

Android工程师之Android面试大纲

2017-08-22 Hensen_ 鸿洋

本文作者

作者：Hensen_

链接：

http://blog.csdn.net/qq_30379689/article/details/73698192

本文为作者投稿。

正处于秋招时期，在此，作者要求我帮忙寻求一份互联网公司2018年校招的安卓开发岗位，希望和志同道合的同志一起奋发向上，有朋友有职位推荐可留下联系方式，或者直接联系作者，谢谢。

微信：510402535

邮箱：xyj510402535@qq.com

文章目录如下：

- Activity面试题
- Fragment面试题
- Service面试题
- Broadcast Receiver面试题
- WebView面试题
- Binder面试题
- Handler面试题
- AsyncTask面试题
- HandlerThread面试题
- IntentService面试题
- 视图工作机制面试题
- 事件分发机制面试题
- ListView面试题
- Android项目构建面试题
- ANR面试题
- OOM面试题

- Bitmap面试题
- UI卡顿面试题
- 内存泄漏面试题
- 内存管理面试题
- 冷启动和热启动面试题
- 其他优化面试题
- 架构模式面试题
- 插件化面试题
- 热更新面试题
- 进程保活面试题
- Lint面试题
- Kotlin面试题

1

Activity面试题

1、Activity是什么

Activity是四大组件之一，它提供一个界面让用户点击和各种滑动操作，这就是Activity

2、Activity四种状态

- running
- paused
- stopped
- killed

3、Activity生命周期

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()
- onRestart()

4、进程的优先级

- 空进程
- 后台进程

- 服务进程
- 可见进程
- 前台进程

5、Activity任务栈

- 先进后出

6、Activity启动模式

- standard
- singletop
- singletask
- singleinstance

7、scheme跳转协议

Android中的scheme是一种页面内跳转协议，通过定义自己的scheme协议，可以跳转到app中的各个页面

- 服务器可以定制化告诉app跳转哪个页面
- App可以通过跳转到另一个App页面
- 可以通过H5页面跳转页面

2

Fragment面试题

1、Fragment为什么被称为第五大组件

Fragment比Activity更节省内存，其切换模式也更加舒适，使用频率不低于四大组件，且有自己的生命周期，而且必须依附于Activity

2、Activity创建Fragment的方式

- 静态创建
- 动态创建

3、FragmentManager和FragmentPagerAdapter的区别

- FragmentPagerAdapter在每次切换页面的时候，是将Fragment进行分离，适合页面较少的Fragment使用以保存一些内存，对系统内存不会多大影响
- FragmentStatePagerAdapter在每次切换页面的时候，是将Fragment进行回收，适合页面较多的Fragment使用，这样就不会消耗更多的内存

4、Fragment生命周期

- onAttach()
- onCreate()
- onCreateView()
- onActivityCreated()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroyView()
- onDestroy()
- onDetach()

5、Fragment的通信

- Fragment调用Activity中的方法：getActivity
- Activity调用Fragment中的方法：接口回调
- Fragment调用Fragment中的方法：FragmentManager.findFragmentById

6、Fragment的replace、add、remove方法

- replace：替代Fragment的栈顶页面
- add：添加Fragment到栈顶页面
- remove：移除Fragment栈顶页面

3

Service面试题

1、Service是什么

Service是四大组件之一，它可以在后台执行长时间运行操作而没有用户界面的应用组件

2、Service和Thread的区别

- Service是安卓中系统的组件，它运行在独立进程的主线程中，不可以执行耗时操作。Thread是程序执行的最小单元，分配CPU的基本单位，可以开启子线程执行耗时操作
- Service在不同Activity中可以获取自身实例，可以方便的对Service进行操作。Thread在不同的Activity中难以获取自身实例，如果Activity被销毁，Thread实例就很难再获取得到

3、Service启动方式

- startService
- bindService

4、Service生命周期

1. startService

- onCreate()
- onStartCommand()
- onDestroy()

2. bindService

- onCreate()
- onBind()
- onUnbind()
- onDestroy()

4

Broadcast Receiver面试题

1、Broadcast Receiver是什么

Broadcast是四大组件之一，是一种广泛运用在应用程序之间传输信息的机制，通过发送Intent来传送我们的数据

2、Broadcast Receiver的使用场景

- 同一App具有多个进程的不同组件之间的消息通信
- 不同App之间的组件之间的消息通信

3、Broadcast Receiver的种类

- 普通广播
- 有序广播
- 本地广播
- Sticky广播

4、Broadcast Receiver的实现

- 静态注册：注册后一直运行，尽管Activity、进程、App被杀死还是可以接收到广播
- 动态注册：跟随Activity的生命周期

5、Broadcast Receiver实现机制

- 自定义广播类继承BroadcastReceiver，复写onReceiver()
- 通过Binder机制向AMS进行注册广播
- 广播发送者通过Binder机制向AMS发送广播
- AMS查找符合相应条件的广播发送到BroadcastReceiver相应的循环队列中
- 消息队列执行拿到广播，回调BroadcastReceiver的onReceiver()

6、LocalBroadcastManager特点

- 本地广播只能在自身App内传播，不必担心泄漏隐私数据
- 本地广播不允许其他App对你的App发送该广播，不必担心安全漏洞被利用
- 本地广播比全局广播更高效
- 以上三点都是源于其内部是用Handler实现的

5

WebView面试题

1、WebView安全漏洞

- API16之前存在远程代码执行安全漏洞，该漏洞源于程序没有正确限制使用WebView.addJavascriptInterface方法，远程攻击者可通过使用Java反射机制利用该漏洞执行任意Java对象的方法

2、WebView销毁步骤

- WebView在其他容器上时（如：LinearLayout），当销毁Activity时，需要在onDestroy()中先移除容器上的WebView，然后再将WebView.destroy()，这样就不会导致内存泄漏

3、WebView的jsbridge

- 客户端和服务端之间可以通过Javascript来互相调用各自的方法

4、WebViewClient的onPageFinished

- WebViewClient的onPageFinished在每次完成页面的时候调用，但是遇到未加载完成的页面跳转其他页面时，就会一直调用，使用WebChromeClient.onProgressChanged可以替代

5、WebView后台耗电

- 在WebView加载页面的时候，会自动开启线程去加载，如果不很好的关闭这些

线程，就会导致电量消耗加大，可以采用暴力的方法，直接在onDestroy方法中System.exit(0)结束当前正在运行中的java虚拟机

6、WebView硬件加速

Android3.0引入硬件加速，默认会开启，WebView在硬件加速的情况下滑动更加平滑，性能更加好，但是会出现白块或者页面闪烁的副作用，建议WebView暂时关闭硬件加速

7、WebView内存泄漏

由于WebView是依附于Activity的，Activity的生命周期和WebView启动的线程的生命周期是不一致的，这会导致WebView一直持有对这个Activity的引用而无法释放，解决方案如下

- 独立进程，简单暴力，不过可能涉及到进程间通信（推荐）
- 动态添加WebView，对传入WebView中使用的Context使用弱引用

6

Binder面试题

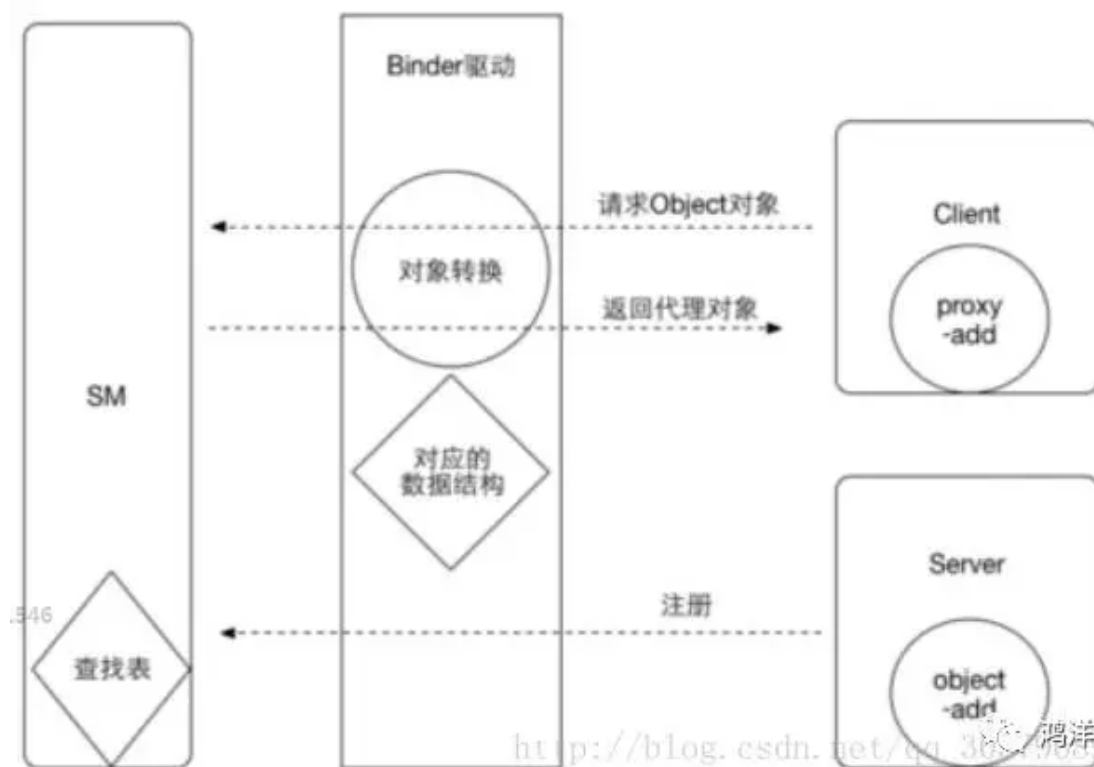
1、Linux内核的基本知识

- 进程隔离/虚拟地址空间：进程间是不可以共享数据的，相当于被隔离，每个进程被分配到不同的虚拟地址中
- 系统调用：Linux内核对应用有访问权限，用户只能在应用层通过系统调用，调用内核的某些程序
- binder驱动：它负责各个用户的进程，通过binder通信内核来进行交互的模块

2、为什么使用Binder

- 性能上，相比传统的Socket更加高效
- 安全性高，支持协议双方互相校验

3、Binder通信模型



- Service服务端通过Binder驱动在ServiceManager的查找表中注册Object对象的add方法
- Client客户端通过Binder驱动在ServiceManager的查找表中找到Object对象的add方法，并返回proxy的add方法，add方法是个空实现，proxy也不是真正的Object对象，是通过Binder驱动封装好的代理类的add方法
- 当Client客户端调用add方法时，Client客户端通过Binder驱动将proxy的add方法，请求ServiceManager来找到Service服务端真正对象的add方法，进行调用

4、AIDL

- 客户端通过aidl文件的Stub.asInterface()方法，拿到Proxy代理类
- 通过调用Proxy代理类的方法，将参数进行封包后，调用底层的transact()方法
- transact()方法会回调onTransact()方法，进行参数的解封
- 在onTransact()方法中调用服务端对应的方法，并将结果返回

7 Handler面试题

1、Handler是什么

Handler通过发送和处理Message和Runnable对象来关联相对应线程的MessageQueue

2、Handler使用方法

- post(runnable)
- sendMessage(message)

3、Handler工作原理

- Android进阶——Android消息机制之Looper、Handler、MessageQueue
- http://blog.csdn.net/qq_30379689/article/details/53394061

4、Handler引起的内存泄漏

- 原因：非静态内部类持有外部类的匿名引用，导致Activity无法释放
- 解决：
 - Handler内部持有外部Activity的弱引用
 - Handler改为静态内部类
 - Handler.removeCallback()

8

AsyncTask面试题

1、AsyncTask是什么

它本质上就是一个封装了线程池和Handler的异步框架

2、AsyncTask使用方法

- 三个参数
 - Params：表示后台任务执行时的参数类型，该参数会传给AsyncTask的doInBackground()方法
 - Progress：表示后台任务的执行进度的参数类型，该参数会作为onProgressUpdate()方法的参数
 - Result：表示后台任务的返回结果的参数类型，该参数会作为onPostExecute()方法的参数
- 五个方法
 - onPreExecute()：异步任务开启之前回调，在主线程中执行
 - doInBackground()：执行异步任务，在线程池中执行
 - onProgressUpdate()：当doInBackground中调用publishProgress时回调，在主线程中执行
 - onPostExecute()：在异步任务执行之后回调，在主线程中执行
 - onCancelled()：在异步任务被取消时回调

3、AsyncTask工作原理

- Android进阶——多线程系列之异步任务AsyncTask的使用与源码分析
- http://blog.csdn.net/qq_30379689/article/details/53203556

4、AsyncTask引起的内存泄漏

- 原因：非静态内部类持有外部类的匿名引用，导致Activity无法释放
- 解决：
 - AsyncTask内部持有外部Activity的弱引用
 - AsyncTask改为静态内部类
 - AsyncTask.cancel()

5、AsyncTask生命周期

在Activity销毁之前，取消AsyncTask的运行，以此来保证程序的稳定

6、AsyncTask结果丢失

由于屏幕旋转、Activity在内存紧张时被回收等情况下，Activity会被重新创建，此时，旧的AsyncTask持有旧的Activity引用，这个时候会导致AsyncTask的onPostExecute()对UI更新无效

7、AsyncTask并行or串行

- AsyncTask在Android 2.3之前默认采用并行执行任务，AsyncTask在Android 2.3之后默认采用串行执行任务
- 如果需要在Android 2.3之后采用并行执行任务，可以调用AsyncTask的executeOnExecutor()

9

HandlerThread面试题

1、HandlerThread产生背景

当系统有多个耗时任务需要执行时，每个任务都会开启一个新线程去执行耗时任务，这样会导致系统多次创建和销毁线程，从而影响性能。为了解决这一问题，Google提供了HandlerThread，HandlerThread是在线程中创建一个Looper循环器，让Looper轮询消息队列，当有耗时任务进入队列时，则不需要开启新线程，在原有的线程中执行耗时任务即可，否则线程阻塞

2、HandlerThread的特点、

- HandlerThread本质上是一个线程，继承自Thread
- HandlerThread有自己的Looper对象，可以进行Looper循环，可以创建

Handler

- HandlerThread可以在Handler的handlerMessage中执行异步方法
- HandlerThread优点是异步不会堵塞，减少对性能的消耗
- HandlerThread缺点是不能同时继续进行多任务处理，需要等待进行处理，处理效率较低
- HandlerThread与线程池不同，HandlerThread是一个串行队列，背后只有一个线程。

10

IntentService面试题

1、IntentService是什么

IntentService是继承自Service并处理异步请求的一个类，其内部采用HandlerThread和Handler实现的，在IntentService内有一个工作线程来处理耗时操作，其优先级比普通Service高。当任务完成后，IntentService会自动停止，而不需要手动调用stopSelf()。另外，可以多次启动IntentService，每个耗时操作都会以工作队列的方式在IntentService中onHandlerIntent()回调方法中执行，并且每次只会执行一个工作线程

2、IntentService使用方法

- 创建Service继承自IntentService
- 覆写构造方法和onHandlerIntent()方法
- 在onHandlerIntent()中执行耗时操作

11

视图工作机制面试题

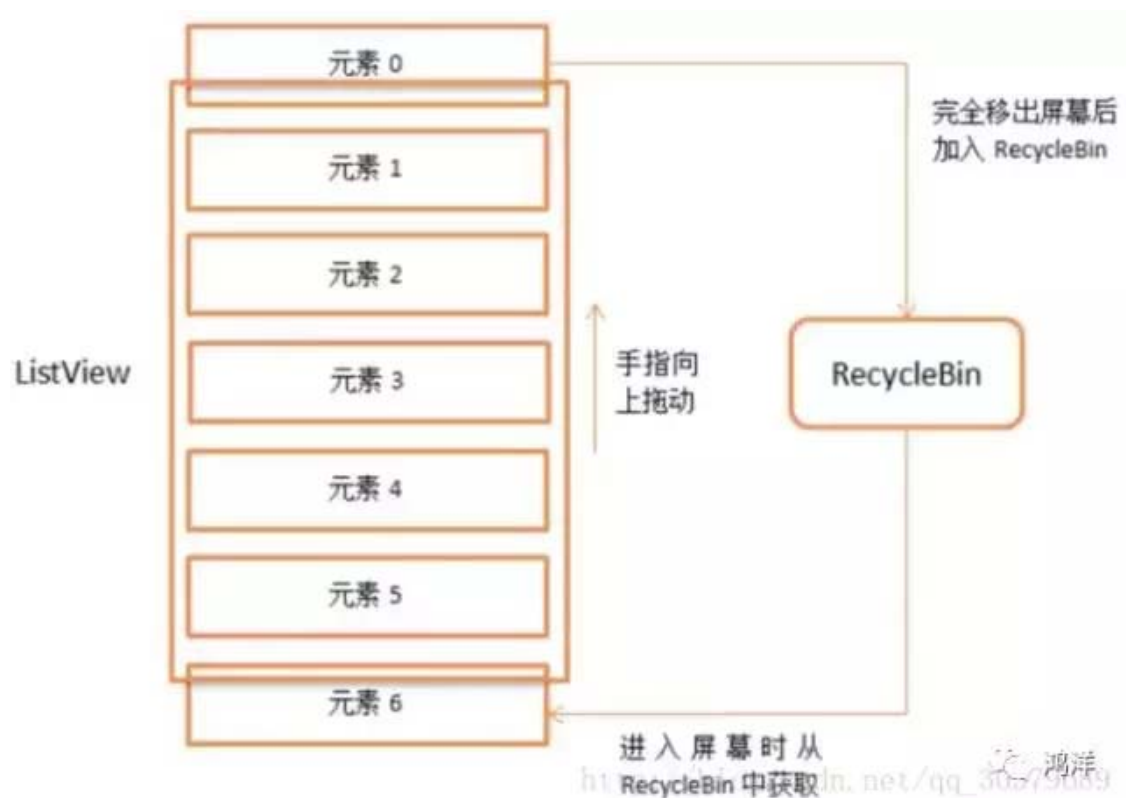
- Android进阶——Android视图工作机制之measure、layout、draw
- http://blog.csdn.net/qq_30379689/article/details/54588736
- Android事件分发机制之dispatchTouchEvent、onInterceptTouchEvent、onTouchEvent
- http://blog.csdn.net/qq_30379689/article/details/53967177

12 ListView面试题

1、ListView是什么

ListView是能将一个数据集合以动态滚动的方式展示到用户界面上的View

2、ListView的RecycleBin机制

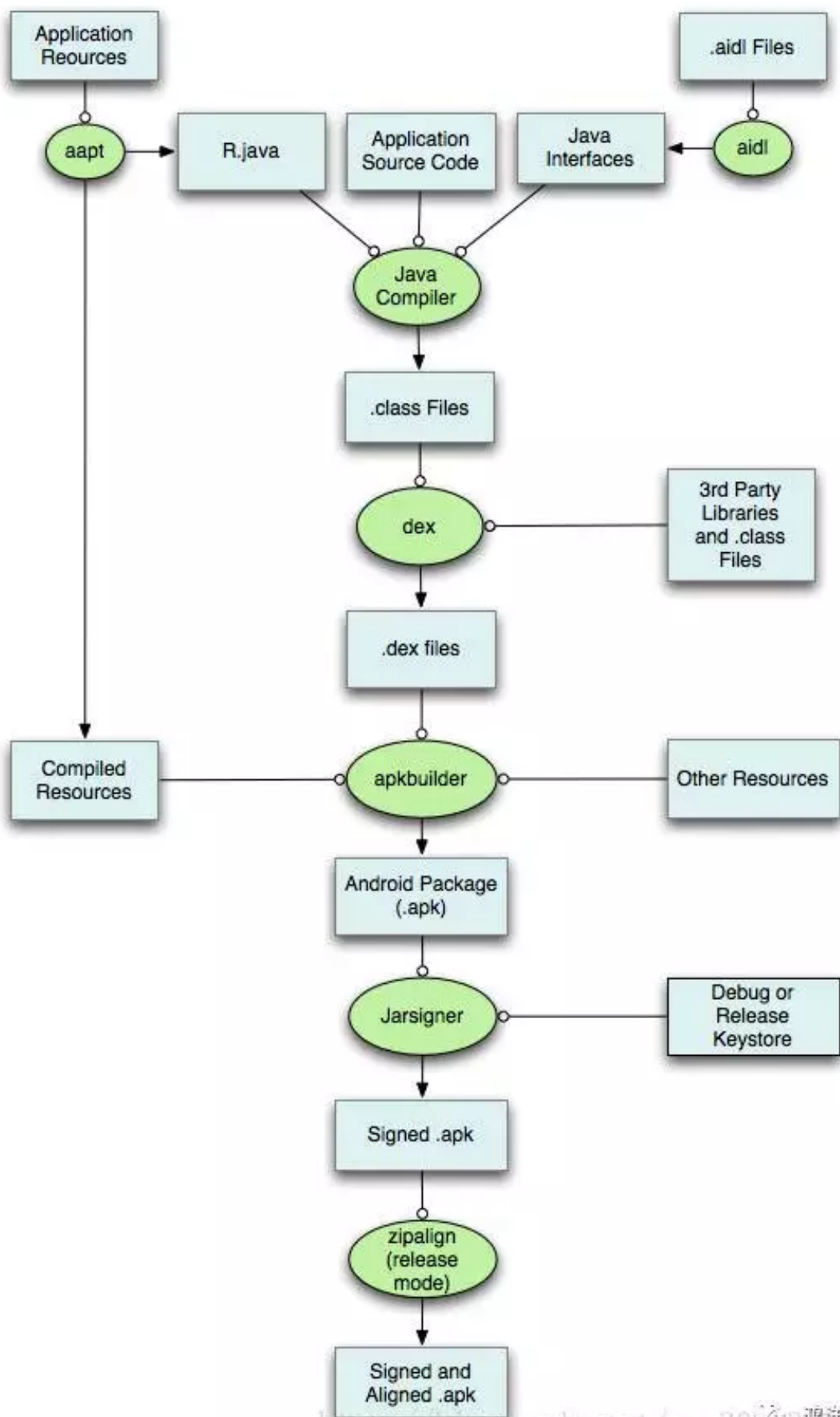


3、ListView的优化

- 重用convertView
- 使用ViewHolder
- 图片三级缓存
- 监听滑动事件
- 少用透明View
- 开启硬件加速

13 Android项目构建面试题

1、android构建流程



2、jenkins持续集成构建

- 这里可参考蒲公英文档
- <http://www.pgyer.com/doc/view/jenkins>

3、Git常用命令

- git init : 仓库的初始化
- git status : 查看当前仓库的状态
- git diff : 查看仓库与上次修改的内容
- git add : 将文件放进暂存区
- git commit : 提交代码
- git clone : 克隆代码
- git bransh : 查看当前分支
- git checkout : 切换当前分支

4、git工作流

- fork/clone (主流)
 - fork : 将别人的仓库代码fork到自己的仓库上
 - clone : 克隆下自己仓库的代码
 - update、commit : 修改代码并提交到自己的仓库
 - push : 提交到自己的仓库
 - pull request : 请求添加到别人的仓库
- clone

5、proguard是什么

ProGuard工具是用于压缩、优化和混淆我们的代码，其主作用是移除或混淆代码中无用类、字段、方法和属性

6、proguard技术功能

- 压缩
- 优化
- 混淆
- 预检测

7、proguard工作原理

将无用的字段或方法存入到EntryPoint中，将非EntryPoint的字段和方法进行替换

8、为什么要混淆

由于Java是一门跨平台的解释性语言，其源代码被编译成class字节码来适应其他平台，而class文件包含了Java源代码信息，很容易被反编译

14 ANR面试题

1、什么是ANR

Application Not Responding，页面无响应的对话框

2、发生ANR的条件

应用程序的响应性是由ActivityManager和WindowManager系统服务监视的，当ANR发生条件满足时，就会弹出ANR的对话框

- Activity超过5秒无响应
- BroadcastReceiver超过10秒无响应
- Service超过20秒无响应

3、造成ANR的主要原因

主线程被IO操作阻塞

- Activity的所有生命周期回调都是执行在主线程的
- Service默认执行在主线程中
- BoardcastReceiver的回调onReceive()执行在主线程中
- AsyncTask的回调除了doInBackground，其他都是在主线程中
- 没有使用子线程Looper的Handler的handlerMessage，post(Runnable)都是执行在主线程中

4、如何解决ANR

- 使用AsyncTask处理耗时IO操作
- 使用Thread或HandlerThread提供优先级
- 使用Handler处理工作线程的耗时操作
- Activity的onCreate和onResume回调尽量避免耗时操作

15 OOM面试题

1、什么是OOM

OOM指Out of memory（内存溢出），当前占用内存加上我们申请的内存资源超过了Dalvik虚拟机的最大内存限制就会抛出Out of memory异常

2、OOM相关概念

- 内存溢出：指程序在申请内存时，没有足够的空间供其使用
- 内存泄漏：指程序分配出去的内存不再使用，无法进行回收
- 内存抖动：指程序短时间内大量创建对象，然后回收的现象

3、解决OOM

Bitmap相关

- 图片压缩
- 加载缩略图
- 在滚动时不加载图片
- 回收Bitmap
- 使用inBitmap属性
- 捕获异常

其他相关

- listview重用convertView、使用lru
- 避免onDraw方法执行对象的创建
- 谨慎使用多进程

16 Bitmap面试题

1、recycle


- 在安卓3.0以前Bitmap是存放在堆中的，我们只要回收堆内存即可
- 在安卓3.0以后Bitmap是存放在内存中的，我们需要回收native层和Java层的内存
- 官方建议我们3.0以后使用recycle方法进行回收，该方法也可以不主动调用，因为垃圾回收器会自动收集不可用的Bitmap对象进行回收
- recycle方法会判断Bitmap在不可用的情况下，将发送指令到垃圾回收器，让其回收native层和Java层的内存，则Bitmap进入dead状态
- recycle方法是不可逆的，如果再次调用getPixels()等方法，则获取不到想要的结果

2、LruCache原理

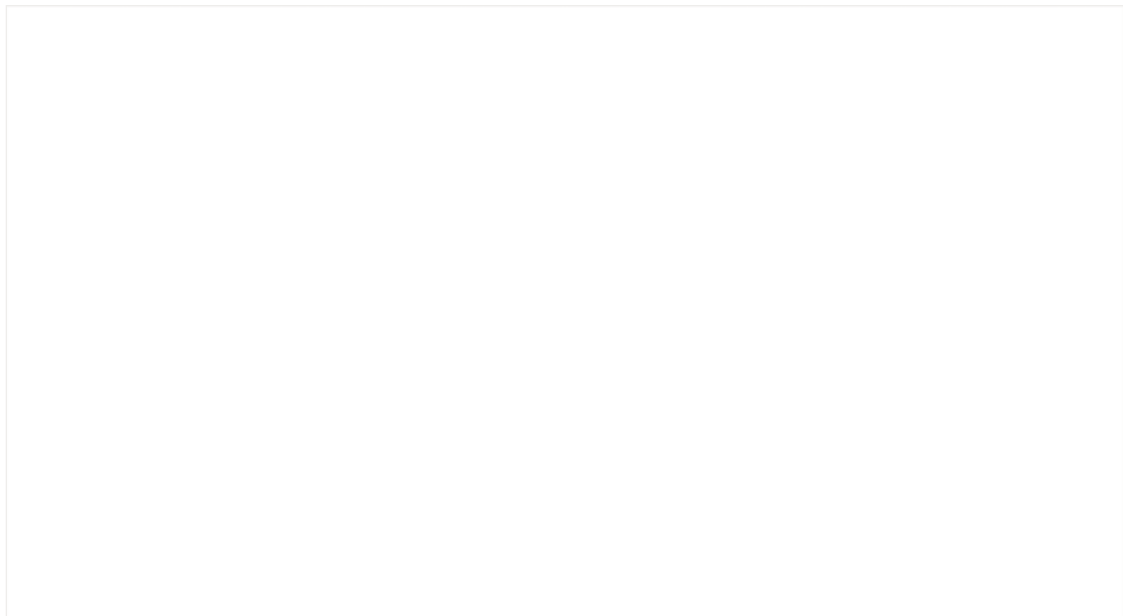
LruCache是个泛型类，内部采用LinkedHashMap来实现缓存机制，它提供get方法和put方法来获取缓存和添加缓存，其最重要的方法trimToSize是用来移除最少使用的缓存和使用最久的缓存，并添加最新的缓存到队列中

3、计算inSampleSize


```
public static int calculateInSampleSize(BitmapFactory.Options options,  
                                       int reqWidth, int reqHeight) {  
    final int height = options.outHeight;  
    final int width = options.outWidth;  
    int inSampleSize = 1;  
  
    if (height > reqHeight || width > reqWidth) {  
        if (width > height) {  
            inSampleSize = Math.round((float)height / (float)reqHeight);  
        } else {  
            inSampleSize = Math.round((float)width / (float)reqWidth);  
        }  
    }  
    return inSampleSize;  
}
```

 鸿洋

4、缩略图



5、保存Bitmap

```

public static String save(Bitmap bitmap, Bitmap.CompressFormat format,
                           int quality, File destFile) {
    try {
        FileOutputStream out = new FileOutputStream(destFile);
        if (bitmap.compress(format, quality, out)) {
            out.flush();
            out.close();
        }
        if (bitmap != null && !bitmap.isRecycled()) {
            bitmap.recycle();
        }
        return destFile.getAbsolutePath();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}

```

鸿洋

6、保存到SD卡

```

public static String save(Bitmap bitmap, Bitmap.CompressFormat format,
                           int quality, Context context) {
    if (!Environment.getExternalStorageState().equals(
        Environment.MEDIA_MOUNTED)) {
        return null;
    }
    File dir = new File(Environment.getExternalStorageDirectory()
        + "/" + context.getPackageName() + "/save/");
    if (!dir.exists()) {
        dir.mkdirs();
    }
    File destFile = new File(dir, UUID.randomUUID().toString());
    return save(bitmap, format, quality, destFile);
}

```

鸿洋

7、三级缓存

- 网络缓存
- 本地缓存
- 内存缓存

17

UI卡顿面试题

1、UI卡顿原理

View的绘制帧数保持60fps是最佳，这要求每帧的绘制时间不超过

16ms (1000/60)，如果安卓不能在16ms内完成界面的渲染，那么就会出现卡顿现象

2、UI卡顿的原因分析

- 在UI线程中做轻微的耗时操作，导致UI线程卡顿

- 布局Layout过于复杂，无法在16ms内完成渲染
- 同一时间动画执行的次数过多，导致CPU和GPU负载过重
- overDraw，导致像素在同一帧的时间内被绘制多次，使CPU和GPU负载过重
- View频繁的触发measure、layout，导致measure、layout累计耗时过多和整个View频繁的重新渲染
- 频繁的触发GC操作导致线程暂停，会使得安卓系统在16ms内无法完成绘制
- 冗余资源及逻辑等导致加载和执行缓慢
- ANR

3、UI卡顿的优化

- 布局优化
 - 使用include、ViewStub、merge
 - 不要出现过于嵌套和冗余的布局
 - 使用自定义View取代复杂的View
- ListView优化
 - 复用convertView
 - 滑动不加载
- 背景和图片优化
 - 缩略图
 - 图片压缩
- 避免ANR
 - 不要在UI线程中做耗时操作

18 内存泄漏面试题

1、Java内存泄漏引起的主要原因

长生命周期的对象持有短生命周期对象的引用就很可能发生内存泄漏

2、Java内存分配策略

- 静态存储区：又称方法区，主要存储全局变量和静态变量，在整个程序运行期间都存在
- 栈区：方法体的局部变量会在栈区创建空间，并在方法执行结束后会自动释放变量的空间和内存
- 堆区：保存动态产生的数据，如：new出来的对象和数组，在不使用的时候由Java回收器自动回收

3、Android解决内存泄漏的例子

- 单例造成的内存泄漏：在单例中，使用context.getApplicationContext()作为单例的context
- 匿名内部类造成的内存泄漏：由于非静态内部类持有匿名外部类的引用，必须将内部类设置为static
- Handler造成的内存泄漏：使用static的Handler内部类，同时在实现内部类中持有Context的弱引用
- 避免使用static变量：由于static变量会跟Activity生命周期一致，当Activity退出后台被后台回收时，static变量是不安全，所以也要管理好static变量的生命周期
- 资源未关闭造成的内存泄漏：比如Socket、Broadcast、Cursor、Bitmap、ListView等，使用完后要关闭
- AsyncTask造成的内存泄漏：由于非静态内部类持有匿名内部类的引用而造成内存泄漏，可以通过AsyncTask内部持有外部Activity的弱引用同时改为静态内部类或在onDestroy()中执行AsyncTask.cancel()进行修复

19 内存管理面试题

1、Android内存管理机制

- 分配机制
- 管理机制

2、内存管理机制的特点

- 更少的占用内存
- 在合适的时候，合理的释放系统资源
- 在系统内存紧张的时候，能释放掉大部分不重要的资源
- 能合理的在特殊生命周期中，保存或还原重要数据

3、内存优化方法

- Service完成任务后应停止它，或用IntentService（因为可以自动停止服务）代替Service
- 在UI不可见的时候，释放其UI资源
- 在系统内存紧张的时候，尽可能多的释放非重要资源
- 避免滥用Bitmap导致内存浪费
- 避免使用依赖注入框架
- 使用针对内存优化过的数据容器

- 使用ZIP对齐的APK
- 使用多进程

20 冷启动和热启动面试题

1、什么是冷启动和热启动

- 冷启动：在启动应用前，系统中没有该应用的任何进程信息
- 热启动：在启动应用时，在已有的进程上启动应用（用户使用返回键退出应用，然后马上又重新启动应用）

2、冷启动和热启动的区别

- 冷启动：创建Application后再创建和初始化MainActivity
- 热启动：创建和初始化MainActivity即可

3、冷启动时间的计算

这个时间值从应用启动（创建进程）开始计算，到完成视图的第一次绘制为止

4、冷启动流程

- Zygote进程中fork创建出一个新的进程
- 创建和初始化Application类、创建MainActivity
- inflate布局、当onCreate/onStart/onResume方法都走完
- contentView的measure/layout/draw显示在界面上

总结：Application构造方法->attachBaseContext()->onCreate()->Activity构造方法->onCreate()->配置主题中背景等属性->onStart()->onResume()->测量布局绘制显示在界面上

5、冷启动优化

- 减少第一个界面onCreate()方法的工作量
- 不要让Application参与业务的操作
- 不要在Application进行耗时操作
- 不要以静态变量的方式在Application中保存数据
- 减少布局的复杂性和深度
- 不要在mainThread中加载资源
- 通过懒加载方式初始化第三方SDK

21 其他优化面试题

1、Android不用静态变量存储数据

- 静态变量等数据由于进程已经被杀死而被初始化
- 使用其他数据传输方式：文件/sp/contentProvider

2、SharePreference安全问题

- 不能跨进程同步
- 文件不宜过大

3、内存对象序列化

- Serializeble：是java的序列化方式，Serializeble在序列化的时候会产生大量的临时对象，从而引起频繁的GC
- Parcelable：是Android的序列化方式，且性能比Serializeble高，Parcelable不能使用在要将数据存储在硬盘上的情况

4、避免在UI线程中做繁重的操作

22 架构模式面试题

- Android基础——框架模式MVC在安卓中的实践
- http://blog.csdn.net/qq_30379689/article/details/52909656
- Android基础——框架模式MVP在安卓中的实践
- http://blog.csdn.net/qq_30379689/article/details/52910567
- Android基础——框架模式MVVM之DataBinding的实践
- http://blog.csdn.net/qq_30379689/article/details/53037430

23 插件化面试题

1、插件化解决的问题

- 动态加载APK（反射、类加载器）
- 资源加载（反射、AssetManager、独立资源、分段资源）

- 代码加载（反射获取生命周期）

2、类加载器（Java中字节码添加到虚拟机中）

- DexClassLoader：能够加载未安装的jar/apk/dex，主要用于动态加载和代码热更新
- PathClassLoader：只能加载系统中已经安装过的apk

24

热更新面试题

1、热更新主要流程

- 线上检查到Crash
- 拉出Bugfix分支修复Crash问题
- jenkins构建和补丁生成
- app通过推送或主动拉取补丁文件
- 将Bugfix代码合到master上

2、热更新主流框架

- Dexposed
- AndFix
- Nuwa
- Tinker

3、热更新的原理

- 在ClassLoader创建一个dexElements数组
- 将修复好的dex文件存放在dexElements数组的最前面
- ClassLoader会遍历dexElements数组，找到最前面的dex文件优先加载

25

进程保活面试题

1、进程的优先级

- 空进程
- 后台进程
- 服务进程

- 可见进程
- 前台进程

2、Android进程回收策略

- Low memory Killer（定时执行）：通过一些比较复杂的评分机制，对进程进行打分，然后将分数高的进程判定为bad进程，杀死并释放内存
- OOM_ODJ：判别进程的优先级

3、Android保活方案

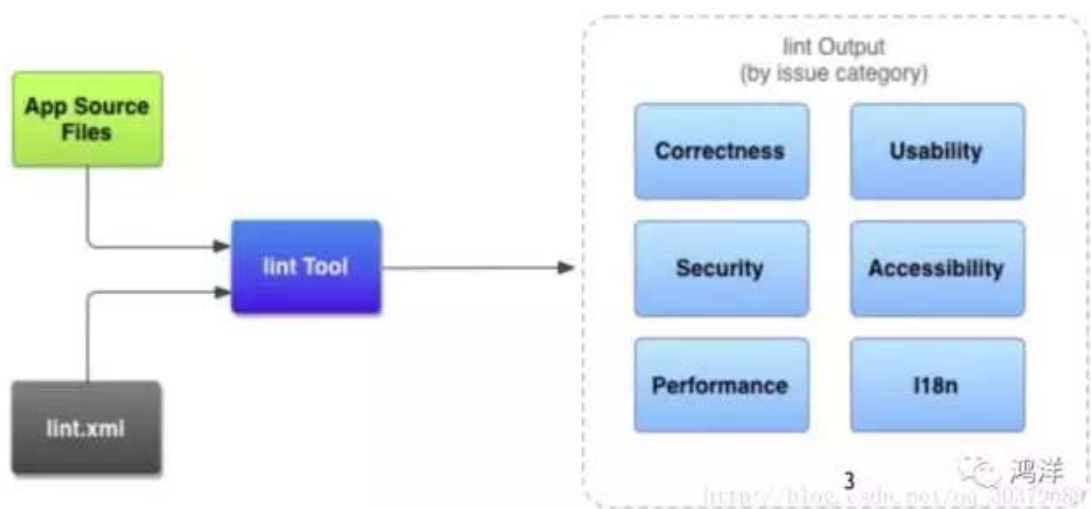
- 利用系统广播拉活
- 利用系统Service机制拉活
- 利用Native进程拉活
- 利用JobScheduler机制拉活
- 利用账号同步机制拉活

26 Lint面试题

1、什么是Android Lint

Android Lint是一个静态代码分析工具，它能够对你的Android项目中潜在的Bug、可优化的代码、安全性、性能、可用性、可访问性、国际化等进行检查

2、Lint工作流程



3、配置Lint

- 创建Lint.xml到根目录下，自定义Lint安全等级等
- 在Java文件中可以使用@suppressLint(“NewApi”)来忽视Lint的报错
- 在xml文件中可以使用tool:ignore(“UnusedResources”)来忽视Lint的报错

- 自定义Lint检查，可以创建类，继承Detector和实现JavaPsiScanner

27 Kotlin面试题

1、什么是Kotlin

- Kotlin是一种基于JVM的编程语言
- 对Java的一种拓展，比Java更简洁
- Kotlin支持函数式编程
- Kotlin类和Java类可以相互调用

2、Kotlin环境搭建

- 直接在Plugin中下载Kotlin插件即可
- 系统会自动配置到Kotlin环境

如果你有**想学习的文章**直接留言，我会整理征稿。如果你有好的文章想和大家分享欢迎投稿，直接向我投递**文章链接**即可。

欢迎**长按下图**->识别图中**二维码**或者**扫一扫**关注我的公众号：



[阅读原文](#) 阅读 15703

75

[投诉](#)

精选留言

[写留言](#)



SKink

虽然懂，但还是找不到工作

26

作者回复

早

2017年8月22日



风和白羊

早早早

2017年8月22日

作者回复

早(°Д°)/

2017年8月22日



我叫王哈哈

早

2017年8月22日

作者回复

早

2017年8月22日



刘小涛

早/☀

2017年8月22日

作者回复

早

2017年8月22日



网名

工作中要技术点来什么技术博客，要面试了来面试大纲~大爱

2017年8月22日



嗯保护你

送我书！

2017年8月22日

作者回复

昨天的已经送啦

2017年8月22日



我看完了。来评论

2017年8月22日



彭彭彭

也准备转移阵地了

2017年8月22日



小路

玩币上了瘾，没怎么来学习，点个广告，留个言，跑

2017年8月22日



嘿哈

鸿神早啊

2017年8月22日

作者回复

早

2017年8月22日



毅东

复习一把，加油(ง •̀•́)ง

2017年8月22日



攀登

早呀。。。这么准时！

2017年8月22日



aftermath

是第一吗？鸿洋大神！

2017年8月22日

作者回复

前排

2017年8月22日



大懒猫爱吃肉

前排...早

2017年8月22日

作者回复

早

2017年8月22日



有味
早上好

2017年8月22日



邓焦
早

2017年8月22日

作者回复

早

2017年8月22日



别问云
早~

2017年8月22日

作者回复

早

2017年8月22日



尼蒙
地铁日常打卡

2017年8月22日



Jowan
收藏学习啦，昨天忘记留言了

2017年8月22日



$r=a(1-\sin \theta)$ 嘉宾
早，大神

2017年8月22日

作者回复

早

2017年8月22日

以上留言由公众号筛选后显示

[了解留言功能详情](#)