Android代码内存优化建议-OnTrimMemory优化

androidperformance.com

OnTrimMemory 回调是 Android 4.0 之后提供的一个API，这个 API 是提供给开发者的，它的主要作用是提示开发者在系统内存不足的时候，通过处理部分资源来释放内存，从而避免被 Android 系统杀死。这样应用在下一次启动的时候，速度就会比较快。

本文通过问答的方式，从各个方面来讲解 OnTrimMemory 回调的使用过程和效果。想要开发高性能且用户体验良好的 Android 应用，那么这篇文章你不应该错过。

0. OnTrimMemory回调的作用？

OnTrimMemory是Android在4.0之后加入的一个回调，任何实现了ComponentCallbacks2接口的类都可以重写实现这个回调方法．OnTrimMemory的主要作用就是**指导应用程序在不同的情况下进行自身的内存释放，以避免被系统直接杀掉，提高应用程序的用户体验**.

Android系统会根据不同等级的内存使用情况，调用这个函数，并传入对应的等级：

- TRIM_MEMORY_UI_HIDDEN 表示应用程序的**所有UI界面**被隐藏了，即用户点击了Home键或者Back键导致应用的UI界面不可见．这时候应该释放一些资源．TRIM_MEMORY_UI_HIDDEN这个等级比较常用，和下面六个的关系不是很强，所以单独说．

下面三个等级是当我们的应用程序真正运行时的回调：

- TRIM_MEMORY_RUNNING_MODERATE 表示应用程序正常运行，并且不会被杀掉。但是目前手机的内存已经有点低了，系统可能会开始根据LRU缓存规则来去杀死进程了。
- TRIM_MEMORY_RUNNING_LOW 表示应用程序正常运行，并且不会被杀掉。但是目前手机的内存已经非常低了，我们应该去释放掉一些不必要的资源以提升系统的性能，同时这也会直接影响到我们应用程序的性能。
- TRIM_MEMORY_RUNNING_CRITICAL 表示应用程序仍然正常运行，但是系统已经根据LRU缓存规则杀掉了大部分缓存的进程了。这个时候我们应当尽可能地去释放任何不必要的资源，不然的话系统可能会继续杀掉所有缓存中的进程，并且开始杀掉一些本来应当保持运行的进程，比如说后台运行的服务。

当应用程序是缓存的，则会收到以下几种类型的回调：

- TRIM_MEMORY_BACKGROUND 表示手机目前内存已经很低了，系统准备开始根据LRU缓存来清理进程。这个时候我们的程序在LRU缓存列表的最近位置，是不太可能被清理掉的，但这时去释放掉一些比较容易恢复的资源能够让手机的内存变得比较充足，从而让我们的程序更长时间地保留在缓存当中，这样当用户返回我们的程序时会感觉非常顺畅，而不是经历了一次重新启动的过程。
- TRIM_MEMORY_MODERATE 表示手机目前内存已经很低了，并且我们的程序处于LRU缓存列表的中间位置，如果手机内存还得不到进一步释放的话，那么我们的程序就有被系统杀掉的风险了。
- TRIM_MEMORY_COMPLETE 表示手机目前内存已经很低了，并且我们的程序处于LRU缓存列表的最边缘位置，系统会最优先考虑杀掉我们的应用程序，在这个时候应当尽可能地把一切可以释放的东西都进行释放。

1. 哪些组件可以实现OnTrimMemory回调？

- Application.onTrimMemory()
- Activity.onTrimMemory()
- Fragement.OnTrimMemory()
- Service.onTrimMemory()
- ContentProvider.OnTrimMemory()

2. OnTrimMemory回调中可以释放哪些资源？

通常在架构阶段就要考虑清楚，我们有哪些东西是要常驻内存的，有哪些是伴随界面存在的．一般情况下，有下面几种资源需要进行释放：

- 缓存 缓存包括一些文件缓存，图片缓存等，在用户正常使用的时候这些缓存很有作用，但当你的应用程序UI不可见的时候，这些缓存就可以被清除以减少内存的使用．比如第三方图片库的缓存．
- 一些动态生成动态添加的View．　这些动态生成和添加的View且少数情况下才使用到的View，这时候可以被释放，下次使用的时候再进行动态生成即可．比如原生桌面中，会在OnTrimMemory的TRIM_MEMORY_MODERATE等级中，释放所有AppsCustomizePagedView的资源，来保证在低内存的时候，桌面不会轻易被杀掉．

2.1 例子：释放不常用到的View．

代码出处：Launcher

Launcher.java:

AppsCustomizeTabHost.java:

AppsCustomizePagedView.java:

PagedViewGridLayout.java

2.2 例子: 清除缓存

代码出处：Contact

```
1
2           @Override
3           public void onTrimMemory(int level) {
4               if (level >= ComponentCallbacks2.TRIM_MEMORY_MODERATE) {
5                   // Clear the caches.  Note all pending requests will be removed too.
6                   clear();
7               }
8           }public void clear() {
9               mPendingRequests.clear();
10              mBitmapHolderCache.evictAll();
11              mBitmapCache.evictAll();
12          }
13
```

3. OnTrimMemory和onStop的关系？

onTrimMemory()方法中的TRIM_MEMORY_UI_HIDDEN回调只有当我们程序中的所有UI组件全部不可见的时候才会触发，这和onStop()方法还是有很大区别的，因为onStop()方法只是当一个Activity完全不可见的时候就会调用，比如说用户打开了我们程序中的另一个Activity。

因此，我们可以在onStop()方法中去释放一些Activity相关的资源，比如说取消网络连接或者注销广播接收器等，但是像UI相关的资源应该一直要等到onTrimMemory(TRIM_MEMORY_UI_HIDDEN)这个回调之后才去释放，这样可以保证如果用户只是从我们程序的一个Activity回到了另外一个Activity，界面相关的资源都不需要重新加载，从而提升响应速度。

需要注意的是，onTrimMemory的TRIM_MEMORY_UI_HIDDEN 等级是在onStop方法之前调用的.

4. OnTrimMemory和OnLowMemory的关系？

在引入OnTrimMemory之前都是使用OnLowMemory回调，需要知道的是，OnLowMemory大概和OnTrimMemory中的TRIM_MEMORY_COMPLETE级别相同，如果你想兼容api<14的机器，那么可以用OnLowMemory来实现，否则你可以忽略OnLowMemory，直接使用OnTrimMemory即可.

5. 为什么要调用OnTrimMemory?

尽管系统在内存不足的时候杀进程的顺序是按照LRU Cache中从低到高来的，但是它同时也会考虑杀掉那些占用内存较高的应用来让系统更快地获得更多的内存。

所以如果你的应用占用内存较小，就可以增加不被杀掉的几率，从而快速地恢复（如果不被杀掉，启动的时候就是热启动，否则就是冷启动，其速度差在2~3倍）。

所以说在几个不同的OnTrimMemory回调中释放自己的UI资源，可以有效地提高用户体验。

6. 有哪些典型的使用场景?

6.1 常驻内存的应用

一些常驻内存的应用，比如Launcher、安全中心、电话等，在用户使用过要退出的时候，需要调用OnTrimMemory来及时释放用户使用的时候所产生的多余的内存资源：比如动态生成的View、图片缓存、Fragment等。

6.2 有后台Service运行的应用

这些应用不是常驻内存的，意味着可以被任务管理器杀掉，但是在某些场景下用户不会去杀。
这类应用包括：音乐、下载等。用户退出UI界面后，音乐还在继续播放，下载程序还在运行。这时候音乐应该释放部分UI资源和Cache。