

# Android 获取前台应用



GordenNee (/u/fb5ee6c7113e) [+ 关注](#)

2017.09.12 18:06\* 字数 1805 阅读 112 评论 3 喜欢 4

(/u/fb5ee6c7113e)

## 一.背景：

可以获取到Android设备当前正在显示的前台应用（如果可以，精细到页面）。

## 二.风险点

- 兼容Android 各大版本
- 兼容所有应用

## 三.调研方案

### 3.1 Android 5.0之前getRunningTasks

Android5.0以前，使用ActivityManager的getRunningTasks()方法，可以得到应用包名和Activity；

```
ActivityManager activityManager = (ActivityManager)context.getApplicationContext().getSystemService(Context.ACTIVITY_SERVICE);
ComponentName runningTopActivity = activityManager.getRunningTasks(1).get(0).topActivity;
```

还需要声明权限：

```
<uses-permission android:name="android.permission.GET_TASKS" />
```



这种方法不止能获取包名，还能获取Activity名。但是在Android 5.0以后，系统就不再对第三方应用提供这种方式来获取前台应用了，虽然调用这个方法还是能够返回结果，但是结果只包含你自己的Activity和Launcher了。

具体可见下面的权限判断：

```
private boolean isGetTasksAllowed(String caller, int callingPid, int callingUid) {
    boolean allowed = checkPermission(android.Manifest.permission.REAL_GET_TASKS,
        callingPid, callingUid) == PackageManager.PERMISSION_GRANTED;
    if (!allowed) {
        if (checkPermission(android.Manifest.permission.GET_TASKS,
            callingPid, callingUid) == PackageManager.PERMISSION_GRANTED) {
            // Temporary compatibility: some existing apps on the system image may
            // still be requesting the old permission and not switched to the new
            // one; if so, we'll still allow them full access. This means we need
            // to see if they are holding the old permission and are a system app.
            try {
                if (AppGlobals.getPackageManager().isUidPrivileged(callingUid)) {
                    allowed = true;
                    Slog.w(TAG, caller + ": caller " + callingUid
                        + " is using old GET_TASKS but privileged; allowing");
                }
            } catch (RemoteException e) {
            }
        }
    }
    if (!allowed) {
        Slog.w(TAG, caller + ": caller " + callingUid
            + " does not hold REAL_GET_TASKS; limiting output");
    }
    return allowed;
}
```

### 3.2 通过使用量统计功能获取前台应用

在StackOverFlow上大多数的答案都是使用usage statistics API。

Android提供了usage statistics API。这个API本来是系统用来统计app使用情况的，包含了每个app最近一次被使用的时间。我们只需要找出距离现在时间最短的那个app，就是当前在前台的app。



```
private String getForegroundApp(Context context) {
    UsageStatsManager usageStatsManager =
        (UsageStatsManager) context.getSystemService(Context.USAGE_STATS_SERVICE);
    long ts = System.currentTimeMillis();
    List<UsageStats> queryUsageStats =
        usageStatsManager.queryUsageStats(UsageStatsManager.INTERVAL_BEST, 0, ts);
    UsageEvents usageEvents = usageStatsManager.queryEvents(isInit ? 0 : ts-5000, ts);
    if (usageEvents == null) {
        return null;
    }

    UsageEvents.Event event = new UsageEvents.Event();
    UsageEvents.Event lastEvent = null;
    while (usageEvents.getNextEvent(event)) {
        // if from notification bar, class name will be null
        if (event.getPackageName() == null || event.getClassName() == null) {
            continue;
        }

        if (lastEvent == null || lastEvent.getTimestamp() < event.getTimestamp())
            lastEvent = event;
    }

    if (lastEvent == null) {
        return null;
    }
    return lastEvent.getPackageName();
}
```

### 问题点：

- 这种方式只能拿到包名，无法精确到了Activity了。
- 使用这种方发之前，首先要引导用户开启使用量功能：

```
Intent intent = new Intent(Settings.ACTION_USAGE_ACCESS_SETTINGS);
startActivity(intent);
```

- 还要申明权限：



```
<uses-permission android:name="android.permission.PACKAGE_USAGE_STATS" />
```

这个权限试了下Android Studio 直接提示为系统权限，普通App无法申请。

- 另外！因为在一些手机上，应用发起通知栏消息的时候，或者是下拉通知栏，也会被记录到使用量中，就会导致按最近时间排序出现混乱。而且收起通知栏以后，这种混乱并不会被修正，而是必须重新开启一个应用才行。

到这里基本可以先否定这个方案了，步骤复杂，还需要用户手动开启权限，不可能啦！

### 3.3 通过辅助服务获取前台应用

Android 辅助服务(AccessibilityService)有很多神奇的妙用，比如辅助点击，比如页面抓取，还有就是获取前台应用。

这里简单介绍一下如何使用辅助服务，首先要在AndroidManifest.xml中声明：

```
<service
    android:name=".service.AccessibilityMonitorService"
    android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"
    >
    <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService" />
    </intent-filter>

    <meta-data
        android:name="android.accessibilityservice"
        android:resource="@xml/accessibility" />
</service>
```

然后在res/xml/文件夹下新建文件accessibility.xml，内容如下：



```
<?xml version="1.0" encoding="utf-8"?>
<accessibility-service
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:accessibilityEventTypes="typeViewClicked|typeViewLongClicked|typeWindowS
    android:accessibilityFeedbackType="feedbackGeneric"
    android:accessibilityFlags="flagRetrieveInteractiveWindows"
    android:canRetrieveWindowContent="true"
    android:canRequestFilterKeyEvents="true"
    android:notificationTimeout="10"
    android:packageName="@null"
    android:description="@string/accessibility_des"
    android:settingsActivity="com.pl.recent.MainActivity"
/>
```

关键是typeWindowStateChaged。

新建AccessibilityMonitorService，主要内容如下：

```
public class AccessibilityMonitorService extends AccessibilityService {
    private CharSequence mWindowClassName;
    private String mCurrentPackage;
    @Override
    public void onAccessibilityEvent(AccessibilityEvent event) {
        int type=event.getEventType();
        switch (type){
            case AccessibilityEvent.TYPE_WINDOW_STATE_CHANGED:
                mWindowClassName = event.getClassName();
                mCurrentPackage = event.getPackageName()==null?"":event.getPackageNa
                break;
            case TYPE_VIEW_CLICKED:
            case TYPE_VIEW_LONG_CLICKED:
                break;
        }
    }
}
```

问题点

- 也是需要用户去在“设置里”找到辅助服务并开启即可。



- 辅助服务在一些手机（小米、魅族、华为等国产手机）上，一旦程序被清理后台，就会被关闭。。。

so, 这种方案不太稳定而且也是需要用户手动去开启, 不可能啦!

### 3.4 通过设备辅助应用程序获取前台应用（比较鸡肋）

所谓设备辅助应用程序，是在一些接近原生的系统上，长按Home键就会触发的应用，默认是会触发Google搜索。设备辅助应用程序有点像是需要主动触发的辅助服务，因为应用中是无法主动去触发其功能的，所以说比较鸡肋，

### 3.5 PS 命令

在Android 的ADB命令中我们可以通过 ps 命令来获取到一些应用进程信息，看下官方解释：

P show scheduling policy, either bg or fg are common, but also un and er for failure

大概意思就是说这个会列出系统调度列表，如果是系统的话，那是不是就说明能够得到界面的调度呢？

## Android shell tricks: ps

If you ever played around with the `adb shell` you may have found that the `ps` utility, which lists process lists, is not as verbose as you would expect it to be. And, to make things worse, there's no inline help or `man` entries. Here's the `ps` utility usage line: `ps -t -x -P -p -c [pid|name]`.

[illegible]

## Android shell tricks: ps



- -t show threads, comes up with threads in the list
- -x shows time, user time and system time in seconds
- **-P show scheduling policy, either bg or fg are common, but also un and er for failures to get policy**
- **-p show priorities, niceness level**
- -c show CPU (may not be available prior to Android 4.x) involved
- [pid] filter by PID if numeric, or...
- [name] ...filter by process name

Android's core toolbox (shell utilities) are more primitive than the ones you may be used to. Notice how each argument needs to be separated and you can't just `-txPc` it all, the command line argument parser is non-complex.

It's a pity how command line arguments are not shown. If you need something that's not available by the stock `ps` shell utility, try manually combing through (<https://link.jianshu.com?t=http://archive09.linux.com/feature/126718>) the `/proc` directory. For the command line one would do `cat /proc/<pid>/cmdline`.

首先我们在cmd命令行模式输入：`adb shell ps` 输出一下信息：

```

ud_878 5508 209 045028 20696 ffffffff 00000000 S con.qihoo968.mobilesafe.c
lsan
shell 5575 1 6680 916 ffffffff 00000000 R /sbin/caddb
amnt 5768 2 0 0 ffffffff 00000000 S kunekun/0:33
proc 5762 2 0 0 ffffffff 00000000 S kunekun/0:35
alluq 8278 1 1280 768 ffffffff 00000000 S /system/bin/alluq
proc 9631 2 0 0 ffffffff 00000000 S kunekun/0:2
adlin 9772 269 831026 16640 ffffffff 00000000 S com.qihoo360.qqwsliteglam
e1
ad_82 9818 269 848356 32184 ffffffff 00000000 S com.qihoo360.qqwsliteglam
tionsservice
bank 9901 2 0 0 ffffffff 00000000 S kunekun/0:1
proc 9939 2 0 0 ffffffff 00000000 S kunekun/0:3
system 9900 209 010052 10556 ffffffff 00000000 S com.redbend.vdnc
ud_876 16012 269 848248 27556 ffffffff 00000000 S com.baidu.video-pushservi

```

```

root 10047 2 0 0 ffffffff 00000000 S flush 179:32
uid_000 10051 269 858276 21418 ffffffff 00000000 R com.baidu.MapLocationService
eService 10054 269 851736 32132 ffffffff 00000000 R com.jingdong.app.mall.jdp
ush cloud 10027 2 0 0 ffffffff 00000000 S kunken/0:4
root 10029 2 0 0 ffffffff 00000000 S kunken/0:5
uid_028 10039 269 831622 19700 ffffffff 00000000 S com.google.android.apps.ad
planner 10059 269 832148 19360 ffffffff 00000000 S android.permission.medi
uid_039 10316 269 845736 20501 ffffffff 00000000 E com.qq.tft.idmoh=reactorui
uid_040 10360 269 839840 28332 ffffffff 00000000 E android.process.acore
uid_042 10027 269 830576 10900 ffffffff 00000000 S com.nubia.unashible
uid_072 10014 269 850121 32900 ffffffff 00000000 E com.jingdong.app.mall.jdp
ush
shell 10554 5575 1308 700 c0011020 1023f994 E /system/bin/sh
shell 10556 10554 3076 2008 c0147aac 4001112c S logcat
uid_078 10558 1 809220 5992 ffffffff 00000000 E com.qihoo.appstore_coreba
mon
root 11100 2 0 0 ffffffff 00000000 R nigelini/1
root 11101 2 0 0 ffffffff 00000000 S kunken/1:0
root 11102 2 0 0 ffffffff 00000000 R kunf1img1/1
root 11103 2 0 0 ffffffff 00000000 S watchdog/1
root 11104 2 0 0 ffffffff 00000000 R kunken/1:1
root 11107 2 0 0 ffffffff 00000000 S kunken/1:2
root 11162 1 12848 2192 ffffffff 00000000 R rmlsmon
root 11179 1 7180 402 ffffffff 00000000 S 360sguard
root 11180 1 7180 402 ffffffff 00000000 S 360sguard
uid_079 11036 10058 009220 5900 ffffffff 00000000 S com.qihoo.appstore_CoreBa
mon
shell 11673 5575 1360 700 c0011020 1023f994 E /system/bin/sh
root 11662 2 0 0 ffffffff 00000000 R nigelini/2
root 11663 2 0 0 ffffffff 00000000 E kunther/2:0
root 11664 2 0 0 ffffffff 00000000 S kunf1img1/2
root 11665 2 0 0 ffffffff 00000000 E watchdog/2
root 11672 2 0 0 ffffffff 00000000 S kunken/2:1
root 11673 2 0 0 ffffffff 00000000 E kunther/2:2
shell 11627 5575 1340 700 c0011020 1017b994 S /system/bin/sh
shell 11699 11697 3612 800 c0011020 1023b000 E dumpsys
shell 11700 11643 1444 672 00000000 402ac090 R ps
uid_072 12333 269 953420 105564 ffffffff 00000000 R com.jingdong.app.mall
root 12759 2 0 0 ffffffff 00000000 S kunken/0:0
uid_076 20779 269 830692 22501 ffffffff 00000000 R com.baidu.aishen

```

然后能很清晰的看见各种包名而且都是系统正在运行中的,按照说明如果 `** -p` 参数可以列出前台进程调度的话,如果我们在切换程序或者对出时包名列表都会有变化。

以下是输入 `adb shell ps -p` 后输出的信息：

```

uid_079 5908 269 045020 20696 ffffffff 00000000 S com.qihoo360.mobilesafe.c
mon

```



```

shell 5575 1 6896 316 ffffffff 00000000 R /system/bin/sh
proc 5704 2 0 0 ffffffff 00000000 S kunken/0:11
proc 5762 2 0 0 ffffffff 00000000 S kunken/0:15
dheap 8278 1 1280 768 ffffffff 00000000 E /system/bin/dheapd
proc 9631 2 0 0 ffffffff 00000000 S kunken/0:2
radio 9777 269 831076 16600 ffffffff 00000000 E con.qualcomm.gnsslingstun
al
uid_877 9813 269 848356 27184 ffffffff 00000000 E con.sutanavi.ndinap: loca
tionService
proc 9903 2 0 0 ffffffff 00000000 E kunken/0:1
proc 9909 2 0 0 ffffffff 00000000 S kunken/0:1
bugdama 9983 269 831052 18556 ffffffff 00000000 R com.malland.educ
uid_870 10012 307 848348 27556 ffffffff 00000000 S con.baidu.videotpushservi
proc 10047 2 0 0 ffffffff 00000000 S flush_172:32
uid_888 10051 269 856076 24448 ffffffff 00000000 R com.baidu.BaiduMapMapGoe
Service
uid_772 10124 269 861736 32132 ffffffff 00000000 R com.jingdong.app.mall:jdpu
ushcloud
proc 10227 2 0 0 ffffffff 00000000 R kunken/0:4
proc 10227 2 0 0 ffffffff 00000000 S kunken/0:5
uid_828 10239 269 831622 19700 ffffffff 00000000 S com.google.android.apps.ad
plunder
uid_49 10259 269 832148 19360 ffffffff 00000000 S android.permission.medi
uid_889 10316 269 845736 24504 ffffffff 00000000 E con.ggtf.iaaob:reactServ
ice
uid_88 10360 269 839840 28332 ffffffff 00000000 E android.process.acore
uid_42 10397 269 830576 16988 ffffffff 00000000 R com.mallia.malllib
uid_772 10414 269 860424 32980 ffffffff 00000000 E con.jingdong.app.mall:jdpu
ush
shell 10554 5575 1308 700 c0011020 4020f994 E /system/bin/sh
shell 10556 10554 3076 2008 c014700c 4001112c S logcat
uid_78 10558 1 809220 5992 ffffffff 00000000 E con.qihoo.appstore_coreBa
nnon
proc 11100 2 0 0 ffffffff 00000000 R nigelLim/1
proc 11101 2 0 0 ffffffff 00000000 S kunken/1:0
proc 11102 2 0 0 ffffffff 00000000 R kunfLim/1
proc 11103 2 0 0 ffffffff 00000000 S watchdog/1
proc 11104 2 0 0 ffffffff 00000000 R kunken/1:1
proc 11107 2 0 0 ffffffff 00000000 S kunken/1:2
proc 11162 1 12848 2492 ffffffff 00000000 R malacoma
proc 11179 1 7180 402 ffffffff 00000000 S 360sguard
proc 11180 1 7180 402 ffffffff 00000000 S 360sguard
uid_870 11606 10058 809220 5998 ffffffff 00000000 S con.qihoo.appstore_CoreBa
nnon
shell 11649 5575 1308 700 c0011020 4020f994 S /system/bin/sh
proc 11662 2 0 0 ffffffff 00000000 S nigelLim/2
proc 11663 2 0 0 ffffffff 00000000 E kunken/2:0
proc 11664 2 0 0 ffffffff 00000000 R kunfLim/2
proc 11665 2 0 0 ffffffff 00000000 E watchdog/2
proc 11677 2 0 0 ffffffff 00000000 S kunken/2:1
proc 11687 2 0 0 ffffffff 00000000 E kunken/2:2
shell 11697 5575 1340 700 c0011020 4017b994 S /system/bin/sh
shell 11699 11697 3612 880 c0011020 4020f994 E dumpsys
shell 11700 11643 1444 672 00000000 4020f994 R ps
uid_772 12333 269 953428 105564 ffffffff 00000000 S com.jingdong.app.mall
proc 12559 2 0 0 ffffffff 00000000 S kunken/0:0

```



```
u0 a7c 28779 209 018692 22564 ffffff00 00000000 8 com.baidu.vidon
```

仔细观察会发现u0开头的都是我们正常程序的包名,而且在程序切换到后台以后,这个列表是有变化的,随便启动一个自己安装的应用,列表也刚好出现那个应用。在此大家应该就已经知道怎么写了,这里也提供一下实现思路:

- 1、命令行获取控制台输出流
- 2、找出每行输出的 u0开头的信息获取包名
- 3、用一个列表存入,与每次获取的当前列表项与上一次列表项对比,如果旧的列表不存在此包名,那就证明这个包就是新启动的了,如果没有就不做任何操作。

测试结论:

- 理论上没什么毛病,但是实践中发现存在不稳定的现象,有时候根本拿不到,有时候获取失败
- 比如:最常用的微信,打开微信到登录页面时并没有捕捉到。
- 因此,该方案存在一些不稳定因素

### 3.5 大招

从网络上看到一篇老外大神的做法,中文分析博客已经丢失,google一下也没有啥有效因袭。所以只好自己大概猜测和理解了

上代码:



```
public class SuperRunningPackage {

    /** first app user */
    public static final int AID_APP = 100000;
    /** offset for uid ranges for each user */
    public static final int AID_USER = 1000000;
    public static String getForegroundApp() {
        Log.e("PKG", "VersionCode:" + Build.VERSION.SDK_INT);
        File[] files = new File("/proc").listFiles();
        int lowestOomScore = Integer.MAX_VALUE;
        String foregroundProcess = null;
        for (File file : files) {
            if (!file.isDirectory()) {
                continue;
            }
            int pid;
            try {
                pid = Integer.parseInt(file.getName());
            } catch (NumberFormatException e) {
                continue;
            }
            try {
                String cgroup = read(String.format("/proc/%d/cgroup", pid));
                String[] lines = cgroup.split("\n");
                String cpuSubsystem;
                String cpuacctSubsystem;

                for (int i = 0; i < lines.length; i++) {

                    Log.e("PKG", lines[i]);

                }

                if (lines.length == 2) { //有的手机里cgroup包含2行或者3行，我们取cpu和cpuacct
                    cpuSubsystem = lines[0];
                    cpuacctSubsystem = lines[1];
                } else if (lines.length == 3) {
                    cpuSubsystem = lines[0];
                    cpuacctSubsystem = lines[2];
                } else if (lines.length == 5) { //6.0系统
                    cpuSubsystem = lines[2];
                    cpuacctSubsystem = lines[4];
                } else {
                    continue;
                }
            }
            if (!cpuacctSubsystem.endsWith(Integer.toString(pid))) {
```



```

        // not an application process
        continue;
    }
    if (cpuSubsystem.endsWith("bg_non_interactive")) {
        // background policy
        continue;
    }
    String cmdline = read(String.format("/proc/%d/cmdline", pid));
    if (cmdline.contains("com.android.systemui")) {
        continue;
    }
    int uid = Integer.parseInt(
        cpuacctSubsystem.split(":")[2].split("/")[1].replace("uid_", ""));
    if (uid >= 1000 && uid <= 1038) {
        // system process
        continue;
    }
    int appId = uid - AID_APP;
    int userId = 0;
    // loop until we get the correct user id.
    // 1000000 is the offset for each user.
    while (appId > AID_USER) {
        appId -= AID_USER;
        userId++;
    }
    if (appId < 0) {
        continue;
    }
    // u{user_id}_a{app_id} is used on API 17+ for multiple user account
    // String uidName = String.format("u%d_a%d", userId, appId);
    File oomScoreAdj = new File(String.format("/proc/%d/oom_score_adj",
        pid));
    if (oomScoreAdj.canRead()) {
        int oomAdj = Integer.parseInt(read(oomScoreAdj.getAbsolutePath()));
        if (oomAdj != 0) {
            continue;
        }
    }
    int oomscore = Integer.parseInt(read(String.format("/proc/%d/oom_score",
        pid)));
    if (oomscore < lowestOomScore) {
        lowestOomScore = oomscore;
        foregroundProcess = cmdline;
    }
} catch (IOException e) {
    e.printStackTrace();
}
}
return foregroundProcess;
}

```



```
private static String read(String path) throws IOException {
    StringBuilder output = new StringBuilder();
    BufferedReader reader = new BufferedReader(new FileReader(path));
    output.append(reader.readLine());
    for (String line = reader.readLine(); line != null; line = reader.readLine())
        output.append('\n').append(line);
    }
    reader.close();
    return output.toString().trim();
}

}
```

不详细分析，大概意思就是每次前台应用变动就会改变一个配置文件，因此可以通过读取改配置的方案来获取前台应用。经过测试，基本主流应用在前台时都可以捕捉到。

## 四.大结论

- Android 5.0 以下可以通过 `getRunningTasks` 获取到前台的包名。
- Android 5.0-6.0 可以通过读取系统配置文件来获取当前前台应用。
- Android 7.0+暂时不确定稳定性，需要后期更多实践，我理解7.0 以下已经基本满足需求。

小礼物走一走，来简书关注我

赞赏支持





GordenNee (/u/fb5ee6c7113e)

写了 26201 字, 被 18 人关注, 获得了 69 个喜欢  
(/u/fb5ee6c7113e)

+ 关注

喜欢 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-like-button) | 4



更多分享

(http://cwb.assets.jianshu.io/notes/images/16997006/weibo/image\_



(/apps/download?utm\_source=nbc)



登录 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-comment-form)

3条评论

只看作者

按喜欢排序 按时间正序 按时间倒序



自由翱翔de锤锤 (/u/77aabd2807a9)

2楼 · 2017.09.13 15:19

(/u/77aabd2807a9)  
版主能否分享一下你的全部代码

👍 赞    💬 回复


GordenNee (/u/fb5ee6c7113e) : 上面已经把主要代码都写出来了, 全部代码我后面整理整理 😊

2017.09.14 11:27    💬 回复

自由翱翔de锤锤 (/u/77aabd2807a9) : @GordenNee (/users/fb5ee6c7113e) 谢谢分享

2017.09.15 00:53    💬 回复



 添加新评论

被以下专题收入，发现更多相似内容



Android... (/c/5139d555c94d?utm\_source=desktop&utm\_medium=notes-included-collection)



Android开发 (/c/d1591c322c89?utm\_source=desktop&utm\_medium=notes-included-collection)



程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)



手机移动程序开发 (/c/52b64f3155e1?utm\_source=desktop&utm\_medium=notes-included-collection)



Android知识 (/c/3fde3b545a35?utm\_source=desktop&utm\_medium=notes-included-collection)

## Android - 收藏集 (/p/dad51f6c9c4d?utm\_campaign=maleskine&utm\_c...

用两张图告诉你，为什么你的 App 会卡顿？ - Android - 掘金 Cover 有什么料？从这篇文章中你能获得这些料：知道setContentView()之后发生了什么？ ... Android 获取 View 宽高的常用正确方式，避免为零 - 掘金...



passiontim (/u/e946d18f163c?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

## Android - 收藏集 - 掘金 (/p/5ad013eb5364?utm\_campaign=maleskine&...

用两张图告诉你，为什么你的 App 会卡顿？ - Android - 掘金 Cover 有什么料？从这篇文章中你能获得这些料：知道setContentView()之后发生了什么？ ... Android 获取 View 宽高的常用正确方式，避免为零 - 掘金...



掘金官方 (/u/5fc9b6410f4f?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)



## awesome-android (/p/325d0794fcd0?utm\_campaign=maleskine&utm\_...

作者：snowdream 微信：sn0wdr1am 原文地址：https://github.com/snowdream/awesome-android

awesome-android Introduction android libs from github System re...



snowdream (/u/748f0f7e6432?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

## awesome-android (/p/1a792b14af3b?utm\_campaign=maleskine&utm\_...

afinalAfinal是一个android的ioc，orm框架 https://github.com/yangfuhai/afinal xUtils\*\*android orm, bitmap,

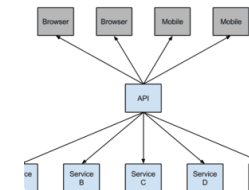
http, view inject... https://github.com...



passiontim (/u/e946d18f163c?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/46fd0faecac1?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

## Spring Cloud (/p/46fd0faecac1?utm\_campaign=maleskine&utm\_conte...

Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线）。分布式系统的协调导致了样板模式，使用Spring Cloud开发人员可...



卡卡罗2017 (/u/d90908cb0d85?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

## Day 396 口才——口才篇 (/p/6c6221c70ed0?utm\_campaign=maleskine&...

（一）本文梗概 这篇小记主要是澄清有关“口才”这方面的概念。（二）主要内容 what：什么是“口才”与“说话” 说话：说话的本质在于大脑中的一些想法，通过语言形式所进行的一个外化 口才：当我们脱稿并对着...



黑七点 (/u/488e1cbfa6c5?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)





(/p/9c2f2dce44b9?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

**zzang!!!我要开始写日记啦~ (/p/9c2f2dce44b9?utm\_campaign=maleskin...**

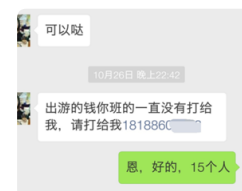
最近好喜欢听粤语歌啊，诶，不对，不是最近，是这一学期 最喜欢的张敬轩的，泳儿声音好好听，可是大多是翻唱，可惜 好想唱粤语歌，却不会粤语，难过 今天单曲循环的是薛凯琪的奇洛李维斯回信 人人都怕难...



若看见请回信 (/u/71492613d6cf?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/30587c864613?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

**锤子手机，只有情怀？ (/p/30587c864613?utm\_campaign=maleskine&ut...**

声明：文中部分内容凭个人记忆，无法确保准确性（孙大贺）经常听身边同学谈到罗永浩和锤子手机，毕竟罗永浩一直是一个处在风口浪尖的新闻人物。2003年就被一些互联网论坛评为年度风云人物（当时排名第...



孙大贺 (/u/d4f09f84dc1c?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

**电子行业销售小白，如何快速成为销售高手？ (/p/7b131075c8b8?utm\_ca...**

真心英雄这首歌唱着：把握生命里的每一分钟，全力以赴我们心中的梦，不经历风雨，怎么见彩虹，没有人能随随便便成功！冰冰那句励志的话：“别低头，皇冠会掉；别流泪，坏人会笑；万箭穿心，习惯就好...”...



减肥百晓生 (/u/4f250a1c65dc?

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/11da648d84ba?

utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)



没有人会太关心你，除非你很漂亮或濒临死亡。(lp/11...

没有人会太关心你，除非你很漂亮或濒临死亡。



呜呼 (/u/c5cd7ea7bfed?



utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

