

个人资料



nei504293736



访问：560071次  
积分：8815  
等级：

BLDC

6

  
排名：第2285名  
  
原创：235 篇 转载：237 篇  
译文：3篇 评论：64条

文章搜索

文章分类

- android\_widgit (50)
- android\_net (45)
- android\_media (1)
- android\_data (38)
- android\_other (40)
- program\_knowledge (65)
- android\_draw (33)
- android\_class-or-method (27)
- android\_service (7)
- android\_BroadcastReceiver (3)
- android\_contentprovider (0)
- 工作中遇到的问题 (83)
- 易忘记 (9)
- android map (0)
- android 系统架构 (7)
- android工具 (12)
- linux (21)
- android framework (9)
- android system (3)
- MTK bug (2)

异步赠书：10月Python畅销书升级 【线路图】人工智能到底学什么？！ 程序员9月书讯 节后荐书：Python、PyQt5、Kotlin（评论送书）

# Android实战技巧：深入解析AsyncTask与thread的区别 THREAD\_POOL\_EXECUTOR

2013-07-27 00:10 5414人阅读 评论(1) 收藏 举报

分类： android\_class-or-method (26)

目录(?) [+]

此文转载声明：<http://blog.csdn.net/hitlion2008/article/details/>

## AsyncTask的介绍及基本使用方法

关于AsyncTask的介绍和基本使用方法可以参考[官方文档](#)和Android实战技巧：[AsyncTask](#)这里就不重复。

## AsyncTask引发的一个问题

上周遇到了一个极其诡异的问题，一个小功能从网络上下载一个图片，然后放在ImageView中，是用AsyncTask来实现的，本身逻辑也很简单，仅是在doInBackground中用HTTP请求把图片的输入流取出，然后用BitmapFactory去解析，然后再把得到的Bitmap放到ImageView中。这个应用是用4.0的SDK开发的，也是运行在4.0上面的。但是有时候下载这张图片去要用很久很久，甚至要等上几分钟。通过调试发现一个令人难以接受的事实：竟然是doInBackground()未及时执行，也就是它并没有在#execute()调用之后马上执行，而是等待了很久才得以执行。

神马情况，难道AsyncTask不是线程，难道不是异步，难道AsyncTask另有内幕？

## AsyncTask的内幕

AsyncTask主要有二个部分：一个是与主线各的交互，另一个就是线程的管理调度。虽然可能多个AsyncTask的子类的实例，但是AsyncTask的内部Handler和ThreadPoolExecutor都是进程范围内共享的，其都是static的，也即属于类的，类的属性的作用范围是CLASSPATH，因为一个进程一个VM，所以是AsyncTask控制着进程范围内所有的子类实例。

### 与主线程交互

与主线程交互是通过Handler来进行的，因为本文主要探讨AsyncTask在任务调度方面的，所以对于这部分不做细致介绍，感兴趣的朋友可以去看[AsyncTask的源码](#)

### 线程任务的调度

内部会创建一个进程作用域的线程池来管理要运行的任务，也就就是说当你调用了AsyncTask#execute()后，AsyncTask会把任务交给线程池，由线程池来管理创建Thread和运行Therad。对于内部的线程池不同版本的Android的实现方式是不一样的：

民间特效与民间插件 (3)

android-面试 (1)

git (1)

文章存档

2017年05月 (1)

2017年03月 (3)

2017年02月 (3)

2017年01月 (1)

2016年12月 (1)

展开

阅读排行

Java子类与父类的初 (29958)

android Intent.FLAG\_ (13347)

[Android]中国大部分 (10879)

android应用程序跳转 (9293)

android 出现java.lang (7903)

android adb shell:unk (6957)

android 获得一个应用 (6205)

android使用自定义属 (5746)

Android面试准备复习 (5503)

Android实战技巧：深 (5414)

评论排行

android 出现java.lang (4)

android 获得一个应用 (4)

android adb shell:unk (4)

Android 保存图片到S (4)

Android面试准备复习 (3)

如今找一份andorid工 (3)

android列出手机SDc (2)

[Android]中国大部分 (2)

【Android实例】通过 (2)

android应用程序跳转 (2)

推荐文章

\* 【观点】第二十三期：程序员应该如何积累财富？

\* Android检查更新下载安装

\* 动手打造史上最简单的Recycleview 侧滑菜单

\* TCP网络通讯如何解决分包粘包问题

\* SDCC 2017之大数据技术实战线上峰会

\* 快速集成一个视频直播功能

Android2.3以前的版本，也即SDK/API 10和以前的版本

内部的线程池限制是5个，也就是说同时只能有5个线程运行，超过的线程只能等待，等待前面的线程某个执行完了才被调度和运行。换句话说，如果一个进程中的AsyncTask实例个数超过5个，那么假如前5个都运行很长时间的的话，那么第6个只能等待机会了。这是AsyncTask的一个限制，而且对于2.3以前的版本无法解决。如果你的应用需要大量的后台线程去执行任务，那么你能只能放弃使用AsyncTask，自己创建线程池来管理Thread，或者干脆不用线程池直接使用Thread也无妨。不得不说，虽然AsyncTask较Thread使用起来比较方便，但是它最多只能同时运行5个线程，这也大大局限了它的实力，你必须要小心的设计你的应用，错开使用AsyncTask的时间，尽力做到分时，或者保证数量不会大于5个，否则就可能遇到上面提到的问题。要不然就只能使用JavaSE中的API了。



Android 3.0以后，也即SDK/API 11和以后的版本

可能是Google意识到了AsyncTask的局限性了，从Android 3.0开始对AsyncTask的API做出了一些调整：

- 1. #execute()提交的任务，按先后顺序每次只运行一个

也就是说它是按提交的次序，每次只启动一个线程执行一个任务，完成之后再执行第二个任务，也就是相当于只有一个后台线程在执行所提交的任务 (Executors.newSingleThreadPool())。

关闭

## 最新评论

Android 保存图片到SQLite  
chy0551: 求源码

Android之monkey Test,Monkey测试  
zhanglu-cs: Monkey测试时,怎么设置手机不下拉状态栏 看不明白啊

手机低至或是高至一定温度  
vae1573: 这个路径终于对了, lz有实验过吗? 温度过高有关机提示吗?

Android 保存图片到SQLite  
crazycat喵: 有点乱, 有demo吗, 亲

Android 保存图片到SQLite  
crazycat喵: 对, 很简单的道理你存了路径, 假如你删了那那张图片, 以后你去拿就是空了, 很多人没考虑这种结果吧

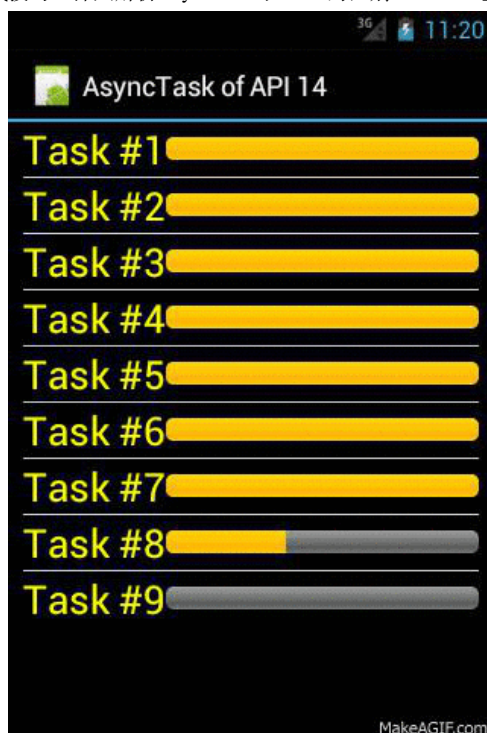
android UI进阶之弹窗的使用  
和小小\_love: 说好的效果呢。。

Android 保存图片到SQLite  
crazycat喵: 详细用完回来再评论

Android Text控件之属性: aizhengpinchopin: 谢谢

android 获得一个应用程序的天煞魔猎手: ApplicationInfo根本就没有intent属性, 为什么那么喜欢抄代码, 这种东西如果是自己...

android 获得一个应用程序的sj940826:  
@Huanglong1987: 同上。我也是ComponentName  
aName = a.inten...



## 2. 新增了接口#executeOnExecutor()

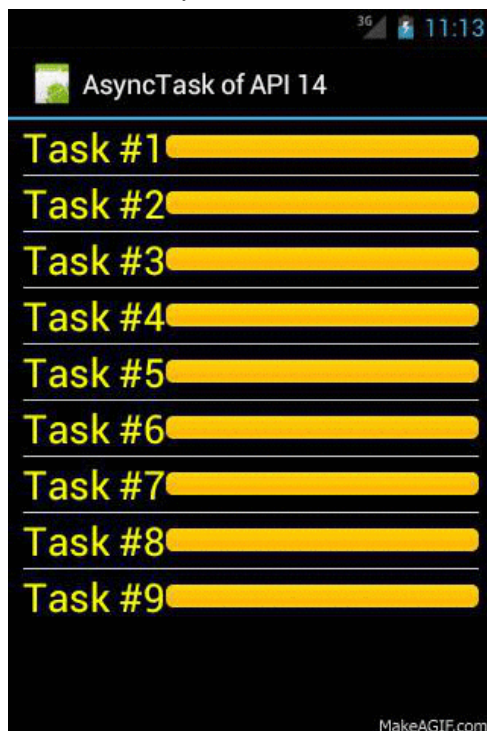
这个接口允许开发者提供自定义的线程池来运行和调度Thread, 如果你想让所有的任务都能并发同时运行, 那就创建一个没有限制的线程池

(Executors.newCachedThreadPool()), 并提供给AsyncTask。这样这个例子就有了自己的线程池而不必使用AsyncTask默认的。

## 3. 新增了二个预定义的线程池SERIAL\_EXECUTOR和THREAD\_POOL\_EXECUTOR

其实THREAD\_POOL\_EXECUTOR并不是新增的, 之前的就有, 只不过之前(Android 2.3)它是AsyncTask私有的, 未公开而已。THREAD\_POOL\_EXECUTOR是一个corePoolSize为5的线程池, 也就是说最多只有5个线程同时运行, 超过5个的就要等待。所以如果使用executeOnExecutor(AsyncTask.THREAD\_POOL\_EXECUTOR)就跟2.3版本的AsyncTask.execute()效果是一样的。

关闭



而`SERIAL_EXECUTOR`是新增的，它的作用是保证任务执行的顺序，也保证提交的任务确实是按照先后顺序执行的。它的内部有一个队列用来存任务，保证当前只运行一个，这样就可以保证任务是完全按照顺序执行的。`execute()`使用的就是这个，也就是`executeOnExecutor(AsyncTask.SERIAL_EXECUTOR)`与`execute()`是一个东西。

## 前面问题的解法

了解了AsyncTask的内幕就知道了前面问题的原因：因为是4.0平台，所以所有的AsyncTask并不都会运行在单独的线程中，而是被`SERIAL_EXECUTOR`顺序的使用线程执行。因为应用中可能还有其他地方使用AsyncTask，所以到网络取图片的AsyncTask也许会等待到其他任务都完成时才得以执行而不是调用`executor()`之后马上执行。

那么解决方法其实很简单，要么直接使用Thread，要么创建一个单独的线程池(`Executors.newCachedThreadPool()`)。或者最简单的解法就是使用`executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR)`，这样起码不用等到前面的都结束了再执行。

## AsyncTask的使用注意事项

前面的文章曾建议使用AsyncTask而不是使用Thread，但是AsyncTask似乎又有它的限制，这就要根据具体的需求情况而选择合适的工具，**No Silver Bullet**。下面是一些建议：

### 改善你的设计，少用异步处理

线程的开销是非常大的，同时异步处理也容易出错，难调试，难维护，所以改善你的设计，尽可能的少用异步。对于一般性的数据库查询，少量的I/O操作是没有必要启动线程的。

与主线程有交互时用AsyncTask，否则就用Thread

AsyncTask被设计出来的目的就是为了满足Android的特殊需求：非主线程不能操作(UI)组件，所以AsyncTask扩展Thread增强了与主线程的交互的能力。如果你的应用没有与主线程交互，那么就直接使用Thread就好了。

[关闭](#)

## 当有需要大量线程执行任务时，一定要创建线程池

线程的开销是非常大的，特别是创建一个新线程，否则就不必设计线程池之类的工具了。当需要大量线程执行任务时，一定要创建线程池，无论是使用AsyncTask还是Thread，因为使用AsyncTask它内部的线程池有数量限制，可能无法满足需求；使用Thread更是要线程池来管理，避免虚拟机创建大量的线程。比如从网络上批量下载图片，你不想一个一个的下，或者5个5个的下载，那么就创建一个CorePoolSize为10或者20的线程池，每次10个或者20个这样的下载，即满足了速度，又不至于耗费无用的性能开销去无限制的创建线程。

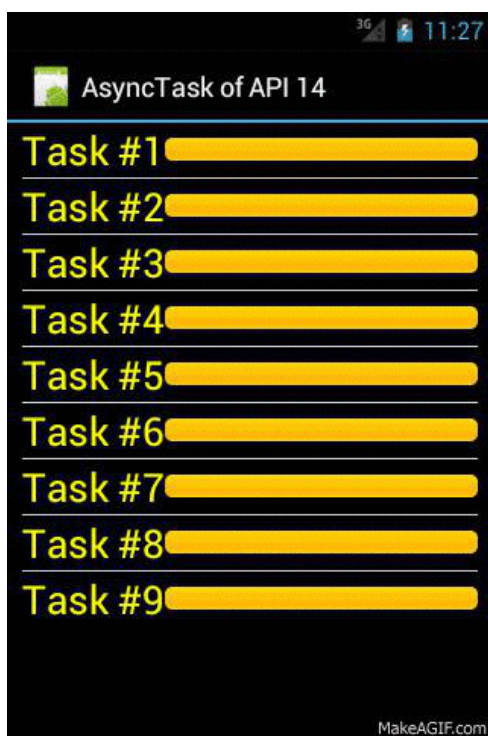
对于想要立即开始执行的异步任务，要么直接使用Thread，要么单独创建线程池提供给AsyncTask

默认的AsyncTask不一定会立即执行你的任务，除非你提供给他一个单独的线程池。如果不与主线程交互，直接创建一个Thread就可以了，虽然创建线程开销比较大，但如果这不是批量操作就没有问题。

Android的开发没有想像中那样简单，要多花心思和时间在代码上和测试上面，以确信程序是优质的

附上相关资源：

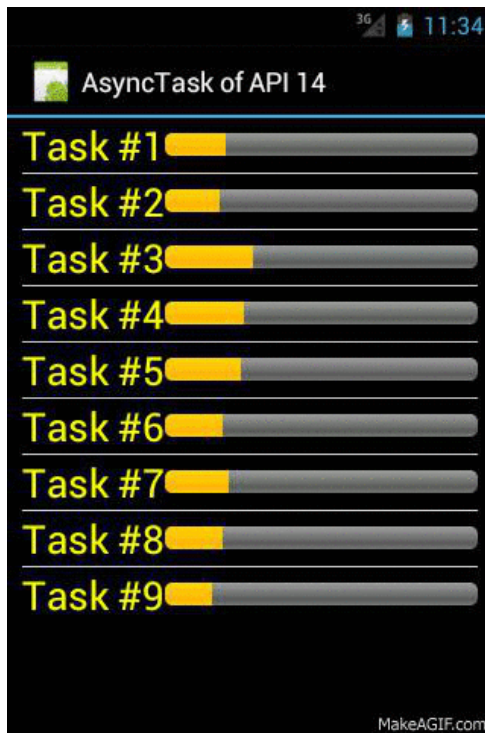
使用自定义的CorePoolSize为7的Executor(Executors.newFixedThreadPool(



使用未设限制的Executor(Executors.newCachedThreadPool()):

关闭





这些例子所用的代码：



[html] view plain copy print ?

```
01. <?xml version="1.0" encoding="utf-8"?>
02. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
03.     android:layout_width="match_parent"
04.     android:layout_height="match_parent"
05.     android:paddingLeft="10dip"
06.     android:paddingRight="10dip"
07.     android:orientation="vertical" >
08.     <ListView android:id="@+id/task_list"
09.         android:layout_width="fill_parent"
10.         android:layout_height="wrap_content"
11.         android:divider="#cccccc"
12.         android:dividerHeight="0.6dip"
13.         android:footerDividersEnabled="true"
14.         android:headerDividersEnabled="true" />
15. </LinearLayout>
```

[html] view plain copy print ?

```
01. <?xml version="1.0" encoding="utf-8"?>
02. <com.hilton.effectiveandroid.os.TaskItem xmlns:anc
03.     android:layout_width="match_parent"
04.     android:layout_height="50dip"
05.     android:gravity="center_vertical"
06.     android:layout_gravity="center_vertical"
07.     android:orientation="horizontal" >
08.     <TextView android:id="@+id/task_name"
09.         android:layout_width="wrap_content"
10.         android:layout_height="wrap_content"
11.         android:textColor="#ffff00"
12.         android:textSize="26sp" />
13.     <ProgressBar android:id="@+id/task_progress"
14.         android:layout_width="fill_parent"
15.         android:layout_height="15dip"
16.         android:max="100"
```

关闭



```
17.         style="@android:style/Widget.ProgressBar.Horizontal" />
18.     </com.hilton.effectiveandroid.os.TaskItem >
```

顶

1

踩

0

上一篇

andorid4.2 mtk无论在什么情况下按耳机按钮都会调用音乐播放器

下一篇

adb server is out of date. killing...


相关文章推荐

- 理解Thread Pool, Executor, Callable/Future
  - Python全栈工程师入门指南
  - 深度理解Thread Pool, Executor, Callable/...
  - 自然语言处理在“天猫精灵”的实践应用--姜...
  - Android中的Thread与AsyncTask的区别
  - Vue2.x基本特性解析
  - Android中的Thread与AsyncTask的区别
  - 程序员都应该掌握的Git和Github实用教程
- Android中的Thread与AsyncTask的区别?
  - 深度学习项目实战-人脸检测
  - Android中的Thread与AsyncTask的
  - Shell脚本编程
  - android Handler Thread AsyncTask
  - 两种线程池：THREAD\_POOL\_EXECUTOR
  - Android实战技巧：深入解析AsyncTask(转)
  - Android实战技巧：深入解析AsyncTask



查看评论

1楼 lhy6962 2013-10-22 14:00发表



题外话，有个开源的图片加载工具：ImageLoader  
挺好用

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场