**图片缓存相关的内存优化**

# Bitmap

Bitmap是一个极容易消耗内存的大胖子，减小创建出来的Bitmap的内存占用可谓是重中之重，通常来说有以下2个措施：

* 缩放比例，在把图片载入内存之前，我们需要先计算出一个合适的缩放比例，避免不必要的大图载入。
* 解码格式，选择ARGB\_8888/RBG\_565/ARGB\_4444/ALPHA\_8，存在很大差异。
* 在ListView与GridView等显示大量图片的控件里，需要使用LRU的机制来缓存处理好的Bitmap.
* 使用弱引用，弱引用的对象拥有更短暂的生命周期。在垃圾回收器线程扫描它 所管辖的内存区域的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。
* 一张522\*686的 PNG 图片，我把它放到 drawable-xxhdpi 目录下，在三星s6上加载，占用内存2547360B

# ImageView设置操作

# setImageBitmap()执行过程



# setImageBitmap和setImageDrawable的区别

实际上setImageBitmap做的事情就是把Bitmap对象封装成Drawable对象，然后调用setImageDrawable来设置图片。如果需要频繁调用这个方法的话最好自己封装个固定的Drawable对象，直接调用setImageDrawable，这样可以减少Drawable对象。因为每次调用setImageBitmap方法都会对Bitmap对象new出一个Drawable。setImageDrawable参数是Drawable,也是可以接受不同来源的图片，方法中所做的事情就是更新ImageView的图片。上面两个方法实际上最后调用的都是setImageDrawable，所以综合来看setImageDrawable是最省内存高效的，如果担心图片过大或者图片过多影响内存和加载效率，可以自己解析图片然后通过调用setImageDrawable方法进行设置。

# PackageManager加载图标操作



# 加载过程



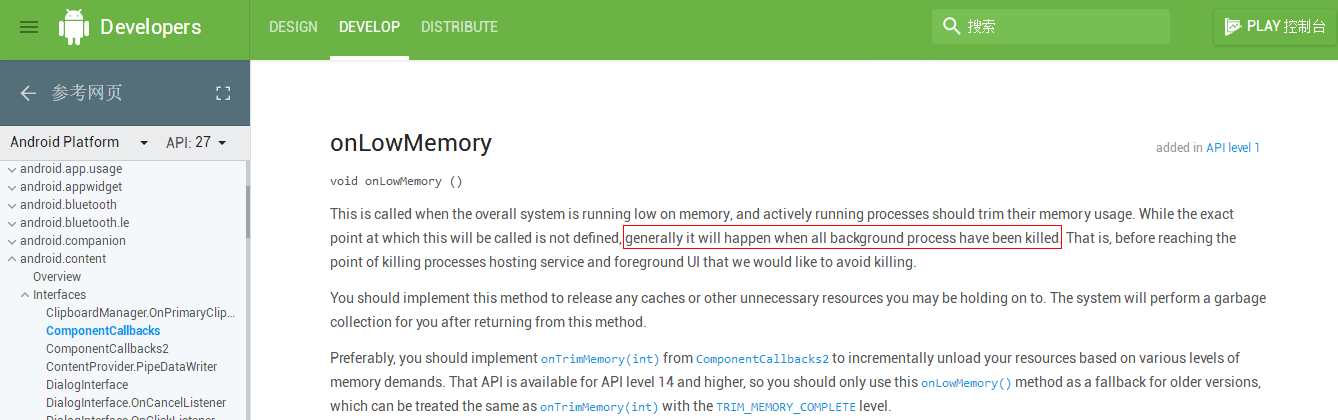
# Drawable和Bitmap的关系图：



# 优化办法

# 主动释放

* Android 系统提供了一些回调来通知当前应用的内存使用情况，通常来说当所有的 Background 应用都被 kill 掉的时候，Forground 应用会收到 onLowMemory() 的回调，在这种情况下需要尽快释放当前应用的非必须的内存资源，从而确保系统能够继续稳定运行。



* Android 系统从 4.0 开始还提供了 onTrimMemory() 的回调，当系统内存达到某些条件的时候，所有正在运行的应用都会收到这个回调，同时在这个回调里面会传递指定的参数，代表不同的内存使用情况，收到 onTrimMemory() 回调的时候，需要根据传递的参数类型进行判断，合理的选择释放自身的一些内存占用，一方面可以提高系统的整体运行流畅度，另外也可以避免自己被系统判断为优先需要杀掉的应用，
* 按时发因为 onTrimMemory() 的回调是在 API 14 才被加进来的，对于老的版本，你可以使用 onLowMemory 回调来进行兼容，onLowMemory 相当与 TRIM\_MEMORY\_COMPLETE。
* [参考链接1](https://developer.android.com/topic/performance/memory.html)。

# 弱引用

在GC的时候，会回收弱引用指向的堆区内存。在android有下列5钟情况会触发GC：

* 当堆内存增长到一定程度时会触发。此时触发可以对堆中的没有用的对象及时进行回收，腾出空间供新的对象申请，避免进行不必要的增大堆内存的操作。
* 当程序中调用System.gc()方法触发。这个方法应避免出现在程序中调用。因为JVM有足够的能力来控制垃圾回收。
* 当Bitmap和NIO Direct ByteBuffer对象分配外部存储（机器内存，非Dalvik堆内存）触发。这个日志只有在2.3之前存在，从2.3系统开始，垃圾回收进行了调整，前面的对象都会存储到Dalivik堆内存中。所以在2.3系统之后，你就再也不会看到这种信息了。
* 当堆内存已满，系统需要更多内存的时候触发。这条日志出现后意味着JVM要暂停你的程序进行垃圾回收操作。
* 当创建一个内存分析文件HPROF时触发。

# LRU

# 压缩图片

[参考链接1](http://blog.chinaunix.net/uid-26930580-id-4138306.html)。[参考链接2](http://www.heqiangfly.com/2017/06/15/android-knowledge-point-drawable-cache/)。[参考链接3](http://blog.csdn.net/qq_23547831/article/details/51465071)。[参考链接4](http://blog.csdn.net/whb20081815/article/details/70243105)。[参考链接5](https://mp.weixin.qq.com/s/GkPrmlNm8p3fkeh4vo3Htg)。