### 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/n0391if5dtb.html)
* [配套视频](https://v.qq.com/x/page/q03917e4zk5.html)
* [配套视频](https://v.qq.com/x/page/j03927ullcj.html)

### 9-内存泄漏和内存溢出分别是什么？它们有什么关系？

* 内存泄露是指保存了不可能再被访问的变量引用，导致垃圾回收器无法回收内存。 也就是说：在Java中有些对象的生命周期是有限的，当它们完成了特定的逻辑后将会被垃圾回收；但是，如果在对象的生命周期本来该被垃圾回收时这个对象还被别的对象所持有引用，那就会导致内存泄漏
* 内存溢出是指虚拟机内存耗尽，无法为新对象分配内存，导致引用崩溃。典型的情况为加载多张大图，导致内存耗尽。



img

* 当某个界面存在内存泄露，反复进入该界面，将导致一直有新对象创建但是无法回收，最终内存耗尽，产生内存溢出。

### 10-什么情况下会导致内存泄漏

* 资源释放问题 程序代码的问题，长期保持某些资源，如Context、Cursor、IO流的引用，资源得不到释放造成内存泄露。
* 对象内存过大问题 保存了多个耗用内存过大的对象（如Bitmap、XML文件），造成内存超出限制。
* static关键字的使用问题 static是Java中的一个关键字，当用它来修饰成员变量时，那么该变量就属于该类，而不是该类的实例。所以用static修饰的变量，它的生命周期是很长的，如果用它来引用一些资源耗费过多的实例（Context的情况最多），这时就要谨慎对待了。 针对static的解决方案：
1. 应该尽量避免static成员变量引用资源耗费过多的实例，比如Context。
2. Context尽量使用ApplicationContext，因为Application的Context的生命周期比较长，引用它不会出现内存泄露的问题。
3. 使用WeakReference代替强引用。比如可以使用WeakReference mContextRef;
* 线程导致内存溢出 线程产生内存泄露的主要原因在于线程生命周期的不可控。 针对这种线程导致的内存泄露问题的解决方案：
* 将线程的内部类，改为静态内部类（因为非静态内部类拥有外部类对象的强引用，而静态类则不拥有）。
* 在线程内部采用弱引用保存Context引用。
* 查询数据库没有关闭cursor 程序中经常会进行查询数据库的操作，但是经常会有使用完毕Cursor后没有关闭的情况。如果我们的查询结果集比较小，对内存的消耗不容易被发现，只有在常时间大量操作的情况下才会出现内存问题，这样就会给以后的测试和问题排查带来困难和风险。
* 构造Adapter没有复用convertview 在使用ListView的时候通常会使用Adapter，那么我们应该尽可能的使用ConvertView。 为什么要复用convertView? 当convertView为空时，用setTag()方法为每个View绑定一个存放控件的ViewHolder对象。当convertView不为空，重复利用已经创建的view的时候，使用getTag()方法获取绑定的ViewHolder对象，这样就避免了findViewById对控件的层层查询，而是快速定位到控件。
* Bitmap不再使用时没有调用recycle()释放内存

有时我们会手工的操作Bitmap对象，如果一个Bitmap对象比较占内存，当它不再被使用的时候，可以调用Bitmap.recycle()方法回收此对象的像素所占用的内存，但这不是必须的，视情况而定。

* 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