Android 面试帮助篇

1、要做一个尽可能流畅的 ListView，你平时在工作中如何进行优化的？①Item 布局，层级越少越好，使用 hierarchyview 工具查看优化。

②复用 convertView

③使用 ViewHolder

④item 中有图片时，异步加载

⑤快速滑动时，不加载图片

⑥item 中有图片时，应对图片进行适当压缩

⑦实现数据的分页加载

2、对于 Android 的安全问题，你知道多少

①错误导出组件

② 参数校验不严

③WebView 引入各种安全问题,webview 中的 js 注入

④不混淆、不防二次打包

⑤明文存储关键信息

⑦ 错误使用 HTTPS

⑧山寨加密方法

⑨滥用权限、内存泄露、使用 debug 签名

3、如何缩减 APK 包大小？

代码

保持良好的编程习惯，不要重复或者不用的代码，谨慎添加 libs，移除使用不到

的 libs。

使用 proguard 混淆代码，它会对不用的代码做优化，并且混淆后也能够减少安装包的大小。

native code 的部分，大多数情况下只需要支持 armabi 与 x86 的架构即可。如果非必须，可以考虑拿掉 x86 的部分。

资源

使用 Lint 工具查找没有使用到的资源。去除不使用的图片，String，XML 等等。assets 目录下的资源请确保没有用不上的文件。

生成 APK 的时候，aapt 工具本身会对 png 做优化，但是在此之前还可以使用其他工具如 tinypng 对图片进行进一步的压缩预处理。

jpeg 还是 png，根据需要做选择，在某些时候 jpeg 可以减少图片的体积。 对于 9.png 的图片，可拉伸区域尽量切小，另外可以通过使用 9.png 拉伸达到大图效果的时候尽量不要使用整张大图。

策略

有选择性的提供 hdpi，xhdpi，xxhdpi 的图片资源。建议优先提供 xhdpi 的图片，对于 mdpi，ldpi 与 xxxhdpi 根据需要提供有差异的部分即可。

尽可能的重用已有的图片资源。例如对称的图片，只需要提供一张，另外一张图片可以通过代码旋转的方式实现。

能用代码绘制实现的功能，尽量不要使用大量的图片。例如减少使用多张图片组成 animate-list 的 AnimationDrawable，这种方式提供了多张图片很占空间。

4、Android 与服务器交互的方式中的对称加密和非对称加密是什么？

对称加密，就是加密和解密数据都是使用同一个 key，这方面的算法有 DES。

非对称加密，加密和解密是使用不同的 key。发送数据之前要先和服务端约定生成公钥和私钥，使用公钥加密的数据可以用私钥解密，反之。这方面的算法有 RSA。ssh 和 ssl 都是典型的非对称加密。

5、设备横竖屏切换的时候，接下来会发生什么？

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

6、Android 启动 Service 的两种方式是什么？它们的适用情况是什么？

如果后台服务开始后基本可以独立运行的话，可以用 startService。音乐播放器就可以这样用。它们会一直运行直到你调用 stopSelf 或者 stopService。你可以通过发送 Intent 或者接收 Intent 来与正在运行的后台服务通信，但大部分时间，你只是启动服务并让它独立运行。如果你需要与后台服务通过一个持续的连接来比较频繁地通信，建议使用 bind()。比如你需要定位服务不停地把更新后的地理位置传给 UI。Binder 比 Intent 开发起来复杂一些，但如果真的需要，你也只能使用它。

startService：生命周期与调用者不同。启动后若调用者未调用 stopService 而直接退出，Service 仍会运行

bindService：生命周期与调用者绑定，调用者一旦退出，Service 就会调用 unBind->onDestroy

7、谈谈你对 Android 中 Context 的理解？

Context:包含上下文信息(外部值) 的一个参数. Android 中的 Context 分三种,Application Context ,Activity Context ,Service Context.

它描述的是一个应用程序环境的信息，通过它我们可以获取应用程序的资源和类，也包括一些应用级别操作，例如：启动一个 Activity，发送广播，接受 Intent 信息等

8、Service 的 onCreate 回调在 UI 线程中吗？

Service 生命周期的各个回调和其他的应用组件一样，是跑在主线程中，会影响到你的 UI 操作或者阻塞主线程中的其他事情

9、请介绍下 AsyncTask 的内部实现，适用的场景是？

AsyncTask 内部也是 Handler 机制来完成的，只不过 Android 提供了执行框架来提供线程池来执行相应地任务，因为线程池的大小问题，所以 AsyncTask 只应该用

来执行耗时时间较短的任务，比如 HTTP 请求，大规模的下载和数据库的更改不适用于 AsyncTask，因为会导致线程池堵塞，没有线程来执行其他的任务，导致的情形是会发生 AsyncTask 根本执行不了的问题。

10、谈谈你对 binder 机制的理解?

binder 是一种 IPC 机制,进程间通讯的一种工具.

Java 层可以利用 aidl 工具来实现相应的接口.

11、Android 中进程间通信有哪些实现方式？

Intent，Binder（AIDL），Messenger，BroadcastReceiver

12、介绍下实现一个自定义 view 的基本流程

1、自定义 View 的属性 编写 attr.xml 文件

2、在 layout 布局文件中引用，同时引用命名空间

3、在 View 的构造方法中获得我们自定义的属性 ，在自定义控件中进行读取（构造方法拿到 attr.xml 文件值）

4、重写 onMesure

5、重写 onDraw

13、Android 中 touch 事件的传递机制是怎样的?

1、Touch 事件传递的相关 API 有 dispatchTouchEvent、onTouchEvent、onInterceptTouchEvent

2、Touch 事件相关的类有 View、ViewGroup、Activity

3、Touch 事件会被封装成 MotionEvent 对象，该对象封装了手势按下、移动、松开等动作

4、Touch 事件通常从 Activity#dispatchTouchEvent 发出，只要没有被消费，会一直往下传递，到最底层的 View。

5、如果 Touch 事件传递到的每个 View 都不消费事件，那么 Touch 事件会反向向上传递,最终交由 Activity#onTouchEvent 处理.

6、onInterceptTouchEvent 为 ViewGroup 特有，可以拦截事件.

7、Down 事件到来时，如果一个 View 没有消费该事件，那么后续的 MOVE/UP 事件都不会再给它

14、Android 多线程的实现方式有哪些?

Thread & AsyncTask

Thread 可以与 Loop 和 Handler 共用建立消息处理队列

AsyncTask 可以作为线程池并行处理多任务

15、Android 开发中何时使用多进程？使用多进程的好处是什么？

要想知道如何使用多进程，先要知道 Android 里的多进程概念。一般情况下，一个应用程序就是一个进程，这个进程名称就是应用程序包名。我们知道进程是系统分配资源和调度的基本单位，所以每个进程都有自己独立的资源和内存空间，别的进程是不能任意访问其他进程的内存和资源的。

那如何让自己的应用拥有多个进程？

很简单，我们的四大组件在 AndroidManifest 文件中注册的时候，有个属性是 android:process，

1、这里可以指定组件的所处的进程。默认就是应用的主进程。指定为别的进程之后，系统在启动这个组件的时候，就先创建（如果还没创建的话）这个进程，然后再创建该组件。你可以重载 Application 类的 onCreate 方法，打印出它的进程名称，就可以清楚的看见了。再设置 android:process 属性时候，有个地方需要注意：如果是 android:process=”:deamon”，以:开头的名字，则表示这是一个应用程序的私有进程，否则它是一个全局进程。私有进程的进程名称是会在冒号前自动加上包名，而全局进程则不会。一般我们都是有私有进程，很少使用全局进程。他们的具体区别不知道有没有谁能补充一下。

2、使用多进程显而易见的好处就是分担主进程的内存压力。我们的应用越做越大，内存越来越多，将一些独立的组件放到不同的进程，它就不占用主进程的内存空间了。当然还有其他好处，有心人会发现 Android 后台进程里有很多应用是多个进程的，因为它们要常驻后台，特别是即时通讯或者社交应用，不过现在多

进程已经被用烂了。典型用法是在启动一个不可见的轻量级私有进程，在后台收发消息，或者做一些耗时的事情，或者开机启动这个进程，然后做监听等。还有就是防止主进程被杀守护进程，守护进程和主进程之间相互监视，有一方被杀就重新启动它。应该还有还有其他好处，这里就不多说了。

3、坏处的话，多占用了系统的空间，大家都这么用的话系统内存很容易占满而导致卡顿。消耗用户的电量。应用程序架构会变复杂，应为要处理多进程之间的通信。这里又是另外一个问题了。

16、ANR 是什么？怎样避免和解决 ANR?

ANR:Application Not Responding，即应用无响应

ANR 一般有三种类型：

1：KeyDispatchTimeout(5 seconds) －主要类型

按键或触摸事件在特定时间内无响应

2：BroadcastTimeout(10 seconds)

BroadcastReceiver 在特定时间内无法处理完成

3：ServiceTimeout(20 seconds) －小概率类型

Service 在特定的时间内无法处理完成

超时的原因一般有两种：

(1)当前的事件没有机会得到处理（UI 线程正在处理前一个事件没有及时完成或者 looper 被某种原因阻塞住）

(2)当前的事件正在处理，但没有及时完成

UI 线程尽量只做跟 UI 相关的工作，耗时的工作（数据库操作，I/O，连接网络或者其他可能阻碍 UI 线程的操作）放入单独的线程处理，尽量用 Handler 来处理 UI thread 和 thread 之间的交互。

UI 线程主要包括如下：

Activity:onCreate(), onResume(), onDestroy(), onKeyDown(), onClick()

AsyncTask: onPreExecute(), onProgressUpdate(), onPostExecute(), onCancel()

Mainthread handler: handleMessage(), post(runnable r)

other

17、Android 下解决滑动冲突的常见思路是什么？

相关的滑动组件 重写 onInterceptTouchEvent，然后判断根据 xy 值，来决定是否要拦截当前操作

18、如何把一个应用设置为系统应用？

成为系统应用，首先要在 对应设备的 Android 源码 SDK 下编译，编译好之后：

此 Android 设备是 Debug 版本，并且已经 root，直接将此 apk 用 adb 工具 push 到 system/app 或 system/priv-app 下即可。

如果非 root 设备，需要编译后重新烧写设备镜像即可。

有些权限(如 WRITE_SECURE_SETTINGS)，是不开放给第三方应用的，只能在对应设备源码中编译然后作为系统 app 使用。

19、Android 内存泄露研究

Android 内存泄漏指的是进程中某些对象（垃圾对象）已经没有使用价值了，但是它们却可以直接或间接地引用到 gc roots 导致无法被 GC 回收。无用的对象占据着内存空间，使得实际可使用内存变小，形象地说法就是内存泄漏了。

场景

类的静态变量持有大数据对象

静态变量长期维持到大数据对象的引用，阻止垃圾回收。

非静态内部类的静态实例

非静态内部类会维持一个到外部类实例的引用，如果非静态内部类的实例是静态的，就会间接长期维持着外部类的引用，阻止被回收掉。

资源对象未关闭

资源性对象如 Cursor、File、Socket，应该在使用后及时关闭。未在 finally 中关闭，会导致异常情况下资源对象未被释放的隐患。

注册对象未反注册

未反注册会导致观察者列表里维持着对象的引用，阻止垃圾回收。

Handler 临时性内存泄露

Handler 通过发送 Message 与主线程交互，Message 发出之后是存储在 MessageQueue 中的，有些 Message 也不是马上就被处理的。在 Message 中存在一个 target，是 Handler 的一个引用，如果 Message 在 Queue 中存在的时间越长，就会导致 Handler 无法被回收。如果 Handler 是非静态的，则会导致 Activity 或者 Service 不会被回收。

由于 AsyncTask 内部也是 Handler 机制，同样存在内存泄漏的风险。

此种内存泄露，一般是临时性的。

20、内存泄露检测有什么好方法？

检测：

1、DDMS Heap 发现内存泄露

dataObject totalSize 的大小，是否稳定在一个范围内，如果操作程序，不断增加，说明内存泄露

2、使用 Heap Tool 进行内存快照前后对比

BlankActivity 手动触发 GC 进行前后对比，对象是否被及时回收

定位：

1、MAT 插件打开.hprof 具体定位内存泄露：

查看 histogram 项，选中某一个对象，查看它的 GC 引用链，因为存在 GC 引用链的，说明无法回收

2、AndroidStudio 的 Allocation Tracker：

观测到期间的内存分配，哪些对象被创建，什么时候创建，从而准确定位