**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是dalivk虚拟机。每一个android应用程序都在自己的进程中运行，都拥有一个dalivk虚拟机实例。而每一个dvm都是在linux的一个进程。所以说可以认为是同一个概念。

**2.android的动画有哪几种？他们的特点和区别是什么？**

答：两种，一种是tween动画，一种是frame动画。tween动画，这种实现方式可以使视图组件移动，放大或缩小以及产生透明度的变化。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。

**3.handler进制的原理：**

答：android提供了handler和looper来满足线程间的通信。Handler先进先出原则。looper用来管理特定线程内对象之间的消息交换（message Exchange）.

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:你可以构造一个handler对象来与looper沟通，以便push新消息到messagequeue里；或者接收looper（从messagequeue里取出）所送来的消息。

3) messagequeue:用来存放线程放入的消息。

4) 线程：UI thread 通常就是main thread,而android启动程序时会为它建立一个message queue.

**4.android view的刷新：**

答：android中对View的更新有很多种方式，使用时要区分不同的应用场合。我感觉最要紧的是分清：多线程和双缓冲的使用情况。  
 1).不使用多线程和双缓冲  
 这种情况最简单了，一般只是希望在View发生改变时对UI进行重绘。你只需在Activity中显式地调用View对象中的invalidate()方法即可。系统会自动调用 View的onDraw()方法。  
 2).使用多线程和不使用双缓冲  
 这种情况需要开启新的线程，新开的线程就不好访问View对象了。强行访问的话会报：android.view.ViewRoot$CalledFromWrongThreadException：Only the originalthread that created a view hierarchy can touch its views.  
 这时候你需要创建一个继承了android.os.Handler的子类，并重写handleMessage(Messagemsg)方法。android.os.Handler是能发送和处理消息的，你需要在Activity中发出更新UI的消息，然后再你的Handler（可以使用匿名内部类）中处理消息（因为匿名内部类可以访问父类变量，你可以直接调用View对象中的invalidate()方法 ）。也就是说：在新线程创建并发送一个Message，然后再主线程中捕获、处理该消息  
 3).使用多线程和双缓冲  
 Android中SurfaceView是View的子类，她同时也实现了双缓冲。你可以定义一个她的子类并实现SurfaceHolder.Callback接口。由于实现SurfaceHolder.Callback接口，新线程就不需要android.os.Handler帮忙了。SurfaceHolder中lockCanvas()方法可以锁定画布，绘制玩新的图像后调用unlockCanvasAndPost(canvas)解锁（显示），还是比较方便得。

**5.说说mvc模式的原理，它在android中的运用:**

答：android的官方建议应用程序的开发采用mvc模式。何谓mvc？

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。  
2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。

3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。 **6.Activity的生命周期:**

答：onCreate: 在这里创建界面，做一些数据 的初始化工作

　　onStart: 到这一步变成用户可见不可交互的

onResume:变成和用户可交互 的，(在activity 栈系统通过栈的方式管理这些个Activity的最上面，运行完弹出栈，则回到上一个Activity)

　　onPause: 到这一步是可见但不可交互的，系统会停止动画 等消耗CPU 的事情从上文的描述已经知道，应该在这里保存你的一些数据,因为这个时候你的程序的优先级降低，有可能被系统收回。在这里保存的数据，应该在

　　onstop: 变得不可见，被下一个activity覆盖了

onDestroy: 这是activity被干掉前最后一个被调用方法了，可能是外面类调用finish方法或者是系统为了节省空间将它暂时性的干掉 **7.让Activity变成一个窗口：**

答：Activity属性设定：有时候会做个应用程序是漂浮在手机主界面的。这个只需要在设置下Activity的主题theme,即在Manifest.xml定义Activity的地方加一句：

android :theme="@android:style/Theme.Dialog"  
如果是作半透明的效果：

android:theme="@android:style/Theme.Translucent" **8.Android中常用的五种布局:**

答：LinearLayout线性布局；AbsoluteLayout绝对布局；TableLayout表格布局；RelativeLayout相对布局；FrameLayout帧布局； **9.Android的五种数据存储方式：**

答：sharedPreferences；文件；SQLite；contentProvider；网络 **10.请解释下在单线程模型中Message、Handler、Message Queue、Looper之间的关系：**

答：Handler获取当前线程中的looper对象，looper用来从存有Message的Message Queue里取出message，再由Handler进行message的分发和处理。

**11.AIDL的全称是什么?如何工作?能处理哪些类型的数据?**

答：AIDL(AndroidInterface Definition Language)android接口描述语言 **12.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：**

答：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。代码如下：  
 Intent intent = new Intent();

Intent.setAction(“android.intent.action.View”);

Uri uriBrowsers = Uri.parse(“[http://www.sina.com.cn](http://www.sina.com.cn/)”);

Intent.setData(uriBrowsers);

//包名、要打开的activity  
intent.setClassName(“com.android.browser”,”com.android.browser.BrowserActivity”);

startActivity(intent); **13.什么是ANR,如何避免？**

答：ANR的定义：

在android上，如果你的应用程序有一段时间响应不移灵敏，系统会向用户提示“应用程序无响应”（ANR：application Not Responding）对话框。因此，在程序里对响应性能的设计很重要，这样，系统不会显示ANR给用户。

如何避免：

首先来研究下为什么它会在android的应用程序里发生和如何最佳构建应用程序来避免ANR.  
 android应用程序通常是运行在一个单独的线程（例如：main）里，这就意味你的应用程序所做的事情如果在主线程里占用了大长时间的话，就会引发ANR对话框，因为你的应用程序并没有给自己机会来处理输入事件或者Intent广播。

因此，运行在主线程里的任何访求都尽可能少做事情。特别是，activity应该在它的关键生命周期方法（onCreate()和onResume()）里尽可能少的去作创建操作。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。然而，不是说你的主线程阻塞在那里等待子线程的完成---也不是调用Thread.wait()或者Thread.sleep()。替代的方法是：主线程应该为子线程提供一个Handler,以便完成时能够提交给主线程。以这种方式设计你的应用程序，将能保证你的主线程保持对输入的响应性并能避免由5秒输入事件的超时引发的ANR对话框。这种做法应该在其它显示UI的线程里效仿，因为它们都受相同的超时影响。

IntentReceiver执行时间的特殊限制意味着它应该做：在后台里做小的、琐碎的工作，如保存设定或注册一个Notification。和在主线程里调用的其它方法一样，应用程序应该避免在BroadcastReceiver里做耗时的操作或计算，但也不是在子线程里做这些任务（因为BroadcastReceiver的生命周期短），替代的是，如果响应Intent广播需要执行一个耗时的动作的话，应用程序应该启动一个Service。顺便提及一句，你也应该避免在Intent Receiver里启动一个Activity，因为它会创建一个新的画面，并从当前用户正在运行的程序上抢夺焦点。如果你的应用程序在响应Intent广播时需要向用户展示什么，你应该使用Notification Manager来实现。

一般来说，在应用程序里，100到200ms是用户能感知阻滞的时间阈值，下面总结了一些技巧来避免ANR,并有助于让你的应用程序看起来有响应性。

如果你的应用程序为响应用户输入正在后台工作的话，可以显示工作的进度（ProgressBar和ProgressDialog对这种情况来说很有用）。特别是游戏，在子线程里做移动的计算。如果你的程序有一个耗时的初始化过程的话，考虑可以显示一个Splash Screen或者快速显示主画面并异步来填充这些信息。在这两种情况下，你都应该显示正在进行的进度，以免用户认为程序被冻结了。 **14.什么情况会导致Force Close？如何避免？能否捕获导致其的异常？**

答：如空指针等可以导致ForceClose;可以看Logcat，然后找到对应的程序代码来解决错误。 **15.横竖屏切换时候的activity的生命周期：**

答：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ctrl+F12切换成横屏时

onSaveInstanceState()、onPause()、onStop()、onDestroy()、onCreate()、

onStart()、onRestoreInstanceState()、onResume()

4) 再按ctrl+f12切换成竖屏时，发现打印了两次相同的Log

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) 修改AndroidManifest.xml，把该Activity添加android:configChanges=“orientation”,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|keyboradHidden”,执行步骤3，就只打印onConfigChanged

onConfigurationChanged()

8) 把步骤5的android:configChanges=“orientation”改成

android:configChanges=“orientation|keyboradHidden”,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, 切屏会重新调用各个生命周期，切横屏、竖屏时都只会执行一次，但是竖屏最后多打印一条onConfigurationChanged()

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(),横屏一次，竖屏两次

再总结下整个activity的生命周期：

1） 当前activity产生事件弹出Toast和AlertDialog的时候Activity的生命周期不会有改变

2） Activity运行时按下HOME键（跟被完全覆盖一样的）

onSavaInstanceState === onPause === onStop === onRestart === onStart === onResume

3) 未被完全覆盖，只是失去焦点：

onPause ===onResume **16.如何将SQLite数据库(.db文件)与apk文件一起发布?**

答：可以将.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.db文件复制到res aw目录中 **17.如何将打开res aw目录中的数据库文件?**

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。 **18.android 中有哪几种解析xml的类？官方推荐哪种？以及它们的原理和区别：**

答：XML解析主要有三种方式，SAX、DOM、PULL。常规在PC上开发我们使用Dom相对轻松些，但一些性能敏感的数据库或手机上还是主要采用SAX方 式，SAX读取是单向的，优点:不占内存空间、解析属性方便，但缺点就是对于套嵌多个分支来说处理不是很方便。而DOM方式会把整个XML文件加载到内存 中去，这里[Android开发](http://www.apkbus.com/)网提醒大家该方法在查找方面可以和XPath很好的结合如果数据量不是很大推荐使用，而PULL常常用在J2ME对于节点处 理比较好，类似SAX方式，同样很节省内存，在J2ME中我们经常使用的KXML库来解析。 **19.DDMS和TraceView的区别?**

答：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 **20.谈谈Android的IPC机制：**

答：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。 **21.NDK是什么：**

答：NDK是一系列工具的集合

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 **22.描述一下android的系统架构：**

答：android系统架构分从下往上为Linux内核层、运行库、应用程序框架层和应用程序层。

Linux内核层：负责硬件的驱动程序、网络、电源、系统安全以及内存管理等功能。

运行库和androidruntion：运行库：即c/c++函数库部分，大多数都是开放源代码的函数库，例如webkit，该函数库负责android网页浏览器的运行；例如标准的c函数库libc、openssl、sqlite等，当然也包括支持游戏开发的2dsgl和3dopengles，在多媒体方面有mediaframework框架来支持各种影音和图形文件的播放与显示，如mpeg4、h.264、mp3、aac、amr、jpg和png等众多的多媒体文件格式。Androidruntion负责解释和执行生成的dalvik格式的字节码

应用软件架构：java应用程序开发人员主要是使用该层封装好的api进行快速开发的。

应用程序层：该层是java的应用程序层，android内置的googlemaps、email、IM、浏览器等，都处于该层，java开发人员工发的程序也处于该层，而且和内置的应用程序具有平等的地位，可以调用内置的应用程序，也可以替换内置的应用程序 **23.Activity 与 Task的启动模式有哪些，它们含义具体是什么?**

答：在一个activity中，有多次调用startActivity来启动另一个activity，要想只生成一个activity实例，可以设置启动模式。

一个activity有四种启动模式：standed,signleTop,singleTask,singleInstance

Standed:标准模式，一调用startActivity()方法就会产生一个新的实例。

SingleTop:如果已经有一个实例位于activity栈顶，就不产生新的实例，而只是调用activity中的newInstance()方法。如果不位于栈顶，会产生一个新的实例。

singleTask:会在一个新的task中产生这个实例，以后每次调用都会使用这个，不会去产生新的实例了。

SingleInstance:这个和singleTask基本一样，只有一个区别：在这个模式下的activity实例所处的task中，只能有这个activity实例，不能有其他实例

**24.Application类的作用：**

答：API里的第一句是：

Base class for those who need to maintain global application state

如果想在整个应用中使用全局变量，在java中一般是使用静态变量，public类型；而在android中如果使用这样的全局变量就不符合Android的框架架构，但是可以使用一种更优雅的方式就是使用Application context。   
 首先需要重写Application，主要重写里面的onCreate方法，就是创建的时候，初始化变量的值。然后在整个应用中的各个文件中就可以对该变量进行操作了。   
 启动Application时，系统会创建一个PID，即进程ID，所有的Activity就会在此进程上运行。那么我们在Application创建的时候初始化全局变量，同一个应用的所有Activity都可以取到这些全局变量的值，换句话说，我们在某一个Activity中改变了这些全局变量的值，那么在同一个应用的其他Activity中值就会改变

**25.说明onSaveInstanceState() 和 onRestoreInstanceState()在什么时候被调用：**

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()才会被调用。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。

**26.android的service的生命周期？哪个方法可以多次被调用：**

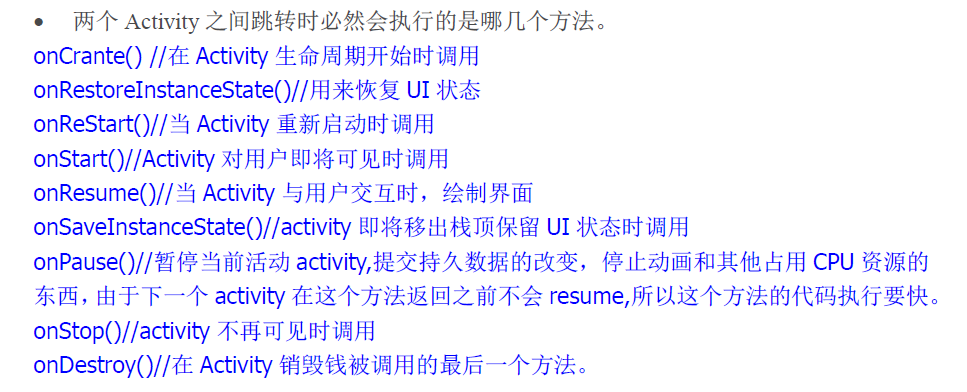
答：1) 与采用Context.startService()方法启动服务有关的生命周期方法  
onCreate() -> onStart() -> onDestroy()  
onCreate()该方法在服务被创建时调用，该方法只会被调用一次，无论调用多少次startService()或bindService()方法，服务也只被创建一次。  
onStart() 只有采用Context.startService()方法启动服务时才会回调该方法。该方法在服务开始运行时被调用。多次调用startService()方法尽管不会多次创建服务，但onStart() 方法会被多次调用。  
onDestroy()该方法在服务被终止时调用。  
2) 与采用Context.bindService()方法启动服务有关的生命周期方法  
onCreate() -> onBind() -> onUnbind() -> onDestroy()  
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。  
如果先采用startService()方法启动服务,然后调用bindService()方法绑定到服务，再调用unbindService()方法解除绑定，最后调用bindService()方法再次绑定到服务，触发的生命周期方法如下：  
onCreate() ->onStart() ->onBind() ->onUnbind()[重载后的方法需返回true] ->onRebind()

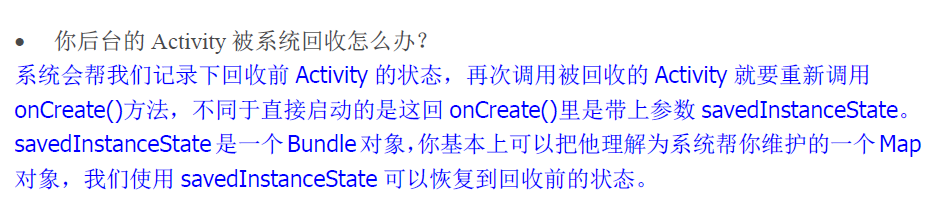
**27.android的broadcast的生命周期：**

答：1) Broadcast receiver生命周期中仅有一个回调方法：   
void onReceive(Context curContext, Intent broadcastMsg)   
当接收器接收到一条broadcast消息，Android就会调用onReceiver(),并传递给它一个Intent对象，这个对象携带着那条broadcast消息。我们认为仅当执行这个方式时，Broadcast receiver是活动的；这个方法返回时，它就终止了。这就是Broadcast receiver的生命周期。   
2) 由于Broadcast receiver的生命周期很短，一个带有活动的Broadcast receiver的进程是受保护的，以避免被干掉；但是别忘了有一点，Android会在任意时刻干掉那些携带不再活动的组件的进程，所以很可能会造成这个问题。   
3) 解决上述问题的方案采用一个Service来完成这项工作，Android会认为那个进程中（Service所在的进程）仍然有在活动的组件。  **28.android view，surfaceview，glsurfaceview的区别：**

答：SurfaceView是从View基类中派生出来的显示类，直接子类有GLSurfaceView和VideoView，可以看出GL和视频播放以及Camera摄像头一般均使用SurfaceView  
SurfaceView和View最本质的区别在于，surfaceView是在一个新起的单独线程中可以重新绘制画面而View必须在UI的主线程中更新画面。   
那么在UI的主线程中更新画面 可能会引发问题，比如你更新画面的时间过长，那么你的主UI线程会被你正在画的函数阻塞。那么将无法响应按键，触屏等消息。   
当使用surfaceView 由于是在新的线程中更新画面所以不会阻塞你的UI主线程。但这也带来了另外一个问题，就是事件同步。比如你触屏了一下，你需要surfaceView中thread处理，一般就需要有一个event queue的设计来保存touch event，这会稍稍复杂一点，因为涉及到线程同步。   
所以基于以上，根据游戏特点，一般分成两类。   
1)被动更新画面的。比如棋类，这种用view就好了。因为画面的更新是依赖于 onTouch 来更新，可以直接使用 invalidate。 因为这种情况下，这一次Touch和下一次的Touch需要的时间比较长些，不会产生影响。   
2)主动更新。比如一个人在一直跑动。这就需要一个单独的thread不停的重绘人的状态，避免阻塞main UI thread。所以显然view不合适，需要surfaceView来控制。











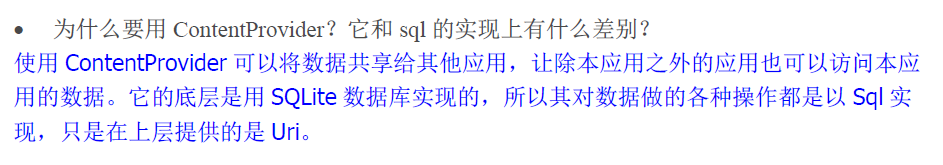
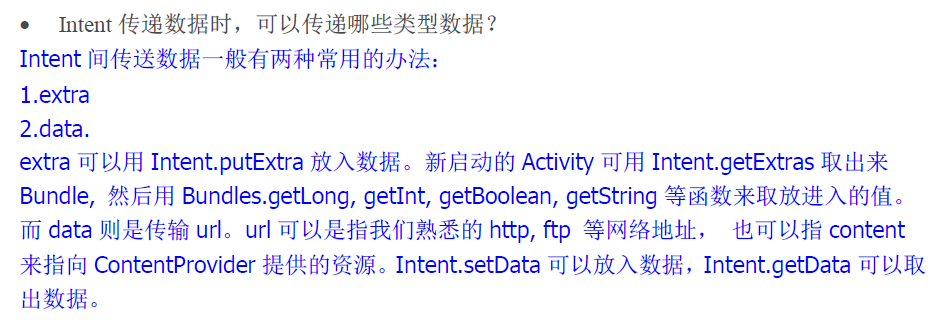
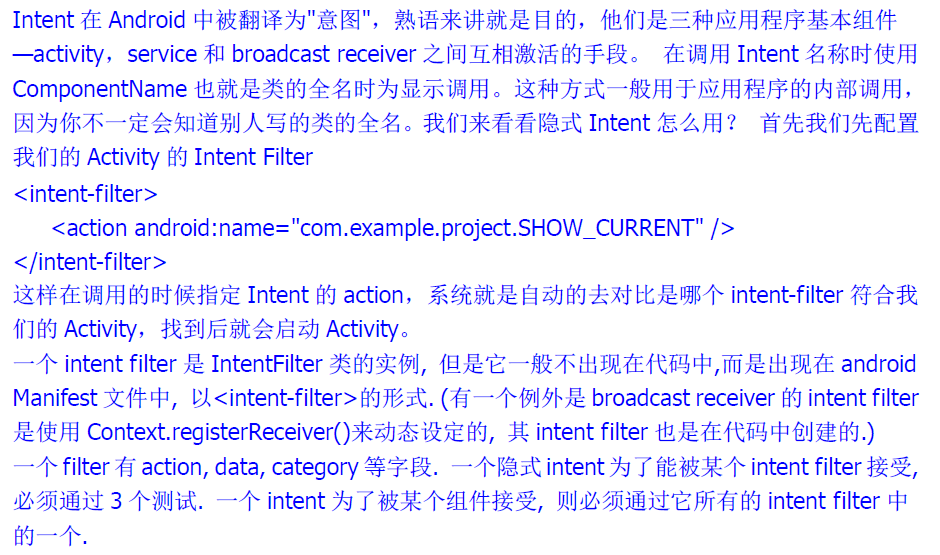












38. ListView如何提高其效率？

复用convertview , 历史的view对象

异步加载数据, 分页加载数据,

使用静态的view对象 避免创建过多的view.

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。