

## [Android] Android开发优化之——对Bitmap的内存优化

原创 2012年09月07日 09:51:04 标签： android / outofmemoryerror / exception / null / jni / crash

45182

在Android应用里，最耗费内存的就是图片资源。而且在Android系统中，读取位图Bitmap时，分给虚拟机中的图片的堆栈大小只有8M，如果超出了，就会出现OutOfMemory异常。所以，对于图片的内存优化，是Android应用开发中比较重要的内容。

### 1) 要及时回收Bitmap的内存

Bitmap类有一个方法recycle()，从方法名可以看出意思是回收。这里就有疑问了，Android系统有自己的垃圾回收机制，可以不定期的回收掉不使用的内存空间，当然也包括Bitmap的空间。那为什么还需要这个方法呢？

Bitmap类的构造方法都是私有的，所以开发者不能直接new出一个Bitmap对象，只能通过BitmapFactory类的各种静态方法来实例化一个Bitmap。仔细查看BitmapFactory的源代码可以看到，生成Bitmap对象最终都是通过JNI调用方式实现的。所以，加载Bitmap到内存里以后，是包含两部分内存区域的。简单的说，一部分是Java部分的，一部分是C部分的。这个Bitmap对象是由Java部分分配的，不用的时候系统就会自动回收了，但是那个对应的C可用的内存区域，虚拟机是不能直接回收的，这个只能调用底层的功能释放。所以需要调用recycle()方法来释放C部分的内存。从Bitmap类的源代码也可以看到，recycle()方法里也的确是调用了JNI方法了。

那如果不调用recycle()，是否就一定存在内存泄露呢？也不是的。Android的每个应用都运行在独立的进程里，有着独立的内存，如果整个进程被应用本身或者系统杀死了，内存也就都被释放掉了，当然也包括C部分的内存。

Android对于进程的管理是非常复杂的。简单的说，Android系统的进程分为几个级别，系统会在内存不足的情况下杀死一些低优先级的进程，以提供给其它进程充足的内存空间。在实际项目开发过程中，有的开发者会在退出程序的时候使用Process.killProcess(Process.myPid())的方式将自己的进程杀死，但是有的应用仅仅会使用调用Activity.finish()方法的方式关闭掉所有的Activity。

#### 经验分享：

Android手机的用户，根据习惯不同，可能会有两种方式退出整个应用程序：一种是按Home键直接退到桌面；另一种是从应用程序的退出按钮或者按Back键退出程序。那么从系统的角度来说，这两种方式有什么区别呢？按Home键，应用程序并没有被关闭，而是成为了后台应用程序。按Back键，一般来说，应用程序关闭了，但是进程并没有被杀死，而是成为了空进程（程序本身对退出做了特殊处理的不考虑在内）。

Android系统已经做了大量进程管理的工作，这些已经可以满足用户的需求。个人建议，应用程序在退出应用的时候不需要手动杀死自己所在的进程。对于应用程序本身的进程管理，交给Android系统来处理就可以了。应用程序需要做的，是尽量做好程序本身的内存管理工作。

一般来说，如果能够获得Bitmap对象的引用，就需要及时的调用Bitmap的recycle()方法来释放Bitmap占用的内存空间，而不要等Android系统来进行释放。

下面是释放Bitmap的示例代码片段。

```
// 先判断是否已经回收
if(bitmap != null && !bitmap.isRecycled()){
    // 回收并且置为null
    bitmap.recycle();
    bitmap = null;
}
System.gc();
```



要什么昵称嘛

原创 411 粉丝 1664 喜欢 2

等级： 博客 访问量：190

积分：2万+ 排名：472



北大青鸟



### 他的最新文章

第九章 多语言环境的支持和多屏幕配（3）

第九章 多语言环境的支持和多屏幕配（2）

第九章 多语言环境的支持和多屏幕配（1）

第八章 网络的时代—网络开发（4）

第八章 网络的时代—网络开发（3）

### 文章分类

01 J2SE

04 OSGI

02 J2ME

03 J2EE

05 Open Source

06 Cloud Computing

展开

### 文章存档

2016年3月

2016年1月

2015年7月


2015年6月

2015年5月

2015年2月

展开

从上面的代码可以看到，`bitmap.recycle()`方法用于回收该Bitmap所占用的内存，接着将bitmap置空，最后使用`System.gc()`调用一下系统的垃圾回收器进行回收，可以通知垃圾回收器尽快进行回收。这里需要注意的是，调用`System.gc()`并不能保证立即开始进行回收过程，而只是为了加快回收的到来。

如何调用`recycle()`方法进行回收已经了解了，那什么时候释放Bitmap的内存比较合适呢？一般来说，如果代码已经不再需要使用Bitmap对象了，就可以释放了。释放内存以后，就不能再使用该Bitmap对象了，如果再次使用，就会抛出异常。 一定要保证不再使用的时候释放。比如，如果是在某个Activity中使用Bitmap，就可以在Activity的`onStop()`或者`onDestroy()`方法中进行回收。



## 2) 捕获异常



因为Bitmap是吃内存大户，为了避免应用在分配Bitmap内存的时候出现OutOfMemory异常以后Crash掉，需要特别注意实例化Bitmap的代码。通常，在实例化Bitmap的代码中，一定要对OutOfMemory异常进行捕获。



以下是代码示例。



```
Bitmap bitmap = null;
try {
    // 实例化Bitmap
    bitmap = BitmapFactory.decodeFile(path);
} catch (OutOfMemoryError e) {
    //
}
if (bitmap == null) {
    // 如果实例化失败 返回默认的Bitmap对象
    return defaultBitmapMap;
}
```

这里对初始化Bitmap对象过程中可能发生的OutOfMemory异常进行了捕获。如果发生了OutOfMemory异常，应用不会崩溃，而是得到了一个默认的Bitmap图。

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！ 

很多开发者会习惯性的在代码中直接捕获Exception。但是对于OutOfMemoryError来说，这样做是捕获不到的。因为OutOfMemoryError是一种Error，而不是Exception。在此仅仅做一下提醒，避免写错代码而捕获不到OutOfMemoryError。

## 3) 缓存通用的Bitmap对象

有时候，可能需要在在一个Activity里多次用到同一张图片。比如一个Activity会展示一些用户的头像列表，而如果用户没有设置头像的话，则会显示一个默认头像，而这个头像是位于应用程序本身的资源文件中的。

如果有类似上面的场景，就可以对同一Bitmap进行缓存。如果不进行缓存，尽管看到的是同一张图片文件，但是使用BitmapFactory类的方法来实例化出来的Bitmap，是不同的Bitmap对象。缓存可以避免新建多个Bitmap对象，避免内存的浪费。

经验分享：

Web开发者对于缓存技术是很熟悉的。其实在Android应用开发过程中，也会经常使用缓存的技术。这里所说的缓存有两个级别，一个是硬盘缓存，一个是内存缓存。比如说，在开发网络应用过程中，可以将一些从网络上获取的数据保存到SD卡中，下次直接从SD卡读取，而不从网络中读取，从而节省网络流量。这种方式就是硬盘缓存。再比如，应用程序经常会使用同一对象，也可以放到内存中缓存起来，需要的时候直接从内存中读取。这种方式就是内存缓存。

## 他的热门文章

Android中dp和px之间进行转换

📖 234738

[Android] Android开发优化之一——软引用和弱引用

📖 54458

[Android] Android开发优化之一——map的内存优化

📖 45165

Android中如何做到Service被关闭启动

📖 43284

Android中ViewGroup等容器控件

📖 29048

Apache License Version 2.0 英文中文翻译

📖 28434

网页页面浮动窗口示例代码（Java Code）

📖 27004

Android官方网站及镜像地址

📖 22791

Android中Webview使用自定义的ript进行回调

📖 21301

[Android] Android开发优化之一——面UI的优化（1）

📖 20807



## 联系我们



请扫描二维码联系

✉ webmaster@c

☎ 400-660-0108

👤 QQ客服 🗣 客

关于 招聘 广告服务 🐾

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

#### 4) 压缩图片

如果图片像素过大，使用BitmapFactory类的方法实例化Bitmap的过程中，需要大于8M的内存空间，就必定会发生OutOfMemory异常。这个时候该如何处理呢？如果有这种情况，则可以将图片缩小，以减少载入图片过程中的内存的使用，避免异常发生。

使用BitmapFactory.Options设置inSampleSize就可以缩小图片。属性值inSampleSize表示缩略图大小为原始图片大小的几分之一。即如果这个值为2，则取出的缩略图的宽和高都是原始图片的1/2，图片的大小就为原始大小的1/4。

如果知道图片的像素过大，就可以对其进行缩小。那么如何才知道图片过大呢？

使用BitmapFactory.Options设置inJustDecodeBounds为true后，再使用decodeFile()等方法，并不会真正的分配空间，即解码出来的Bitmap为null，但是可计算出原始图片的宽度和高度，即options.outWidth和options.outHeight。通过这两个值，就可以知道图片是否过大了。

```
BitmapFactory.Options opts = new BitmapFactory.Options();
// 设置inJustDecodeBounds为true
opts.inJustDecodeBounds = true;
// 使用decodeFile方法得到图片的宽和高
BitmapFactory.decodeFile(path, opts);
// 打印出图片的宽和高
Log.d("example", opts.outWidth + " , " + opts.outHeight);
```

在实际项目中，可以利用上面的代码，先获取图片真实的宽度和高度，然后判断是否需要跑缩小。如果不需要缩小，设置inSampleSize的值为1。如果需要缩小，则动态计算并设置inSampleSize的值，对图片进行缩小。需要注意的是，在下次使用BitmapFactory的decodeFile()等方法实例化Bitmap对象前，别忘记将opts.inJustDecodeBound设置回false。否则获取的bitmap对象还是null。

经验分享：

如果程序的图片的来源都是程序包中的资源，或者是自己服务器上的图片，图片的大小是开发者可以调整的，那么一般来说，就只需要注意使用的图片不要过大，并且注意代码的质量，及时回收Bitmap对象，就能避免OutOfMemory异常的发生。

如果程序的图片来自外界，这个时候就特别需要注意OutOfMemory的发生。一个是如果载入的图片比较大，就需要先缩小；另一个是一定要捕获异常，避免程序Crash。

优化系列相关博文：

Android开发优化之——对Bitmap的内存优化

Android开发优化之——使用软引用和弱引用

Android开发优化之——从代码角度进行优化

Android开发优化之——对界面UI的优化（1）

Android开发优化之——对界面UI的优化（2）

Android开发优化之——对界面UI的优化（3）

---

<http://blog.csdn.net/arui319>

《Android应用开发精解》已出版，本文是初稿的部分内容。欢迎购买阅读。