Android 面试题大全

一． Android 入门

1、描述一下 android 的系统架构

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 kernel:

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。

Libraries 和 android Runtime:

Libraries：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（引擎），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 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，例如 mpeg4、h.264、mp3、 aac、amr、jpg 和 png 等众多的多媒体文件格式。

Runtime：负责解释和执行生成的 dalvik 格式的字节码。

Application framework（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。

applications:该层是 java 的应用程序层，android 内置 googlemaps、e-mail、即时通信工具、浏览器、mp3 播放器等处于该层，java 开发人员开发的程序也处于该层，而且和内置的应用程序具有平等的位置，可以调用内置的应用程序，也可以替换内置的应用程序。

应用程序层：

android 应用程序使用框架的 api 并在框架下运行，这就带来了程序开发的高度一致性，另一方面也告诉我们，要想写出优质高效的程序就必须对整个 applicationframework 进行非常深入的理解。精通 applicationframework，你就可以真正的理解 android 的设计和运行机制，也就更能够驾驭整个应用层的开发。

总结：

下层为上层服务，上层需要下层的支持，调用下层的服务，这种严格分层的方式带来的极大的稳定性、灵活性和可扩展性，使得不同层的开发人员可以按照规范专心特定层的开发。

2、Dalvik 和标准 Java 虚拟机之间的主要差别？

Dalvik 和标准 Java 虚拟机（JVM）之间的首要差别之一，就是 Dalvik 基于寄存器，而 JVM 基于栈。

Dalvik 和 Java 之间的另外一大区别就是运行环境—Dalvik 经过优化，允许在有限的内存中同时运行多个虚拟机的实例，并且每一个 Dalvik 应用作为一个独立的 Linux 进程执行。

（1）虚拟机很小，使用的空间也小；

（2）Dalvik 没有 JIT 编译器；

（3）常量池已被修改为只使用 32 位的索引，以简化解释器；

（4）它使用自己的字节码，而非 Java 字节码。

3、Manifest.xml 文件中主要包括哪些信息？

答：manifest：根节点，描述了 package 中所有的内容。
User-sdk：指定支持的手机系统的最小版本
application：包含 package 中 application 级别组件声明的根节点。
activity：Activity 是用来与用户交互的主要工具。
receiver：IntentReceiver 能使的 application 获得数据的改变或者发生的操作，即使它当前不在运行。
service：Service 是能在后台运行任意时间的组件。

provider：ContentProvider 是用来管理持久化数据并发布给其他应用程序使用的组件。
uses-permission：请求你的 package 正常运作所需赋予的安全许可。
permission： 声明了安全许可来限制哪些程序能你 package 中的组件和功能。
instrumentation：声明了用来测试此 package 或其他 package 指令组件的代码。

二． Android 的四大组件

Activity 相关

1、什么是 Activity?

Activity 是一个负责与**用户交互**的组件，Activity 中所有操作都与用户密切相关，可以通过 setContentView(View)来**显示指定控件**。

在一个 android 应用中，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。

2、请描述一下 Activity 生命周期。

onCreate(Bundle savedInstanceState)：
创建 activity 时调用。设置在该方法中，还以 Bundle 的形式提供对以前储存的任何状态的访问！
onStart()：
activity 变为在屏幕上对用户可见时调用。
onResume()：
activity 开始与用户交互时调用（无论是启动还是重新启动一个活动，该方法总是被调用的）。
onPause()：
activity 被暂停或收回 cpu 和其他资源时调用，该方法用于保存活动状态的，也是保护现场，压栈吧！
onStop()：
activity 被停止并转为不可见阶段及后续的生命周期事件时调用。
onRestart()：
重新启动 activity 时调用。该活动仍在栈中，而不是启动新的活动。
onDestroy()：
activity 被完全从系统内存中移除时调用，该方法被调用

3、如何退出 Activity？如何安全退出已调用多个 Activity 的 Application？

在 Android 中退出程序比较麻烦，尤其是在多个 Activity 的程序中，在 2.2 之前可以采用如下代码退出程序：

```
1. ActivityManager am = (ActivityManager)getSystemService (Context.ACTIVIT
   Y_SERVICE);
2. am.restartPackage(getPackageName());
```

此种方法是一种最方便和最简单的退出程序的办法，但是在 2.2 和 2.2 之后就不能用了，一种常用的方法是自定义一个 Activity 的栈，在程序退出时将栈中的所有的 Activity 进行 finish。
还有一些其他的方式，在这
http://alex-yang-xiansoftware-com.iteye.com/blog/1099207 可查看。

4、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

5、 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。

6、 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”

当应用运行起来后就会开启一条线程，线程中会运行一个任务栈，当 Activity 实例创建后就会放入任务栈中。Activity 启动模式的设置在 AndroidManifest.xml 文件中，通过配置 Activity 的属性 android:launchMode=""设置。

1. Standared 模式（默认）

我们平时直接创建的 Activity 都是这种模式的 Activity，这种模式的 Activity 的特点是：只要你创建了 Activity 实例，一旦激活该 Activity，则会向任务栈中加入新创建的实例，退出 Activity 则会在任务栈中销毁该实例。

2. SingleTop 模式

这种模式会考虑当前要激活的 Activity 实例在任务栈中是否正处于栈顶，如果处于栈顶则无需重新创建新的实例，会重用已存在的实例，否则会在任务栈中创建新的实例。

3. SingleTask 模式

如果任务栈中存在该模式的 Activity 实例，则把栈中该实例以上的 Activity 实例全部移除，调用该实例的 newInstance()方法重用该 Activity，使该实例处於栈顶位置，否则就重新创建一个新的 Activity 实例。

4. SingleInstance 模式

当该模式 Activity 实例在任务栈中创建后，只要该实例还在任务栈中，即只要激活的是该类型的 Activity，都会通过调用实例的 newInstance()方法重用该 Activity，此时使用的都是同一个 Activity 实例，它都会处于任务栈的栈顶。此模式一般用于加载较慢的，比较耗性能且不需要每次都重新创建的 Activity。

7、跟 activity 和 Task 有关的 Intent 启动方式有哪些？其含义？

核心的 Intent Flag 有：

FLAG_ACTIVITY_NEW_TASK

FLAG_ACTIVITY_CLEAR_TOP

FLAG_ACTIVITY_SINGLE_TOP

FLAG_ACTIVITY_RESET_TASK_IF_NEEDED

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。

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。

FLAG_ACTIVITY_SINGLE_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。

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。

Service 相关

1、 如何开发一个 Service 组件？

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" />
第三步：启动服务
方法一：context.startService()：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
方法二：context.bindService()：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

2、Service 的生命周期？

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。

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

3、 Service 和 Thread 的区别？

答：servie 是系统的组件，它由系统进程托管（service manager）；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。

4、 简单描述 AIDL

答：由于每个应用程序都运行在自己的进程空间，并且可以从应用程序 UI 运行另一个服务进程，而且经常会在不同的进程间传递对象。在 Android 平台，一个进程通常不能访问另一个进程的内存空间，所以要想对话，需要将对象分解成操作系统可以理解的基本单元，并且有序的通过进程边界。

通过代码来实现这个数据传输过程是冗长乏味的，Android 提供了 AIDL 工具来处理这项工作。

AIDL (Android Interface Definition Language)是一种 IDL 语言，用于生成可以在 Android 设备上两个进程之间进行进程间通信(IPC)的代码。如果在一个进程中（例如 Activity）要调用另一个进程中（例如 Service）对象的操作，就可以使用 AIDL 生成可序列化的参数。

详细查看
http://www.cnblogs.com/over140/archive/2011/03/08/1976890.html

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.

BroadCastReceiver 相关

1、请描述一下 Broadcast Receiver。

Broadcast Receiver 用于接收并处理广播通知(broadcast announcements)。多数的广播是系统发起的，如地域变换、电量不足、来电来信等。程序也可以播放一个广播。程序可以有任意数量的 broadcast receivers 来响应它觉得重要的通知。broadcast receiver 可以通过多种方式通知用户：启动 activity、使用 NotificationManager、开启背景灯、振动设备、播放声音等，最典型的是在状态栏显示一个图标，这样用户就可以点它打开看通知内容。通常我们的某个应用或系统本身在某些事件（电池电量不足、来电来短信）来临时会广播一个 Intent 出去，我们可以利用注册一个 Broadcast Receiver 来监听到这些 Intent 并获取 Intent 中的数据。

2、注册广播有几种方式，这些方式有何优缺点？

答：首先写一个类要继承 BroadcastReceiver

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);
```

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。

ContentProvider 相关

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。

Intent 和 Intent Filter。

1、请描述一下 Intent 和 Intent Filter

Intent 在 Android 中被翻译为"意图"，熟语来讲就是目的，他们是三种应用程序基本组件—activity，service 和 broadcast receiver 之间互相激活的手段。 在调用 Intent 名称时使用 ComponentName 也就是类的全名时为显示调用。这种方式一般用于应用程序的内部调用，因为你不一定会知道别人写的类的全名。我们来看看隐式 Intent 怎么用？ 首先我们先配置我们的 Activity 的 Intent Filter

```
<intent-filter>
    <action android:name="com.example.project.SHOW_CURRENT" />
</intent-filter>
```

这样在调用的时候指定 Intent 的 action，系统就是自动的去对比是哪个 intent-filter 符合我们的 Activity，找到后就会启动 Activity。 一个 intent filter 是 IntentFilter 类的实例，但是它一般不出现在代码中,而是出现在 android Manifest 文件中，以<intent-filter>的形式．（有一个例外是 broadcast receiver 的 intent filter 是使用 Context.registerReceiver()来动态设定的，其 intent filter 也是在代码中创建的.） 一个 filter 有 action，data，category 等字段．一个隐式 intent 为了能被某个 intent filter 接受，必须通过 3 个测试．一个 intent 为了被某个组件接受，则必须通过它所有的 intent filter 中的一个．

详解：http://blog.csdn.net/lmhit/article/details/5576250

综合

1、 Android 的四大组件是哪些，它们的作用？

Activity:

Activity 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，完成相关的事物，定义好需要接受的 Intent 提供同步和异步的接口

Content Provider:

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

BroadCast Receiver:

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

三． Android 的 UI

1、简单介绍一下 Android 中的 View 和 ViewGroup

1、View

在 Andorid 应用程序中，UI 元素称为 View，它们都继承了 android.view.View 类。View 有众多的子类，包括 ViewGroup、基础控件、高级控件和布局。

基础控件主要包括：Button、ImageButton、ToggleButton、TextView、RadioButton、CheckBox、ImageView、ProgressBar、SeekBar 等。

2、ViewGroup

android.view.ViewGroup 类是 android.view.View 重要的子类，ViewGroup 类通常叫做“容器”，它就是由个控件组成的复杂控件，因为它也是 View 类的子类，所以本身也是控件。

ViewGroup 是高级控件的和布局的父类，高级控件是和布局与基础控件一样都是不指具体那个类，而是一类容器的总称。

高级控件都直接或者间接的继承了 android.view.ViewGroup 类，常用的高级控件主要包括：AutoCompleteTextView、Spinner、ListView、GridView、Gallery 等。

2、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）。

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。

LinearLayout：

线性布局，每一个 LinearLayout 里面又可分为垂直布局和水平布局。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。

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；

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。

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。

3、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。

4、ListView 的优化方案

1、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、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、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。

5、View 的刷新：

postinvaliate()可以在分线程刷新
invaliate()只能在主线程中执行

6、NotifactionManager 使用原理

1． 通过 getSystemService 方法获得一个 NotificationManager 对象。

2． 创建一个 Notification 对象。每一个 Notification 对应一个 Notification 对象。在这一步需要设置显示在屏幕上方状态栏的通知消息、通知消息前方的图像资源 ID 和发出通知的时间。一般为当前时间。

3． 由于 Notification 可以与应用程序脱离。也就是说，即使应用程序被关闭，Notification 仍然会显示在状态栏 中。当应用程序再次启动后，又可以重新控制这些 Notification。如清除或替换它们。因此，需要创建一个 PendingIntent 对象。该对象由 Android 系统负责维护，因此，在应用程序关闭后，该对象仍然不会被释放。

4． 使用 Notification 类的 setLatestEventInfo 方法设置 Notification 的详细信息。

5． 使用 NotificationManager 类的 notify 方法显示 Notification 消息。在这一步需要指定标识 Notification 的唯一 ID。这个 ID 必须相对于同一个 NotificationManager 对象是唯一的，否则就会覆盖相同 ID 的 Notificaiton。

7、view 如何刷新？简述什么是双缓冲？

android 中实现 view 的更新有两个方法，一个是 invalidate，另一个是 postInvalidate，其中前者是在 UI 线程自身中使用，而后者在非 UI 线程中使用。

双缓冲

闪烁是图形编程的一个常见问题。当进行复杂的绘制操作时会导致呈现的图像闪烁或具有其他不可接受的外观。双缓冲的使用解决这些问题。双缓冲使用内存缓冲区来解决由多重绘制操作造成的闪烁问题。当使用双缓冲时，首先在内存缓冲区里完成所有绘制操作，而不是在屏幕上直接进行绘图。当所有绘制操作完成后，把内存缓冲区完成的图像直接复制到屏幕。因为在屏幕上只执行一个图形操作，所以消除了由复杂绘制操作造成的图像闪烁问题。

在 android 中实现双缓冲,可以使用一个后台画布 backcanvas,先把所有绘制操作都在这上面进行。等图画好了，然后在把 backcanvas 拷贝到

与屏幕关联的 canvas 上去,如下：

```
Bitmap bitmapBase = new Bitmap()

Canvas backcanvas = new Canvas(bitmapBase)

backcanvas.draw()...//画图

Canvas c = lockCanvas(null);

c.drawbitmap(bitmapBase);//把已经画好的图像输出到屏幕上

unlock(c)....
```

四． Android 数据存储与解析

1、请介绍下 Android 的数据存储方式。

Android 提供了 5 种方式存储数据：
（1）使用 SharedPreferences 存储数据；它是 Android 提供的用来存储一些简单配置信息的一种机制，采用了 XML 格式将数据存储到设备中。只能在同一个包内使用，不能在不同的包之间使用。
（2）文件存储数据；文件存储方式是一种较常用的方法，在 Android 中读取/写入文件

的方法，与 Java 中实现 I/O 的程序是完全一样的，提供了 openFileInput()和 openFileOutput()方法来读取设备上的文件。

（3）SQLite 数据库存储数据；SQLite 是 Android 所带的一个标准的数据库，它支持 SQL 语句，它是一个轻量级的嵌入式数据库。

（4）使用 ContentProvider 存储数据；主要用于应用程序之间进行数据交换，从而能够让其他的应用保存或读取此 Content Provider 的各种数据类型。

（5）网络存储数据；通过网络上提供给我们的存储空间来上传(存储)和下载(获取)我们存储在网络空间中的数据信息。

2、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。

方式一：DOM 解析

优点：

XML 树在内存中完整存储，因此可以直接修改其数据和结构。 2.可以通过该解析器随时访问 XML 树中的任何一个节点。 3.DOM 解析器的 API 在使用上也相对比较简单。

缺点：

如果 XML 文档体积比较大时，将文档读入内存是非常消耗系统资源的。

使用场景：

DOM 是用与平台和语言无关的方式表示 XML 文档的官方 W3C 标准。DOM 是以层次结构组织的节点的集合。这个层次结构允许开发人员在树中寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能进行任何工作。DOM 是基于对象层次结构的。

方式二：SAX 解析

优点：

SAX 对内存的要求比较低，因为它让开发人员自己来决定所要处理的标签。特别是当开发人员只需要处理文档中所包含的部分数据时，SAX 这种扩展能力得到了更好的体现。

缺点：

用 SAX 方式进行 XML 解析时，需要顺序执行，所以很难访问到同一文档中的不同数据。此外，在基于该方式的解析编码过程也相对复杂。

使用场景：

对于含有数据量十分巨大，而又不用对文档的所有数据进行遍历或者分析的时候，使用

该方法十分有效。该方法不用将整个文档读入内存，而只需读取到程序所需的文档标签处即可。

方式三：Xmlpull 解析

android SDK 提供了 xmlpull api，xmlpull 和 sax 类似，是基于流（stream）操作文件，然后根据节点事件回调开发者编写的处理程序。因为是基于流的处理，因此 xmlpull 和 sax 都比较节约内存资源，不会象 dom 那样要把所有节点以对橡树的形式展现在内存中。

xmlpull 比 sax 更简明，而且不需要扫描完整个流。

五． Android 核心机制

消息与异步通信机制

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

简单的说，Handler 获取当前线程中的 looper 对象，looper 用来从存放 Message 的 MessageQueue 中取出 Message，再有 Handler 进行 Message 的分发和处理。

Message Queue（消息队列）：用来存放通过 Handler 发布的消息，通常附属于某一个创建它的线程，可以通过 Looper.myQueue() 得到当前线程的消息队列

Handler：可以发布或者处理一个消息或者操作一个 Runnable，通过 Handler 发布消息，消息将只会发送到与它关联的消息队列，然也只能处理该消息队列中的消息。

Looper：是 Handler 和消息队列之间通讯桥梁，程序组件首先通过 Handler 把消息传递给 Looper，Looper 把消息放入队列。Looper 也把消息队列里的消息广播给所有的

Handler：Handler 接受到消息后调用 handleMessage 进行处理

Message：消息的类型，在 Handler 类中的 handleMessage 方法中得到单个的消息进行处理

在单线程模型下，为了线程通信问题，Android 设计了一个 Message Queue（消息队列），线程间可以通过该 Message Queue 并结合 Handler 和 Looper 组件进行信息交换。下面将对它们进行分别介绍：

1. Message

Message 消息，理解为线程间交流的信息，处理数据后台线程需要更新 UI，则发送 Message 内含一些数据给 UI 线程。

2. Handler

Handler 处理者，是 Message 的主要处理者，负责 Message 的发送，Message 内容的执行处理。后台线程就是通过传进来的 Handler 对象引用来 sendMessage(Message)。而使用 Handler，需要 implement 该类的 handleMessage(Message) 方法，它是处理这些 Message 的操作内容，例如 Update UI。通常需要子类化 Handler 来实现 handleMessage 方法。

3. Message Queue

Message Queue 消息队列，用来存放通过 Handler 发布的消息，按照先进先出执行。

每个 message queue 都会有一个对应的 Handler。Handler 会向 message queue 通过两种方法发送消息：sendMessage 或 post。这两种消息都会插在 message queue 队尾并按先进先出执行。但通过这两种方法发送的消息执行的方式略有不同：通过 sendMessage 发送的是一个 message 对象，会被 Handler 的 handleMessage()函数处理；而通过 post 方法发送的是一个 runnable 对象，则会自己执行。

4．Looper

Looper 是每条线程里的 Message Queue 的管家。Android 没有 Global 的 Message Queue，而 Android 会自动替主线程(UI 线程)建立 Message Queue，但在子线程里并没有建立 Message Queue。所以调用 Looper.getMainLooper()得到的主线程的 Looper 不为 NULL，但调用 Looper.myLooper() 得到当前线程的 Looper 就有可能为 NULL。对于子线程使用 Looper，API Doc 提供了正确的使用方法：这个 Message 机制的大概流程：

1．在 Looper.loop()方法运行开始后，循环地按照接收顺序取出 Message Queue 里面的非 NULL 的 Message。

2．一开始 Message Queue 里面的 Message 都是 NULL 的。当 Handler.sendMessage(Message)到 Message Queue，该函数里面设置了那个 Message 对象的 target 属性是当前的 Handler 对象。随后 Looper 取出了那个 Message，则调用 该 Message 的 target 指向的 Hander 的 dispatchMessage 函数对 Message 进行处理。在 dispatchMessage 方法里，如何处理 Message 则由用户指定，三个判断，优先级从高到低：

1） Message 里面的 Callback，一个实现了 Runnable 接口的对象，其中 run 函数做处理工作；

2） Handler 里面的 mCallback 指向的一个实现了 Callback 接口的对象，由其 handleMessage 进行处理；

3） 处理消息 Handler 对象对应的类继承并实现了其中 handleMessage 函数，通过这个实现的 handleMessage 函数处理消息。

3．Handler 处理完该 Message （update UI） 后，Looper 则设置该 Message 为 NULL，以便回收！

在网上有很多文章讲述主线程和其他子线程如何交互，传送信息，最终谁来执行处理信息之类的，个人理解是最简单的方法—判断 Handler 对象里面的 Looper 对象是属于哪条线程的，则由该线程来执行！

1．当 Handler 对象的构造函数的参数为空，则为当前所在线程的 Looper；

2．Looper.getMainLooper()得到的是主线程的 Looper 对象，Looper.myLooper()得到的是当前线程的 Looper 对象。

2、说说你对 AsyncTask 的理解

在开发 Android 移动客户端的时候往往要使用多线程来进行操作，我们通常会将耗时的操作放在单独的线程执行，避免其占用主线程而给用户带来不好的用户体验。但是在子线程中无法去操作主线程（UI 线程），在子线程中操作 UI 线程会出现错误。因此 android 提供了一个类 Handler 来在子线程中来更新 UI 线程，用发消息的机制更新 UI 界面，呈

现给用户。这样就解决了子线程更新 UI 的问题。但是费时的任务操作总会启动一些匿名的子线程，太多的子线程给系统带来巨大的负担，随之带来一些性能问题。因此 android 提供了一个工具类 AsyncTask，顾名思义异步执行任务。这个 AsyncTask 生来就是处理一些后台的比较耗时
的任务，给用户带来良好用户体验的，从编程的语法上显得优雅了许多，不再需要子线程和
Handler 就可以完成异步操作并且刷新用户界面。

触摸机制

其它

1、说说 mvc 模式的原理，它在 android 中的运用。

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。

六． Android 应用开发相关

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

2、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。

3、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 。

4、Android 中内存泄露出现情况有哪些？

1. 数据库的 cursor 没有关闭， 可以使用 startManagerCursor(cursor)
2. 构造 adapter 时,没有使用缓存 contentview,衍生 listview 的优化问题-----减少创建 view 的对象,充分使用 contentview,可以使用一静态类来优化处理 getview 的过程
3. Bitmap 对象不使用时没有释放，可以通过调用 bitmap.recycle()释放内存

5、什么是 ANR，如何避免它？

ANR：Application Not Responding。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，当用户操作的在 5s 内应用程序没能做出反应，

BroadcastReceiver 在 10 秒内没有执行完毕，就会出现应用程序无响应对话框，这既是 ANR。

避免方法：Activity 应该在它的关键生命周期方法（如 onCreate()和 onResume()）里尽可能少的去做创建操作。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（或者异步方式）来完成。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。

6、DDMS 中 traceview 的作用？

Traceview 是 android 平台配备一个很好的性能分析的工具。它可以通过图形化的方式让我们了解我们要跟踪的程序的性能，并且能具体到 method。
http://wbdban.javaeye.com/blog/564309

7、根据自己的理解描述下 Android 数字签名。

(1)所有的应用程序都必须有数字证书，Android 系统不会安装一个没有数字证书的应用程序
(2)Android 程序包使用的数字证书可以是自签名的，不需要一个权威的数字证书机构签名认证
(3)如果要正式发布一个 Android ，必须使用一个合适的私钥生成的数字证书来给程序签名，而不能使用 adt 插件或者 ant 工具生成的调试证书来发布。
(4)数字证书都是有有效期的，Android 只是在应用程序安装的时候才会检查证书的有效期。如果程序已经安装在系统中，即使证书过期也不会影响程序的正常功能。

七．其它

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。

2、sim 卡的 EF 文件是什么？有何作用

答：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

3、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。

4、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。

5、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。

6、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

7、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

开发性对于 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。

8、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。

9、在 android 中，请简述 jni 的调用过程。

1) 安装和下载 Cygwin，下载 AndroidNDK

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