### 源码分析相关面试题

* [Volley源码分析](http://www.jianshu.com/p/ec3dc92df581)
* [注解框架实现原理](http://www.jianshu.com/p/20da6d6389e1)
* [okhttp3.0源码分析](http://www.jianshu.com/p/9ed2c2f2a52c)
* [onSaveInstanceState源码分析](http://www.jianshu.com/p/cbf9c3557d64)
* [静默安装和源码编译](http://www.jianshu.com/p/2211a5b3c37f)

### Activity相关面试题

* [保存Activity的状态](http://www.jianshu.com/p/cbf9c3557d64)

### 与XMPP相关面试题

* [XMPP协议优缺点](http://www.jianshu.com/p/2c04ac3c526a)
* [极光消息推送原理](http://www.jianshu.com/p/d88dc66908cf)

### 与性能优化相关面试题

* [内存泄漏和内存溢出区别](http://www.jianshu.com/p/5dd645b05c76)
* [UI优化和线程池实现原理](http://www.jianshu.com/p/c22398f8587f)
* [代码优化](http://www.jianshu.com/p/ebd41eab90df)
* [内存性能分析](http://www.jianshu.com/p/2665c31b9c2f)
* [内存泄漏检测](http://www.jianshu.com/p/1514c7804a06)
* [App启动优化](http://www.jianshu.com/p/f0f73fefdd43)
* [与IPC机制相关面试题](http://www.jianshu.com/p/de4793a4c2d0)

### 与登录相关面试题

* [oauth认证协议原理](http://www.jianshu.com/p/2a6ecbf8d49d)
* [token产生的意义](http://www.jianshu.com/p/9b7ce2d6c195)
* [微信扫一扫实现原理](http://www.jianshu.com/p/a9d1f21bd5e0)

### 与开发相关面试题

* [迭代开发的时候如何向前兼容新旧接口](http://www.jianshu.com/p/cbecadec98de)
* [手把手教你如何解决as jar包冲突](http://www.jianshu.com/p/30fdc391289c)
* [context的原理分析](http://www.jianshu.com/p/2706c13a1769)
* [解决ViewPager.setCurrentItem中间很多页面切换方案](http://www.jianshu.com/p/38ab6d856b56)

### 与人事相关面试题

* [人事面试宝典](http://www.jianshu.com/p/d61b553ff8c9)

### 本文配套视频

* [配套视频](https://v.qq.com/x/page/w0392bn6wto.html)
* [配套视频](https://v.qq.com/x/page/j0393ytx9ob.html)
* [线程池原理配套视频](https://v.qq.com/x/page/u0393izwfut.html)

11-说一下如何对Android内存优化上集？

## Bitmap OOM（图片优化）

### 1. 图片处理

#### a． 等比缩放



img

以上代码可以优化内存溢出，但它只是改变图片大小，并不能彻底解决内存溢出。

#### b． 对图片采用软引用，及时地进行recyle()操作



img

#### c． 设置图片解码格式

BitmapFactory.Options options = new BitmapFactory.Options();
options.inPreferredConfig = Bitmap.Config.ARGB\_4444;

#### d． 加载部分图片



img

#### 2. 图片的缓存

1. 网络缓存
2. 内存缓存
3. 磁盘缓存

#### 3. 使用加载图片框架处理图片，如专业处理加载图片的ImageLoader，glide，picasso等图片加载框架。

通过以上方式可以解决图片OOM异常

## UI Review（UI优化）

减少视图层级 减少视图层级可以有效的减少内存消耗，因为视图是一个树形结构，每次刷新和渲染都会遍历一次。

### ViewStub标签

此标签可以使UI在特殊情况下，直观效果类似于设置View的不可见性，但是其更大的意义在于被这个标签所包裹的Views在默认状态下不会占用任何内存空间。

<ViewStub
 android:id="@+id/mystub"
 android:layout\_width="wrap\_content"
 android:layout\_height="wrap\_content"
 android:layout="@layout/common\_msg" >
 </ViewStub>

### include标签

可以通过这个标签直接加载外部的xml到当前结构中，是复用UI资源的常用标签。

<include
 layout="@layout/activity\_main" />

### merge标签

它在优化UI结构时起到很重要的作用。目的是通过删减多余或者额外的层级，从而优化整个Android Layout的结构。

<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android"
 >

 <ImageView
 android:layout\_width="wrap\_content"
 android:layout\_height="wrap\_content"
 android:background="@drawable/ic\_launcher"/>

 <TextView
 android:layout\_width="match\_parent"
 android:layout\_height="wrap\_content"
 android:layout\_marginLeft="50dp"
 android:layout\_marginTop="10dp"
 android:textSize="20sp"
 android:text="我是小黑马"
 />

</merge>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout\_width="match\_parent"
 android:layout\_height="match\_parent"
 android:orientation="horizontal" >

 <ImageView
 android:layout\_width="wrap\_content"
 android:layout\_height="wrap\_content"
 android:background="@drawable/ic\_launcher" />

 <TextView
 android:layout\_width="match\_parent"
 android:layout\_height="wrap\_content"
 android:layout\_marginLeft="50dp"
 android:layout\_marginTop="10dp"
 android:text="我是小黑马"
 android:textSize="20sp" />

</LinearLayout>



img

实现一模一样的效果，我们通过SDK自带工具检测android层级关系

* 使用merge标签，布局少一个层级



img

* 不使用merge标签，布局多一个层级



img

Fragment优化 直接使用系统APP包下面的Fragment，不要使用V4包里面的Fragment可以减少层级结构。

## 使用ThreadPool优化代码（线程池实现原理）

### 1. new Thread的弊端

执行一个异步任务你还只是如下new Thread吗？

new Thread的弊端如下：

* 每次new Thread新建对象性能差。
* 线程缺乏统一管理，可能无限制新建线程，相互之间竞争，及可能占用过多系统资源导致死机或oom。
* 缺乏更多功能，如定时执行、定期执行、线程中断。

相比new Thread，Java提供的四种线程池的好处在于：

* 重用存在的线程，减少对象创建、消亡的开销，性能佳。
* 可有效控制最大并发线程数，提高系统资源的使用率，同时避免过多资源竞争，避免堵塞。
* 提供定时执行、定期执行、单线程、并发数控制等功能。

### 2. Java 线程池

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。

* 欢迎关注微信公众号,长期推荐技术文章和技术视频

微信公众号名称：Android干货程序员

img

img