代码编织梦想

**Android进阶——WMS与AMS浅析-爱代码爱编程**

category: [android](https://icode.best/category/android) Posted on2021-06-29

1.简介

可以毫不夸张的说，Android的framework层主要是由WMS、AMS还有View所构成，这三个模块穿插交互在整个framework中，掌握了它们之间的关系和每一个逻辑步骤，你对framework的了解至少有百分之五十

AMS是Android中最核心的服务，主要负责系统中四大组件的启动、切换、调度及应用进程的管理和调度等工作，其职责与操作系统中的进程管理和调度模块相类似，因此它在Android中非常重要。

在没有深入了解AMS与WMS前，对它俩负责的功能一直很模糊，一直搞不清区别。所以在深入理解其中之一时，先对它们在Android中各自负责的任务有个基本的了解和区分。

* AMS和WMS都属于Android中的系统服务，系统服务有很多，它们在Android体系架构中都属于同一层次，所以最好在深入了解它们各自的运行机理前，对系统的宏观架构能够有所掌握，也就是在自己的脑海中能够搭建一个简单的模型和框架，搞清楚它们在系统中所扮演的角色。

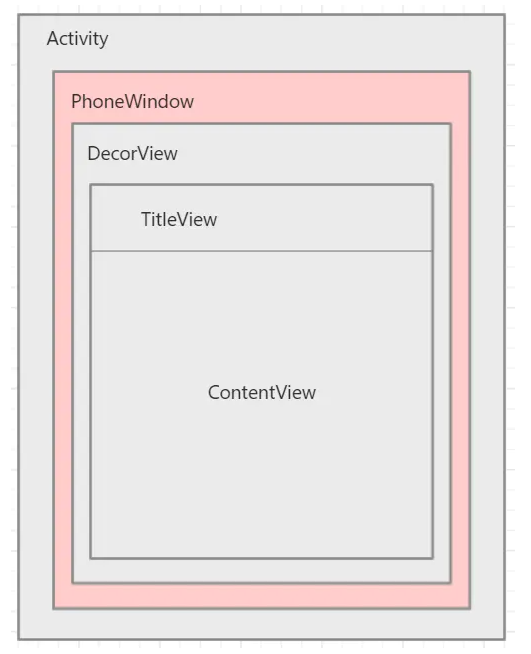
下面先简单对AMS与WMS作个简单的介绍:

**Activity与WIndow：**

* Activity只负责生命周期和事件处理
* Window只控制视图
* 一个Activity包含一个Window，如果Activity没有Window，那就相当于Service

**AMS与WMS：**

* AMS统一调度所有应用程序的Activity
* WMS控制所有Window的显示与隐藏以及要显示的位置

在视图层次中，Activity在WIndow之上，如下图  


2. WMS

WindowManagerService服务的实现是相当复杂的，毕竟它要管理的整个系统所有窗口的UI，而在任何一个系统中，窗口管理子系统都是极其复杂的。

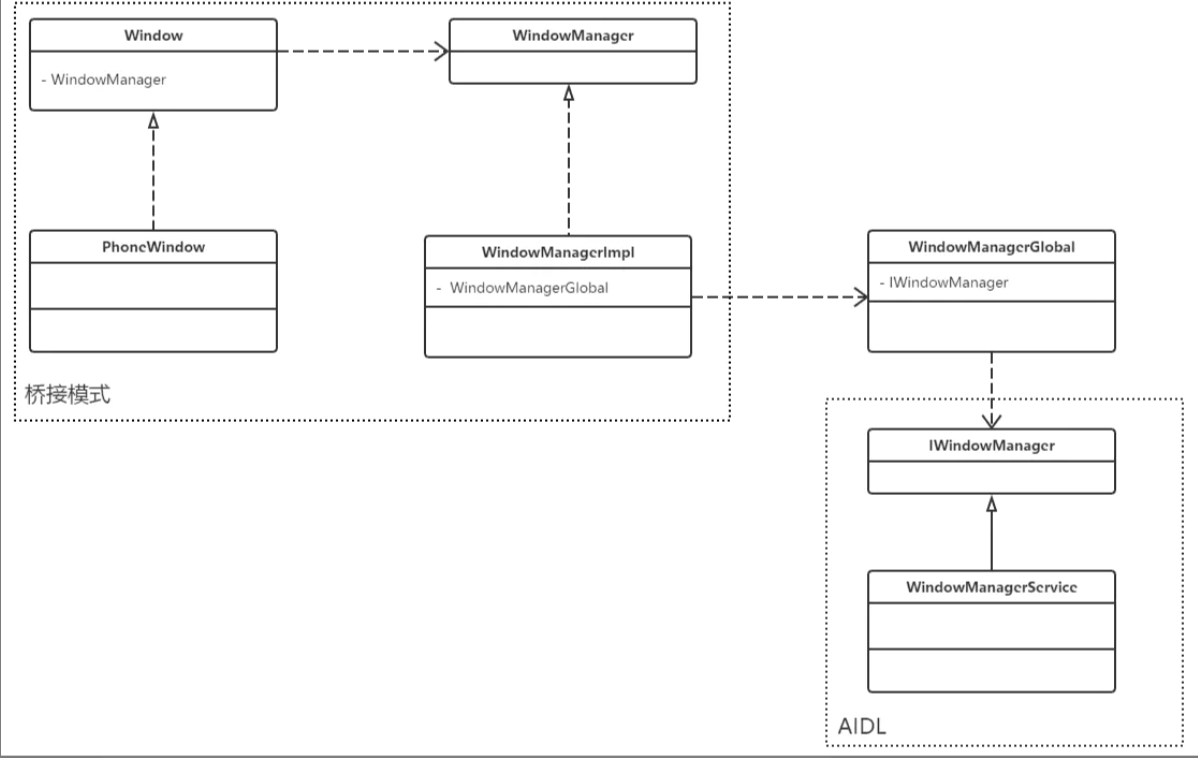
**作用：**

* 为所有窗口分配Surface。客户端向WMS添加一个窗口的过程，其实就是WMS为其分配一块Suiface的过程，一块块Surface在WMS的管理下有序的排布在屏幕上。Window的本质就是Surface。
* 管理Surface的显示顺序、尺寸、位置
* 管理窗口动画
* 输入系统相关：WMS是派发系统按键和触摸消息的最佳人选，当接收到一个触摸事件，它需要寻找一个最合适的窗口来处理消息，而WMS是窗口的管理者，系统中所有的窗口状态和信息都在其掌握之中，完成这一工作不在话下。

**什么是Window?**

* “Window”表明它是和窗口相关的，“窗口”是一个抽象的概念，从用户的角度来讲，它是一个“界面”；从SurfaceFlinger的角度来看，它是一个Layer，承载着和界面有关的数据和属性；从WMS角度来看，它是一个WIndowState，用于管理和界面有关的状态。
* 在《深入理解Android内核设计思想》一书中看到一个比喻非常好，整个界面就像由N个演员参与的话剧：SurfaceFling是摄像机，它只负责客观的捕捉当前的画面，然后真实的呈现给观众；WMS就是导演，它要负责话剧的舞台效果、演员站位；ViewRoot就是各个演员的长相和表情，取决于它们各自的条件与努力。可见，WMS与SurfaceFling的一个重要区别就是——后者只做与“显示”相关的事情，而WMS要处理对输入事件的派发。
* Android支持的窗口类型很多，统一可以分为三大类，另外各个种类下还细分为若干子类型，且都在WindowManager.java中有定义。

**类间关系：**  
上面那些都是对WMS相关功能的介绍，对WMS有个感性的认识，现在开始进入相关类。

主要使用的桥接模式，具体自行学习  
  
**一个问题：**

WMS是系统服务，有SystemServer负责启动，启动时机相对较晚，那么在WMS运行之前，终端显示屏就一团黑？

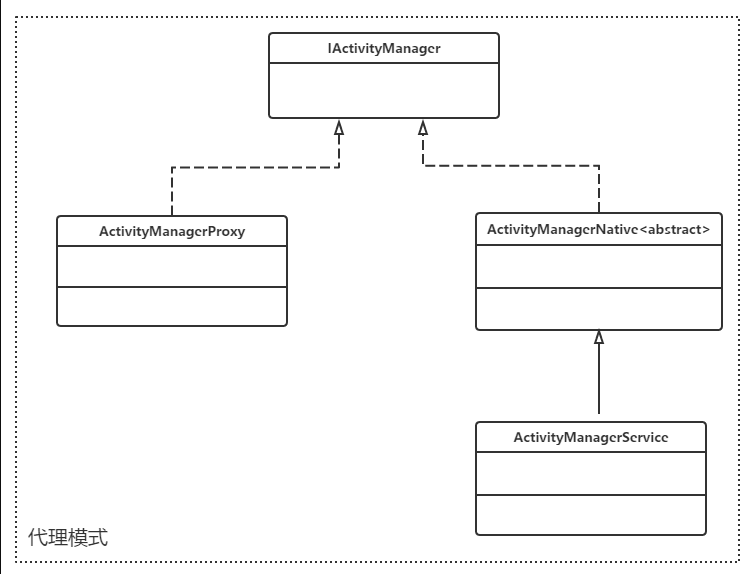
* 在WMS启动之前，系统只需显示开机动画，它们都有特殊的方式来向屏幕输出图像，比如直接通过OpenGL ES与SurfaceFling的配合来完成。
* 这也从侧面告诉我们，要想在Android上显示UI，并不一定要通过WMS。

3. AMS

**作用：**

* 统一调度所有应用程序的Activity的生命周期
* 启动或杀死应用程序的进程
* 启动并调度Service的生命周期
* 注册BroadcastReceiver，并接收和分发Broadcast
* 启动并发布ContentProvider
* 调度task
* 处理应用程序的Crash
* 查询系统当前运行状态

**类间关系：**

主要使用了代理模式，具体可以看我的另外一篇博文：[设计模式——代理模式](https://icode.best/go?go=aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L2x5MDcyNG9rL2FydGljbGUvZGV0YWlscy8xMTY3NTEwNjYv).  


* ActivityManagerProxy和ActivityManagerNative都实现了IActivityManager，ActivityManagerProxy就是代理部分，ActivityManagerNative就是实现部分
* 但ActivityManagerNative是个抽象类，并不处理过多的具体逻辑，大部分具体逻辑是由ActivityManagerService承担，这就是为什么我们说真实部分应该为ActivityManagerService。

**启动流程：**

* 第一阶段：启动ActivityManagerService。
* 第二阶段：调用setSystemProcess。
* 第三阶段：调用installSystemProviders方法。
* 第四阶段：调用systemReady方法。

**工作流程：**

* AMS的工作流程，其实就是Activity的启动和调度的过程，所有的启动方式，最终都是通过Binder机制的Client端，调用Server端的AMS的startActivityXXX()系列方法。所以可见，工作流程又包括Client端和Server端两个。

**Client端流程：**

* auncher主线程捕获onClick()点击事件后，调用Launcher.startActivitySafely()方法。Launcher.startActivitySafely()内部调用了Launcher.startActivity()方法，Launcher.startActivity()内部调用了Launcher的父类Activity的startActivity()方法。
* Activity.startActivity()调用Activity.startActivityForResult()方法，传入该方法的requestCode参数若为-1，则表示Activity启动成功后，不需要执行Launcher.onActivityResult()方法处理返回结果。
* 启动Activity需要与系统ActivityManagerService交互，必须纳入Instrumentation的监控，因此需要将启动请求转交instrumentation，即调用Instrumentation.execStartActivity()方法。
* nstrumentation.execStartActivity()首先通过ActivityMonitor检查启动请求，然后调
* ActivityManagerNative.getDefault()得到ActivityManagerProxy代理对象，进而调用该代理对象的startActivity()方法。
* ActivityManagerProxy是ActivityManagerService的代理对象，因此其内部存储的是BinderProxy，调用ActivityManagerProxy.startActivity()实质是调用BinderProxy.transact()向Binder驱动发送START\_ACTIVITY\_TRANSACTION命令。Binder驱动将处理逻辑从Launcher所在进程切换到ActivityManagerService所在进程。

**Server端流程：**

启动Activity的请求从Client端传递给Server端后，便进入了启动应用的七个阶段，这里也是整理出具体流程，细节可以参考《Android的设计与实现》第十一章内容。

预启动：

* ActivityManagerService.startActivity()
* ActivityStack.startActivityMayWait()
* ActivityStack.startActivityLocked()
* ActivityStack.startActivityUncheckedLocked()
* ActivityStack.startActivityLocked()（重载）
* ActivityStack.resumeTopActivityLocked()

暂停：

* ActivityStack.startPausingLocked()
* ApplicationThreadProxy.schedulePauseActivity()
* ActivityThread.handlePauseActivity()
* ActivityThread.performPauseActivity()
* ActivityManagerProxy.activityPaused()
* completePausedLocked()

启动应用程序进程：

* 第二次进入ActivityStack.resumeTopActivityLocked()
* ActivityStack.startSpecificActivityLocked()
* startProcessLocked()
* startProcessLocked()（重载）
* Process.start()

加载应用程序Activity：

* ActivityThread.main()
* ActivityThread.attach()
* ActivityManagerService.attachApplication()
* ApplicationThread.bindApplication()
* ActivityThread.handleBindApplication()

显示Activity：

* ActivityStack.realStartActivityLocked()
* ApplicationThread.scheduleLaunchActivity()
* ActivityThead.handleLaunchActivity()
* ActivityThread.performLaunchActivity()
* ActivityThread.handleResumeActivity()
* ActivityThread.performResumeActivity()
* Activity.performResume()
* ActivityStack.completeResumeLocked()

Activity Idle状态的处理

停止源Activity：

* ActivityStack.stopActivityLocked()
* ApplicationThreadProxy.scheduleStopActivity()
* ActivityThread.handleStopActivity()
* ActivityThread.performStopActivityInner()

版权声明：本文为博主原创文章，遵循[CC 4.0 BY-SA](https://icode.best/go?go=aHR0cDovL2NyZWF0aXZlY29tbW9ucy5vcmcvbGljZW5zZXMvYnktc2EvNC4wLw==)版权协议，转载请附上原文出处链接和本声明。

本文链接：[https://blog.csdn.net/ly0724ok/article/details/118315548](https://icode.best/go?go=aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L2x5MDcyNG9rL2FydGljbGUvZGV0YWlscy8xMTgzMTU1NDg=)

Post navigation

[Hilt 使用姿势全解析-爱代码爱编程](https://icode.best/i/84110042086205)

[AspectJ在Android 中的使用攻略-爱代码爱编程](https://icode.best/i/84190042086338)

[简述Android framework之AMS、PMS、WMS-爱代码爱编程](https://icode.best/i/77472931775880)

category: [Android Fram](https://icode.best/category/Android+Fram)category: [android面试技术及](https://icode.best/category/android%E9%9D%A2%E8%AF%95%E6%8A%80%E6%9C%AF%E5%8F%8A)

 Posted on2020-03-21

服务端 主要是ActivityManagerService（AMS）， WindowManagerService（WMS），PackageM anerService（PMS） AMS 主要用于管理所有应用程序的Activity WMS 管理各个窗口，隐藏，显示等 PMS 用来管理跟踪所有应用APK，安装，解析，控制权限等. 还有用来处理触摸消息的两

[CONTINUE READING](https://icode.best/i/77472931775880)

[应用程序Activity窗口与WMS、AMS的通信分析（Android Q）-爱代码爱编程](https://icode.best/i/77618935334347)

tag: [android](https://icode.best/tag/android)category: [通信](https://icode.best/category/%E9%80%9A%E4%BF%A1)tag: [Activity](https://icode.best/tag/Activity)tag: [窗口](https://icode.best/tag/%E7%AA%97%E5%8F%A3)tag: [window](https://icode.best/tag/window)category: [android底层原理解](https://icode.best/category/android%E5%BA%95%E5%B1%82%E5%8E%9F%E7%90%86%E8%A7%A3)

 Posted on2020-11-19

应用程序Activity窗口与WMS、AMS的通信分析 Activity 组件在其窗口创建完成后，就会请求与 WindowManagerService（WMS）建立连接，WMS 会为其新增一个 WindowState 对象，用来描述它的窗口状态。 Activity 到 WMS 连接：当应用程序启动第一个 Activity 时，应用程序进程从 WMS 获

[CONTINUE READING](https://icode.best/i/77618935334347)

[Android窗口管理2　WMS启动过程-爱代码爱编程](https://icode.best/i/13887637673118)

category: [Android高级](https://icode.best/category/Android%E9%AB%98%E7%BA%A7)

 Posted on2021-01-20

一　概述 在 Android 系统中，从设计的角度来看，窗口管理系统是基于 C/S 模式的。整个窗口系统分为服务端和客户端两大部分，客户端负责请求创建窗口和使用窗口，服务端完成窗口的维护，窗口显示等。 在 Client 端，并不是直接和 WindowManagerService 交互，而是直接和本地对象 WindowManager 交互，然后由 Wind

[CONTINUE READING](https://icode.best/i/13887637673118)

[不服不行！终于有人把AMS和WMS讲明白了！-爱代码爱编程](https://icode.best/i/77665741209802)

category: [移动开发](https://icode.best/category/%E7%A7%BB%E5%8A%A8%E5%BC%80%E5%8F%91)category: [android](https://icode.best/category/android)category: [framework](https://icode.best/category/framework)category: [wms](https://icode.best/category/wms)category: [ams](https://icode.best/category/ams)

 Posted on2021-05-23

现在的移动开发人才市场上，初中级开发者几乎都是人满为患，互联网公司对移动开发人员的招聘要求也越来越高，尤其是那些大厂，非常看重开发者对底层的掌握。 大家可以先来看一组大厂的面试题： AMS是如何启动的？ AMS在Android起到什么作用？ AMS有哪些应用场景？我们是如何应用AMS核心原理的？ WMS的工作原理说说？ 这些底层问题能回答上来吗

[CONTINUE READING](https://icode.best/i/77665741209802)

[android wms各个类的作用,Android APP/AMS/WMS之间交互总结-爱代码爱编程](https://icode.best/i/05148241486385)

category: [android wms各](https://icode.best/category/android+wms%E5%90%84)

 Posted on2021-05-27

注：以下内容基于Android API Version 27(Android 8.1)Linux Kernel 3.18.0 AMS/WMS/APP角色概述 AMS 掌管着所有App进程的启动，App四大组件的启动，App之间及App内部页面之间切换和各种组件的生命周期，AMS不控制窗口的显示，只控制各种组件的创建和运行。 WMS 掌管着

[CONTINUE READING](https://icode.best/i/05148241486385)

前端技术

[BROWSER](https://icode.best/tag/browser) [CHROME](https://icode.best/tag/Chrome) [EDGE](https://icode.best/tag/Edge) [WECHAT](https://icode.best/tag/wechat) [VUE.JS](https://icode.best/tag/vue) [CSS](https://icode.best/tag/css) [REACT](https://icode.best/tag/React) [FLUTTER](https://icode.best/tag/Flutter) [ANGULAR](https://icode.best/tag/Angular) [NEXT.JS](https://icode.best/tag/Next.js) [IONIC](https://icode.best/tag/Ionic) [NUXT.JS](https://icode.best/tag/Nuxt.js)

编程语言

[C](https://icode.best/tag/c) [JAVA](https://icode.best/tag/Java) [PYTHON](https://icode.best/tag/python) [PHP](https://icode.best/tag/php) [JAVASCRIPT](https://icode.best/tag/js) [NODE.JS](https://icode.best/tag/nodejs) [HTML](https://icode.best/tag/html) [.NET](https://icode.best/tag/dotnet) [RUST](https://icode.best/tag/rust) [C#](https://icode.best/tag/C) [KOTLIN](https://icode.best/tag/kotlin) [TYPESCRIPT](https://icode.best/tag/typescript) [LUA](https://icode.best/tag/lua)

系统软件

[SERVER](https://icode.best/tag/server) [DNS](https://icode.best/tag/dns) [NGINX](https://icode.best/tag/nginx) [APACHE](https://icode.best/tag/apache) [MYSQL](https://icode.best/tag/mysql) [POSTGRESQL](https://icode.best/tag/postgresql) [DOCKER](https://icode.best/tag/docker) [KUBERNETES](https://icode.best/tag/Kubernetes) [MONGODB](https://icode.best/tag/mongodb) [REDIS](https://icode.best/tag/redis) [ELASTICSEARCH](https://icode.best/tag/elasticsearch) [LINUX](https://icode.best/tag/linux) [UBUNTU](https://icode.best/tag/ubuntu) [DEBIAN](https://icode.best/tag/debian) [CENTOS](https://icode.best/tag/centos) [FREEBSD](https://icode.best/tag/freebsd)

RECENTLY POSTS

* [正确的使用Java 中的日志-爱代码爱编程](https://icode.best/i/84474842086118)
* [Lifecycle添加依赖-爱代码爱编程](https://icode.best/i/89691442085876)
* [android studio 打包aar 与 引入aar包-爱代码爱编程](https://icode.best/i/89647442085793)
* [Android 个人中心页面前端代码-爱代码爱编程](https://icode.best/i/89594642085765)
* [Android Studio OpenCV 4.5.2环境搭建-爱代码爱编程](https://icode.best/i/89511542085686)
* [Kotlin 协程，怎么开始的又是怎么结束的？原理讲解！-爱代码爱编程](https://icode.best/i/83960042086557)
* [Android进阶——AMS浅析（1）-爱代码爱编程](https://icode.best/i/90134242086673)
* [ILRuntime--构造方法成员调用--03-爱代码爱编程](https://icode.best/i/83300542087049)
* [低端机Arouter初始化耗时分析与优化-爱代码爱编程](https://icode.best/i/90210942087053)
* [给自己立一个要求，如何保持进化？-爱代码爱编程](https://icode.best/i/92131742091709)

[爱代码爱编程](https://icode.best/)