的地所需的时间

避堵路线方案规划

摄像头提醒

**离线下载**

2D、3D离线地图

分地区下载地图包

全国地图包、全国概要图

**地图搜索**

热门地点、线路搜索

公交、自驾出行线路规划

公交、火车、天气查询服务

**全新引擎**

最新3D版本，360度旋转视角

矢量数据传送

观看更流畅、更清晰

**兴趣点**

餐饮、住宿、优惠、

演出、团购全覆盖

海量兴趣点随意搜

**叫车服务**

2013年7月,高德地图更新，新增打车功能，综合快的打车、嘀嘀打车两家资源，覆盖80%市场份额。与阿里合作后，高德地图凭借高市场占有率，逐渐从单一地图向LBS工具箱转变。新版高德地图不仅增加了可预约的出租车数量，提高了打车效率，覆盖城市也扩展到26个

**200. Activity中 this和getActivity区别**

This指的是Activity当前的上下文

getActivity()指的是在fragment当中调用得到他所在的Activity

**201. 单例模式和抽象工厂各自的优点**

单例模式：减少创建对象的次数，对内存的管理更方便，占用内存更少

抽象工厂：把公共的部分抽取出，提高了代码的复用性，也减少了代码量，便于管理。

**202. Android的几种缓存方式**

内存缓存，文件缓存，网络缓存

**203. 简单讲解一下递归**

[递归算法](http://baike.baidu.com/view/1733593.htm)一般用于解决三类问题：

(1)数据的定义是按递归定义的。（Fibonacci[函数](http://baike.baidu.com/view/15061.htm)）

(2)问题解法按[递归算法](http://baike.baidu.com/view/1733593.htm)实现。

这类问题虽则本身没有明显的递归结构，但用递归求解比迭代求解更简单，如Hanoi问题。

(3)数据的结构形式是按递归定义的。

如二叉树、广义表等，由于结构本身固有的递归特性，则它们的操作可递归地描述。

递归的缺点：

[递归算法](http://baike.baidu.com/view/1733593.htm)解题相对常用的算法如普通循环等，运行效率较低。因此，应该尽量避免使用递归，除非没有更好的[算法](http://baike.baidu.com/view/7420.htm)或者某种特定情况，递归更为适合的时候。在[递归调用](http://baike.baidu.com/view/1265506.htm)的过程当中系统为每一层的返回点、局部量等开辟了栈来存储。递归次数过多容易造成[栈溢出](http://baike.baidu.com/view/2061755.htm)等。

递归典型问题： 梵塔问题（汉诺塔问题）

已知有三根针分别用A, B, C表示，在A中从上到下依次放n个从小到大的盘子，现要求把所有的盘子

从A针全部移到B针，移动规则是：可以使用C临时存放盘子，每次只能移动一块盘子，而且每根针上

不能出现大盘压小盘，找出移动次数最小的方案.

**204. 如何根据权重实现布局**

//权重和父容器orientation有关

horizontal 指水平方向权重  android:layout\_width

vertical  指垂直方向权重   android:layout\_height

Layout\_weight是线性布局，也就是LinearLayout里面用到的

**在layout\_width设置为fill\_parent的时候，layout\_weight代表的是你的控件要优先尽可能的大,但尽可能大是有限度的，即fill\_parent**.

**在layout\_width设置为wrap\_content的时候，layout\_weight代表的是你的控件要优先尽可能的小,但这个小是有限度的，即wrap\_content.**

**205.在Activity中This和GetApplictionContext的区别**

getApplicationContext() 返回应用的上下文，生命周期是整个应用，应用摧毁它才摧毁  
Activity.this的context 返回当前activity的上下文，属于activity ，activity 摧毁他就摧毁  
getBaseContext() 返回由构造函数指定或setBaseContext()设置的上下文  
this.getApplicationContext（）取的是这个应 用程序的Context，Activity.this取的是这个Activity的Context，这两者的生命周期是不同 的，前者的生命周期是整个应用，后者的生命周期只是它所在的Activity。

1. **Android中项目升级注意的问题**

在清单文件里面的android:versionCode和android:versionName有版本文件, 客户端同时存储该应用当前版本号 (SharedPreferences/Sqlite), 每次打开应用,去检测服务器版本号与本地版本号是否一致,如果不一 致,则自定义对话框提示是否下载更新

1. **Xml解析有几种，分别是什么**

三种分别为：sax,pull,dom

dom解析：解析器读入整个文档，然后构建一个驻留内存的树结构，然后就可以使用 DOM 接口来操作这个树结构。优点是对文档增删改查比较方便，缺点占用内存比较大。

sax解析：基于事件驱动型,优点占用内存少，解析速度快，缺点是只适合做文档的读取，不适合做文档的增删改，不能中途停止。

pull解析：同样基于事件驱动型,android 官方API提供,可随时终止,调用next() 方法提取它们

1. **播放视频的实现方式**

1）使用系统自带的播放器来播放，指定Action为ACTION\_VIEW,Data为Uri，Type为其MIME类型。

//调用系统自带的播放器

Intent intent = new Intent(Intent.ACTION\_VIEW);

intent.setDataAndType(uri, "video/mp4");

startActivity(intent);

2)使用VideoView组件来播放, 可以结合MediaController来实现播控, 只是不能随意更改视频的大小及位置。

3)使用MediaPlayer和SurfaceView来实现，这种方式很灵活,可以自定义视频播放的大小和位置。

1. **IPC是什么**

IPC（Inter-Process Communication，进程间通信）,aidl是 Android Interface definition language的缩写，它是一种android内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口.编译器可以通过扩展名为aidl的文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的.

BroadcastReceiver也可以实现进程间通信

ContentProvider 提供进程间数据共享

1. **Viewpager是如何使用**

我们可以做很多事情，从最简单的导航，到页面菜单等等。那如何使用它呢，与LisstView类似，我们也需要一个适配器，他就是PagerAdapter，也可以与fragment结合使用

1. **如何自定义一个Adapter**  
   继承自BaseAdapter实现里面的方法，listView在开始绘制的时候，系统首先调用getCount（）函数，根据他的返回值得到listView的长度，然后根据这个长度，调用getView（）逐一绘制每一行。如果你的getCount（）返回值是0的话，列表将不显示同样return 1，就只显示一行。系统显示列表时，首先实例化一个适配器（这里将实例化自定义的适配器）。当手动完成适配时，必 须手动映射数据，这需要重写getView（）方法。系统在绘制列表的每一行的时候将调用此方法。getView()有三个参数，position表示将显示的是第几行，covertView是从布局文件中inflate来的 布局。我们用LayoutInflater的方法将定义好的main.xml文件提取成View实例用来显示。然后 将xml文件中的各个组件实例化（简单的findViewById()方法）。这样便可以将数据对应到各个组件上了。但是按钮为了响应点击事件，需要为它添加点击监听器，这样就能捕获点击事件。至此一个自定 义的listView就完成了，现在让我们回过头从新审视这个过程。系统要绘制ListView了，他首先获得 要绘制的这个列表的长度，然后开始绘制第一行，怎么绘制呢？调用getView()函数。在这个函数里面 首先获得一个View（实际上是一个ViewGroup），然后再实例并设置各个组件，显示之。好了，绘制完这一行了。那 再绘制下一行，直到绘完为止。在实际的运行过程中会发现listView的每一行没有焦点了，这是因为Button抢夺了listView的焦点，只要布局文件中将Button设置为没有焦点就OK了。