Android内存泄漏总结

https://mp.weixin.qq.com/s?__biz=MzA5MzI3NjE2MA==&mid=2650242747&idx=1&sn=88c4d040c439c7db13033e5be18440b7&chksm=88638fd4bf1406c25fa75c04c5853281facd65afad24c57797cf1c2fe3719168810a2bdfd8ea&scene=38#wechat_redirect

Android内存泄漏总结

原创： 08_carmelo 郭霖 4月9日

今日科技快讯

国家网信办近日依法约谈“快手”和今日头条旗下“火山小视频”相关负责人，提出严肃批评，责令全面进行整改。相关负责人表示完全接受处罚，暂停更新有关频道5天，禁止未满18周岁的未成年人注册网络主播，已有账号一律关停，进一步完善审核管理机制，建立未成年人保护体系，用正确的价值观指导算法，积极传播正能量。

作者简介

本篇来自 **08_carmelo** 的投稿，分享了他对 Androidandroid内存泄漏的整理和总结，一起来看看！希望大家喜欢。

08_carmelo 的博客地址：

https://www.jianshu.com/u/b8dad3885e05

前言

先通俗理解下内存泄漏，内存溢出，OOM，GC回收这几个概念。把app的堆内存空间想成了一个杯子，内存就是里面的水。当你的app启动后，系统会分配给app一个堆空间，起始不会很大比如是32M（根据你的app启动时的内存申请为准）

image.png

随着程序的运行对象的创建越来越多，系统不断加内存分配：32M -> 64M -> ...

而GC回收则会定时扫描内存，发现不被引用的对象即可回收。正常来说你的app堆内存会有升有降。此时如果有某个Activity持有某个引用，在onDestroy时还不把这个引用设为null，那么返回进入退出这个界面，Activity就会创建很多次从而存在多个实例，导致堆内存直升不降！这就叫做内存泄漏。

当用户重复这个操作或者有多个不同Activity内存泄漏时，app运行一段时间堆内存超过系统规定的最大值 heapSize，杯子满了就会发现内存溢出（OOM），app崩溃。

正文

关键点

通过上面这个例子，我们知道查找内存泄漏有如下几点关键点：

- 如何知道你的app上限值heapSize是多少
- 什么情况导致无法GC
- 怎么复现是哪个界面内存泄漏

下面通过一个实例来演示，如何借助AndroidStudio查找内存泄漏：

内存泄漏实例

当你的app在使用中莫名崩溃，如果是OOM那么会有如下日志：

image.png

接下来我们用下面的代码获取heapsize：

```
ActivityManager manager = (ActivityManager)getSystemService(Context.ACTIVITY_SERVICE);
int heapSize = manager.getMemoryClass();
int maxHeapSize = manager.getLargeMemoryClass();  // manafest.xml   android:largeHeap="true"
```

- heapSize是设备分配给app的最大堆内存
- maxHeapSize 是当配置了android:largeHeap="true" 才有的最大堆内存，一般是heapSize的2-3倍

以HTC ONE为例，两个值分别是： 192M和512M。这里我只关注192M，maxHeapSize放到后面再说。然后尝试复现问题，手机连接AndroidStudio打开monitor，反复进入/退出怀疑内存泄漏的界面：

image.png

如果发现内存一直上升，并且到接近192M的时候不动了，此时已经OOM只不过不一定会崩溃（oom异常可以try catch）接着，用AndroidStudio自带的内存分析工具分析，点击Dump Java Heap:

image.png

（备注：ViewDefectPhotoActivity是我的app里面一个界面，用ViewPager展示，ImageLoader加载很多照片）

会在代码区 生成当前时间点的 .hprof格式文件，里面当前的每个对象 内存占用情况，我们现在怀疑ViewDefectPhotoActivity 可能内存泄漏 看一下：ViewDefectPhotoActivity 确实占用了很高的内存 而且有8个实例化对象,足以证明它有内存泄漏，随便点击其中一个，发现都和EditDefectActivity的内部类 ShowEditDefectHelper有关。而且ShowEditDefectHelpe前面带有 黄蓝三角树 这个图标表示，他可以被GC访问到，也就是无法回收：

image.png

接下来看看ShowEditDefectHelper这个对象持有什么引用，右键点击 Go To Instance:

image.png

发现这个内部类也被实例化多次，随便点击一个，发现属于EventBus的引用 ，并且EventBus带有黄蓝图标，那么问题就大概找到了，因为EventBus一直持有这个内部类的引用，导致这个内部类无法被回收，而这个内部类持有ViewDefectPhotoActivity引用，导致ViewDefectPhotoActivity无法被回收 这个界面有很多图片资源 当实例化7,8次之后 出现了OOM。

EventBus是用于事件通知的开源库，应该很多人都了解。它需要在某个类里面绑定和解绑定，我这里是在ShowEditDefectHelper注册的，看下Activity的内部类ShowEditDefectHelper：

image.png

解释下这里的逻辑：进入这个activity我会显示一个进度条，请求网络数据，用户可以随时取消进度条。然后onCancel里面对EventBus解绑定：

但是这里有个低级错误：EventBus.getDefult().unregister(this)传入的this是onCancelled()的匿名内部类而不是ShowEditDefectHelper.class导致EventBus无法解绑！！

所以应该这么解绑：

```
EventBus.getDefult().unregister(ShowEditDefectHelper.Class);
```

通过这个示例，我们回答了上面前两个关键点：

1. 如何知道你的app上限值heapSize是多少
2. 什么情况导致无法GC

第三个关键点：

怎么复现是哪个界面内存泄漏。内存泄漏不是一眼就能看出来了，需要测试人员配合。当然还有一个办法就是facebook的开源库：leakcanary，具体使用我不多介绍了。它是一个apk安装在手机上可以直接列出内存泄漏的Activity，但是有一定误报几率：

image.png

我更喜欢leakcanary + AndroidStudio的方式，精确无误地找出问题。

怎么避免内存

一句话归纳：（生命周期比Activity长的类不要去强引用Activity）

1. 内部类请使用static，因为非静态内部类默认持有外部类的引用，比如在Activity里面直接放一个自定义的Adapter
2. 静态类（比如Application，单例类，其他static类）请不要持有Activity引用，因为静态类生命周期比Activity长。解决办法：在需要的地方用BaseApplication.getTopActivity。或者Activity作为弱引用传入

3. 注意Handler会默认持有当前Activity，用的时候最好不要直接new Handler().post(new Runnable...),除非你确定这个runnable会在Activity销毁前执行完

OK，假设你的项目比较紧急，想暂时规避内存泄漏问题怎么办？可以在manifest.xml加入本文开头给的那个设置：

```
android:largeHeap="true"
```

此时heapsize会增大2-3倍，缓解OOM的发生，但是技术债终究要还的。希望大家认真揣摩本文用AndroidStudio分析泄漏的方法，而不是过分依赖leakcanary这个开源库。