# Android Activity 生命周期详解及监听

[](https://juejin.cn/user/3658822686609774)

# 前言

系列文章：

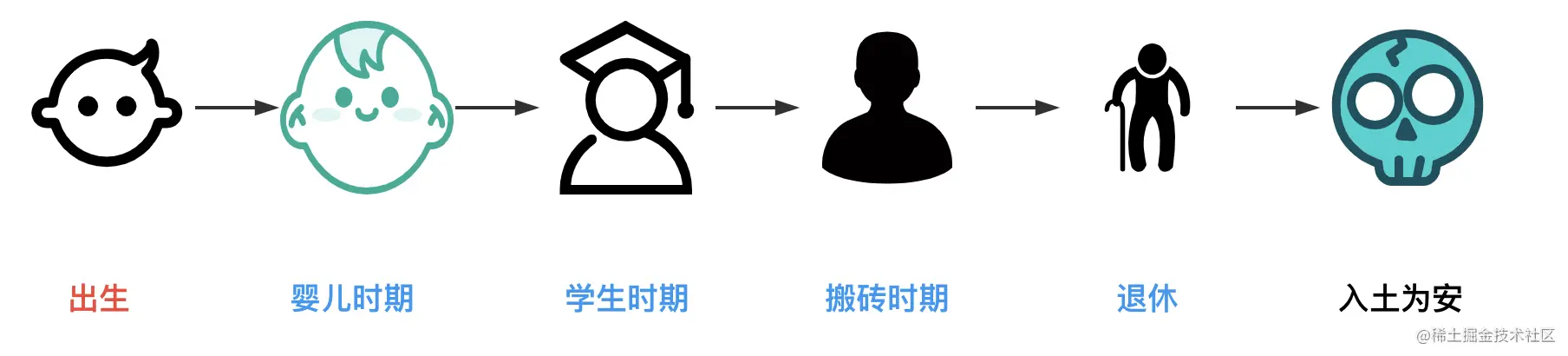
[Android Activity 与View 的互动思考](https://juejin.cn/post/7035235129479921671)  
[Android Activity 生命周期详解及监听](https://juejin.cn/post/7038064327814807589)  
[Android onSaveInstanceState/onRestoreInstanceState 原来要这么理解](https://juejin.cn/post/7040819115874844709)  
[Android Fragment 要你何用？](https://juejin.cn/post/7047891752245526564)

当你编写第一个Android "Hello World" 时，已经不知不觉地与Activity生命周期打交道。这部分是是Android 最基础的知识之一，也是面试的常客，同时也是理解Jetpack 组件的基础。  
网上关于此的文章数不胜数，但大部分只流于表象，没有系统性分析，看过之后容易忘却。本系列将会着重分析由此衍生的一系列知识点。  
通过本篇文章，你将了解到：

1、什么是生命周期？  
2、谁是生命周期的幕后黑手？  
3、如何感知生命周期

# 1、什么是生命周期

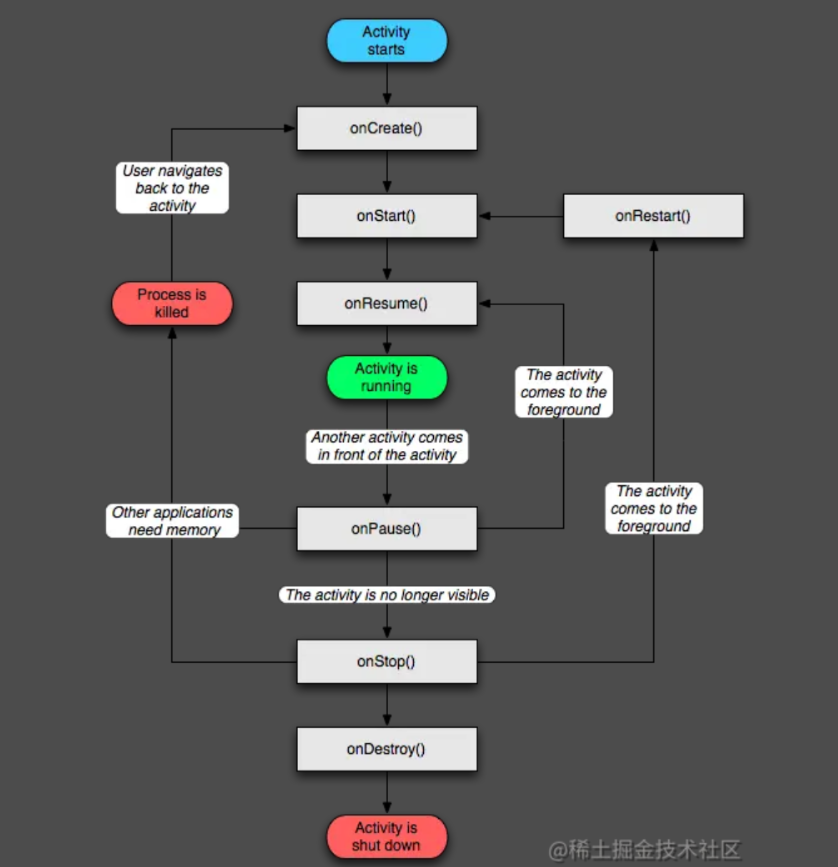
这个名字并不是计算机独创的术语，任何有生命的东西都有周期，从出生到死亡，以人为例：



这是一个人的成长轨迹，也是一个人的生命周期，在不同的年龄段表现各异。  
俗话说：人活一世，草木一秋。  
佛家说的轮回，当一个人入土为安，假设有轮回，在下一个轮回里再经历出生到死亡，这个人还是上一世的那个人吗？  
同样的草木凋谢了，在春天再长出来时，还是曾经的它吗？  
不是，它们都是新的生命。

## Activity 的生命周期

先来看官方经典图：



Activity 作为对象，它是有"生命的"，类似人和草木，它的存活周期就是它的生命周期。

**为什么生命周期要分不同的阶段？**  
就像人一样：

1、当处在婴儿阶段，自己每天喝喝奶，没事就哭闹一下，大人就会安抚你。  
2、当处在学生时代，就认真做题，大家把你当小孩看待，不用你出门赚钱，景点都能给你学生票五折优惠。  
3、当处在工作时代，就好好搬砖，老板会给你福报。  
4、...

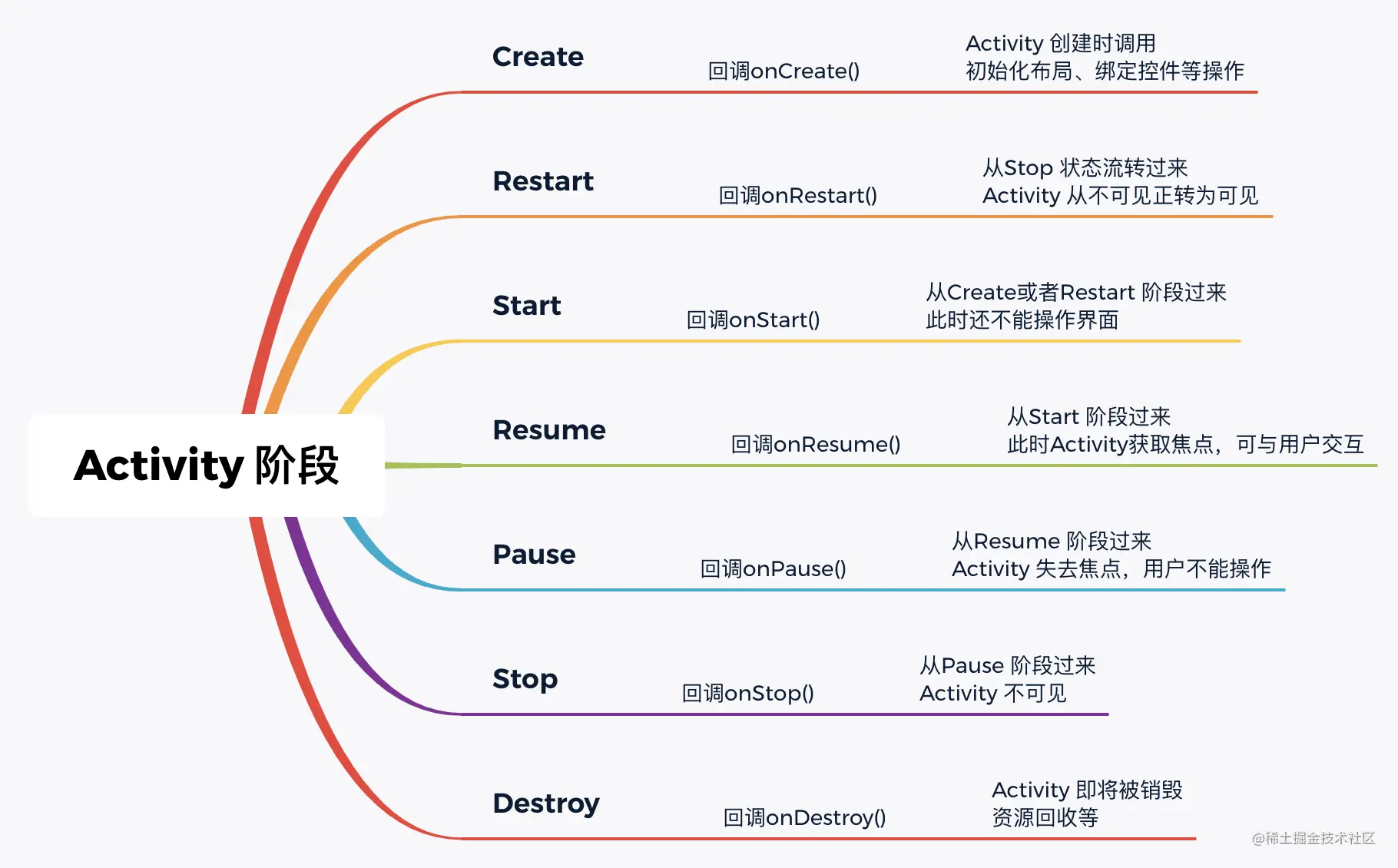
同样的，对于Activity 来说，在不同的阶段自己内部所做的事情不同，外部监听到Activity的处在不同阶段将会采取不同的措施。  
总结来说：

生命周期分阶段既是为了"生命体内部"流转的需要，也是为了让外部知道"生命体"当前所处的阶段进而做出相应的决策。

与人、草木一样，当Activity 经历了一个生命周期后，再次重建时已经不是当初的对象了。

## Activity 生命周期不同阶段

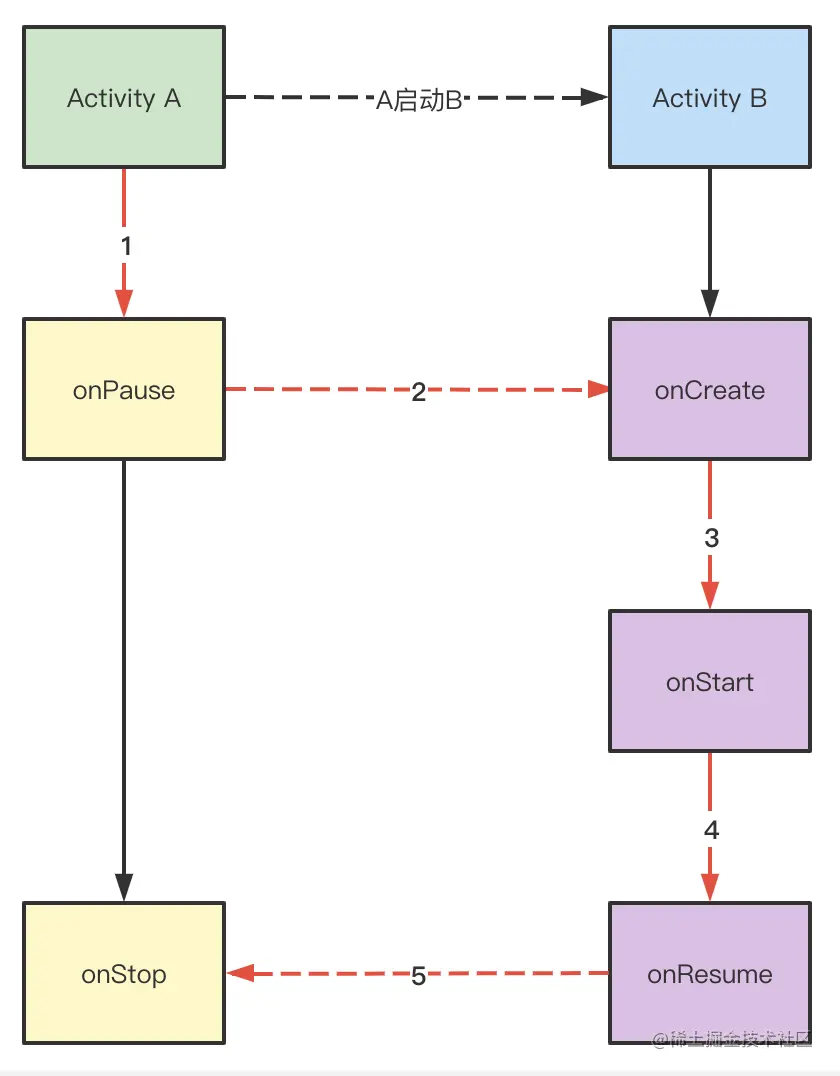
主要分为：



可以看出Create 对应 Destroy，Start对应Stop，Resume对应Pause。  
既然每个阶段都有回调，那么只需要重写对应的回调方法，添加打印即可测试不同的操作后Activity处在哪个阶段。

接下来看看Activity 常见的切换场景：

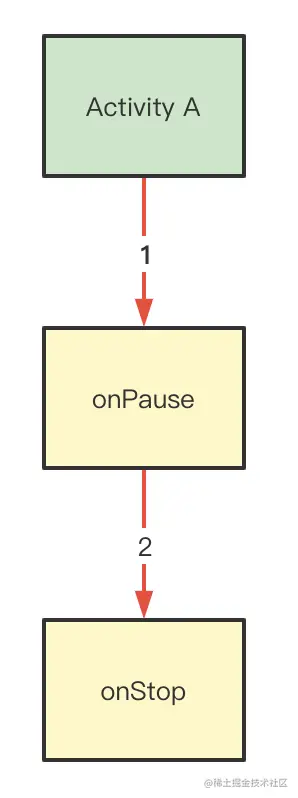
### Activity A 启动Activity B



图上数字表示发生的顺序，具有时间上的先后顺序。

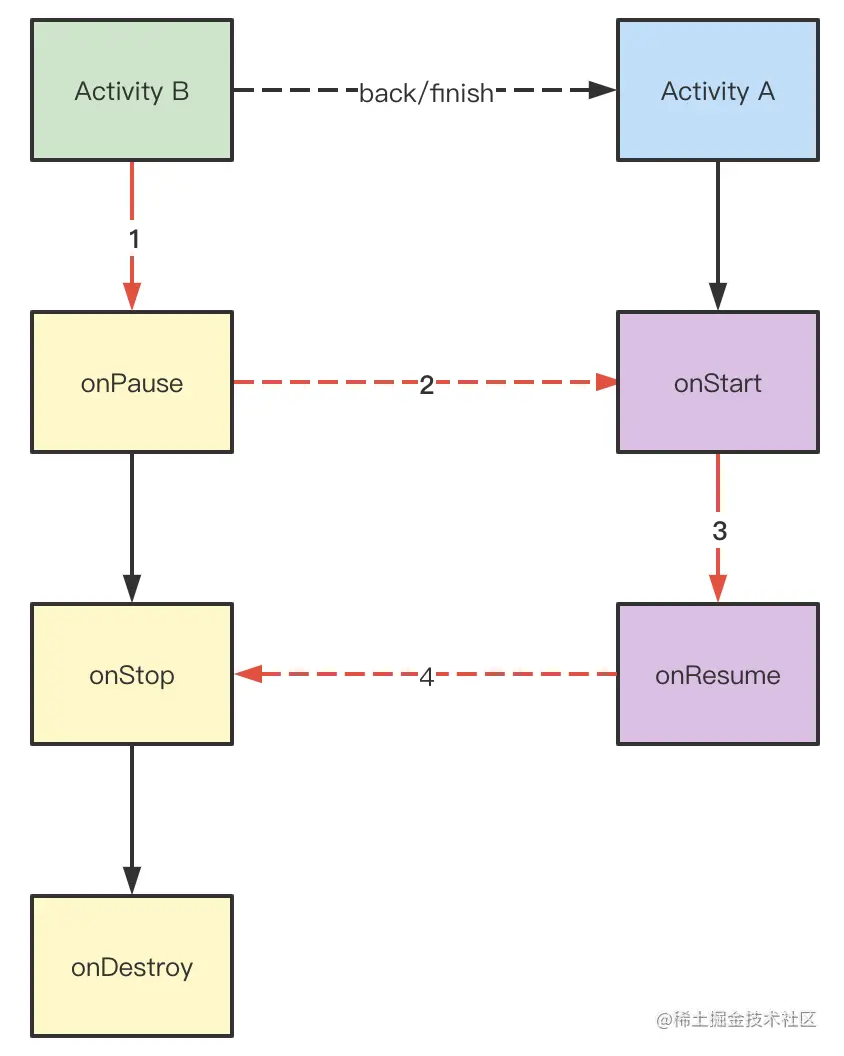
### 从Activity A 按Home回到桌面

回到桌面或者最近任务列表：



### Activity B 按back 键/调用finish

此时Activity B处在栈顶位置，而Activity A 位于它之下。



### Pause 阶段

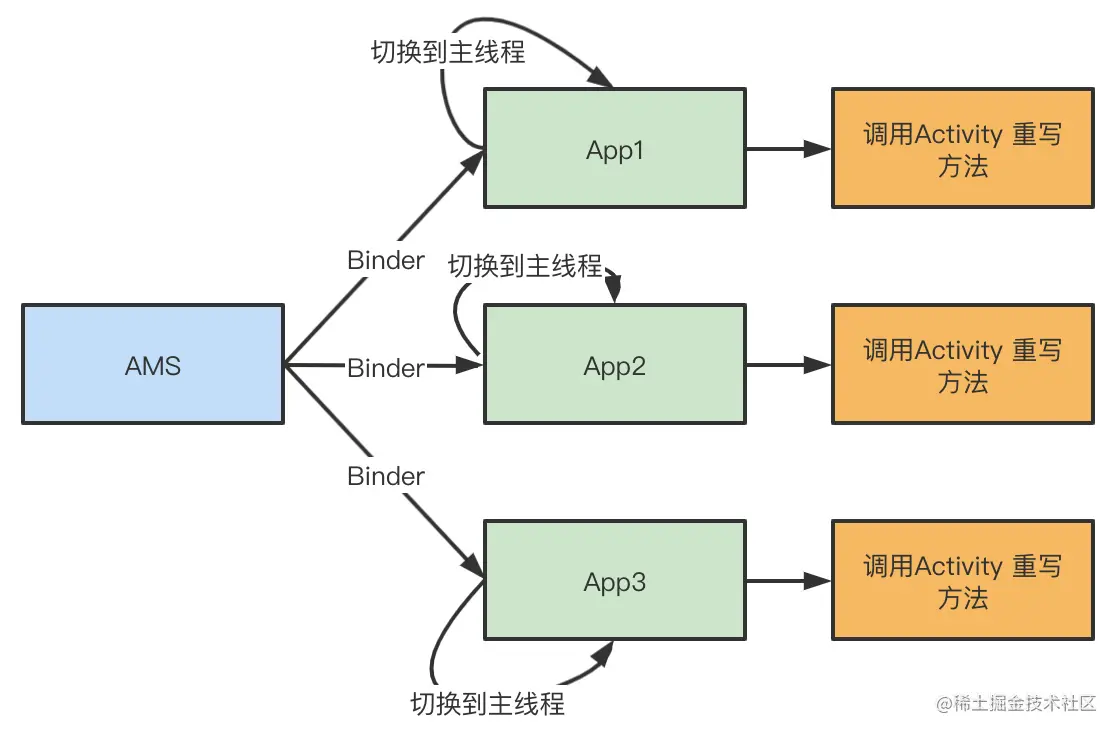
上面列出的场景都是onPause-->onStop，那么Activity 是否存在只停留在onPause的阶段？ Pause阶段是Activity 可见，但是没有焦点因此无法和用户交互。  
可见意味着它的上面有Activity，并且上面的Activity 是透明的，才可以看见底下的Activity。 此种情况下，底下的Activity 处在Pause阶段。



如上图，红色区块是新启动的Activity，该Activity 没有全屏并且透明，可以看到底下Activity 内容，此时它处在Pause阶段。

# 2、谁是生命周期的幕后黑手？

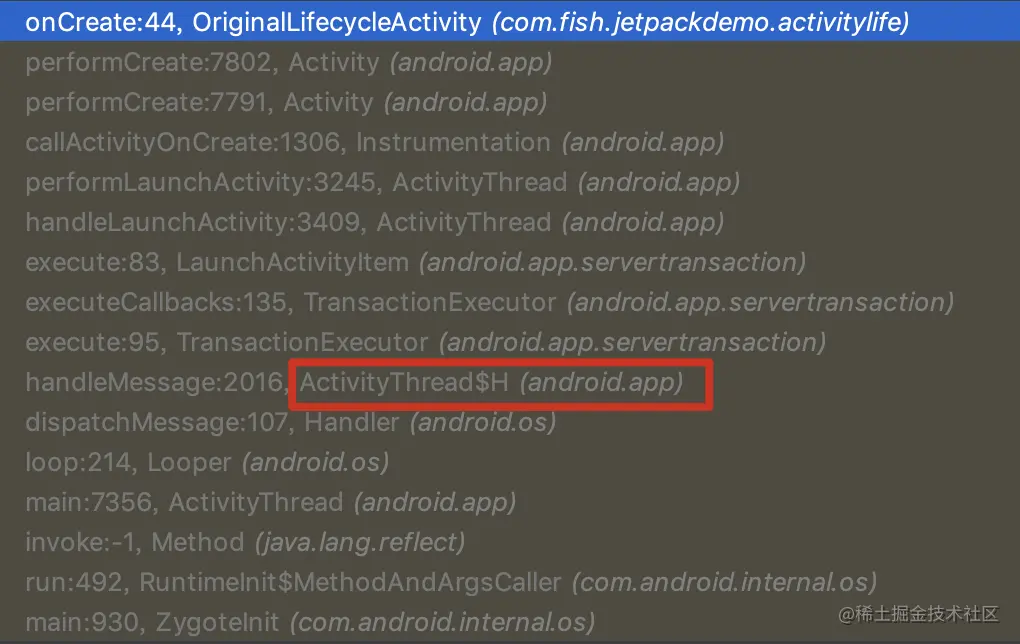
我们知道Android 是事件驱动的，通过Loop来循环执行MessageQueue里的Message。  
问题来了：**onCreate、onStart、onStop 是在同一个Message里执行的吗？**  
要弄清楚这个问题，需要知道回调这些方法的堆栈。众所周知，四大组件都是由AMS（Activity Manager Service)控制的，而AMS 运行在system\_server 进程，与咱们的App是不同的进程，AMS 通过Binder 与各个App进行通信。



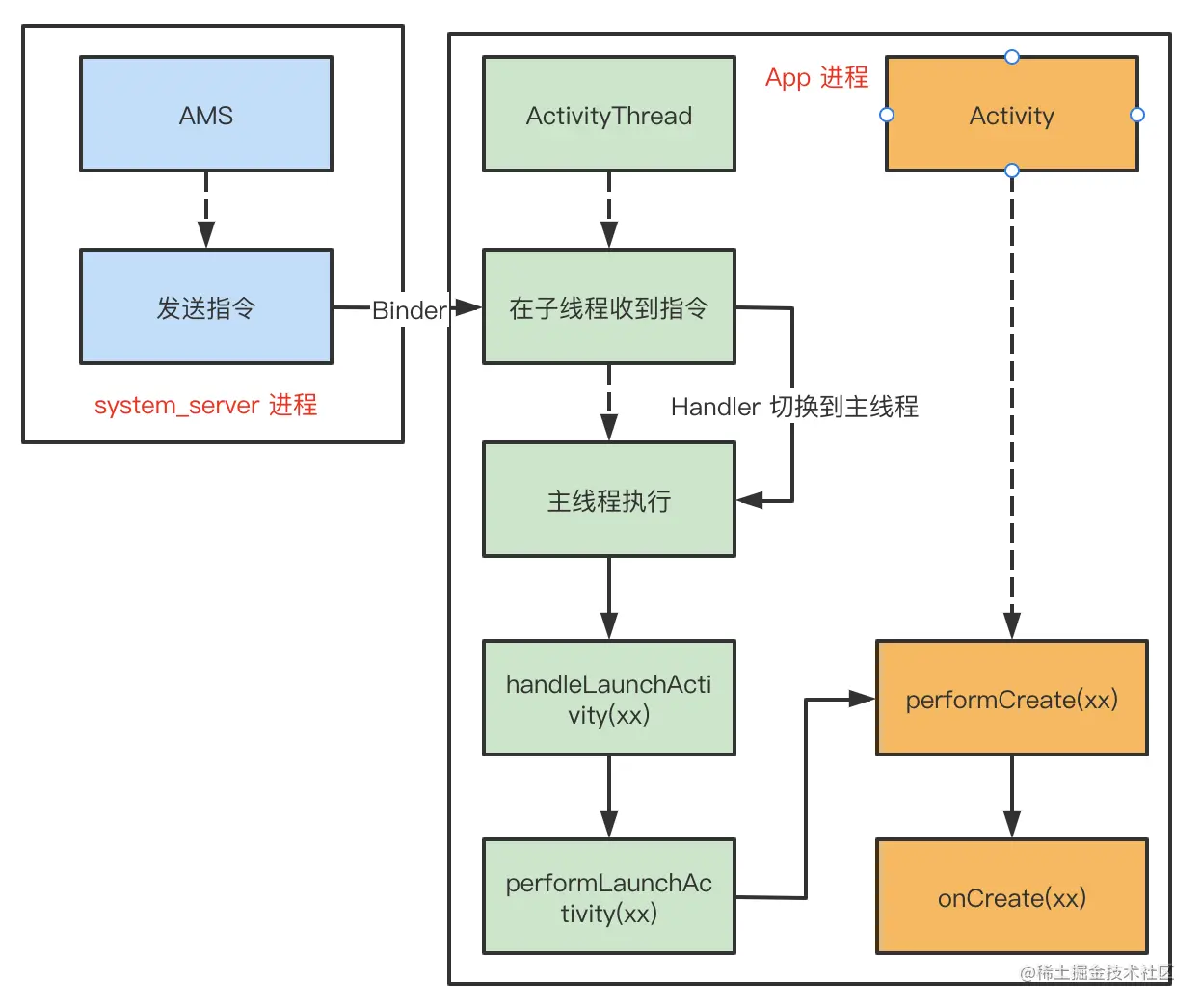
不清楚的可移步：[Android 四大组件通信核心](https://juejin.cn/post/7015964509625909279) 接下来探究AMS 如何控制Activity 生命周期回调的。

## onCreate 调用堆栈

在onCreate()里打个断点，调用栈如下：



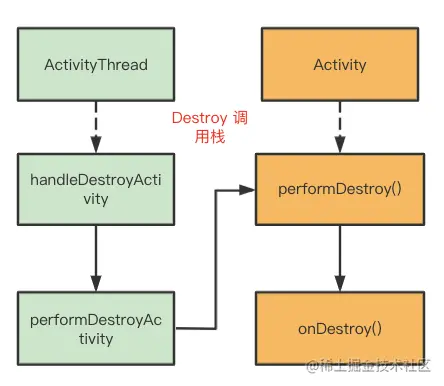
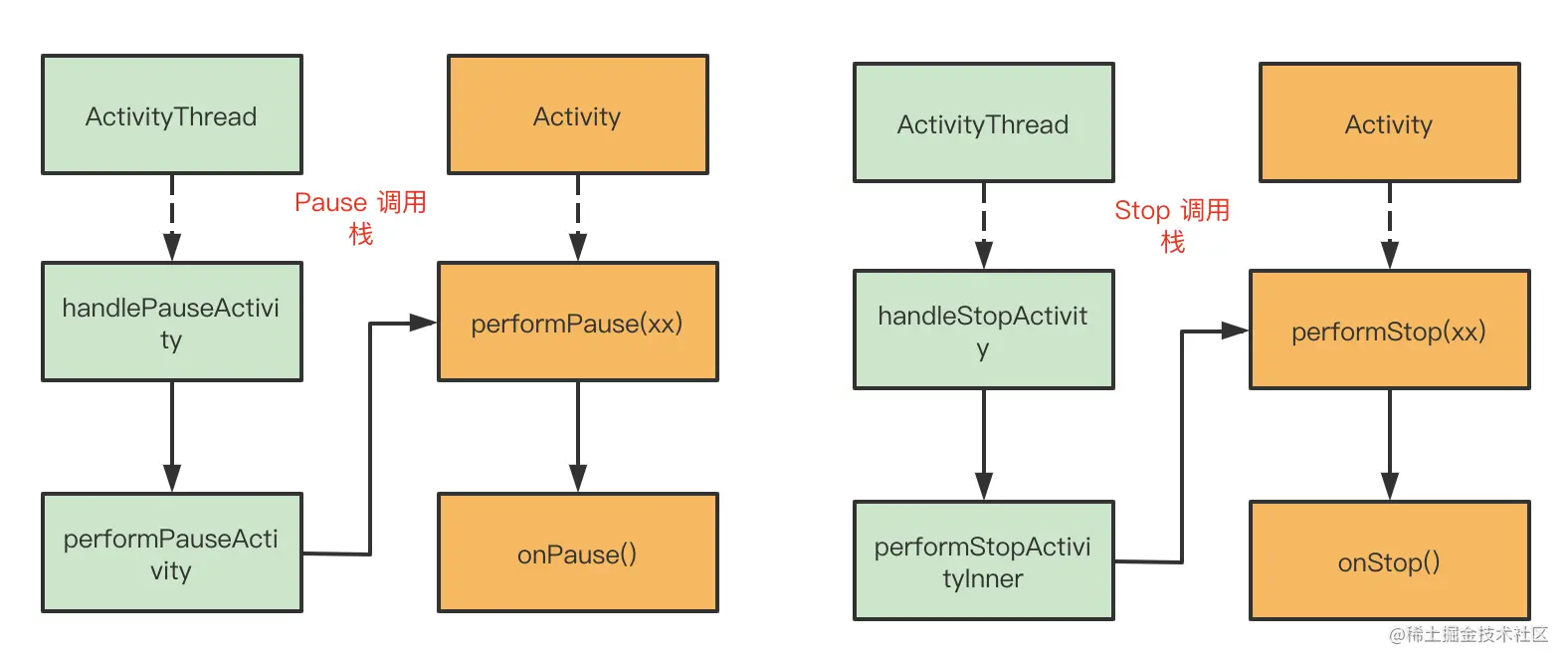
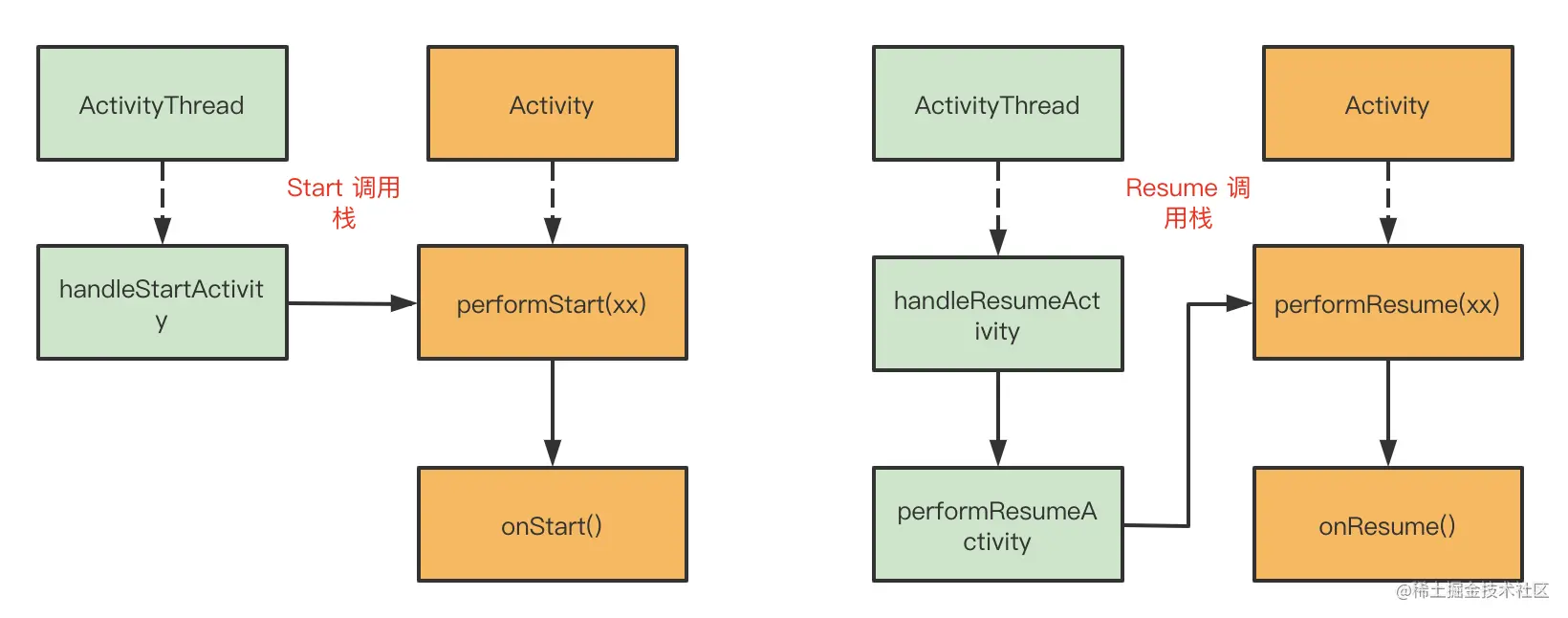
可以看出入口是在ActivityThread.java 类里。



1、App进程启动的时，将ApplicationThread 传递给AMS，类似向AMS 注册回调。  
2、AMS 通过ApplicationThread 通知App Activity 的生命周期。  
3、App 切换到主线程执行对应的方法，最终会执行Activity 的onCreate()、onStart()、onResume()等方法。

## 其它阶段的调用栈

与onCreate 类似，onStart、onResume、onPause、onStop、onDestroy 阶段同样经历了AMS--->App--->切换到主线程--->回调对应方法。  
为方便起见，省去AMS 过程，用图表示如下：



由以上分析可知：

1、Activity 生命周期的幕后黑手是AMS。  
2、各个阶段的调用栈很相似，只是变了方法名而已。  
3、onCreate、onStart、onStop 不是在同一个Message里执行的，但是都是在主线程执行。

# 3、如何感知生命周期

## 注册监听

知道了生命周期的来龙去脉，要监听生命周期就变得非常简单了，只需要在Activity 里重写对应阶段的回调方法即可，这也是我们最基础的选择。  
现在有个需求：监听App处在前台还是后台？  
Activity 处在后台，那么此时阶段为Stop，回到前台那么此时处会经历Start阶段，在进入Start 阶段时记录+1，在进入Stop阶段时记录-1。若最终结果记录=1，说明App在前台。  
这么看来，我们需要一个公共的Activity来进行统一记录，每个新建的Activity 都继承自它，这么做虽然能满足需求，但也有弊端：一是工作量大，二是也不灵活。  
实际上还有更好的方式来全局监听各个Activity 生命周期。

#Application.java

public void registerActivityLifecycleCallbacks(ActivityLifecycleCallbacks callback) {

synchronized (mActivityLifecycleCallbacks) {

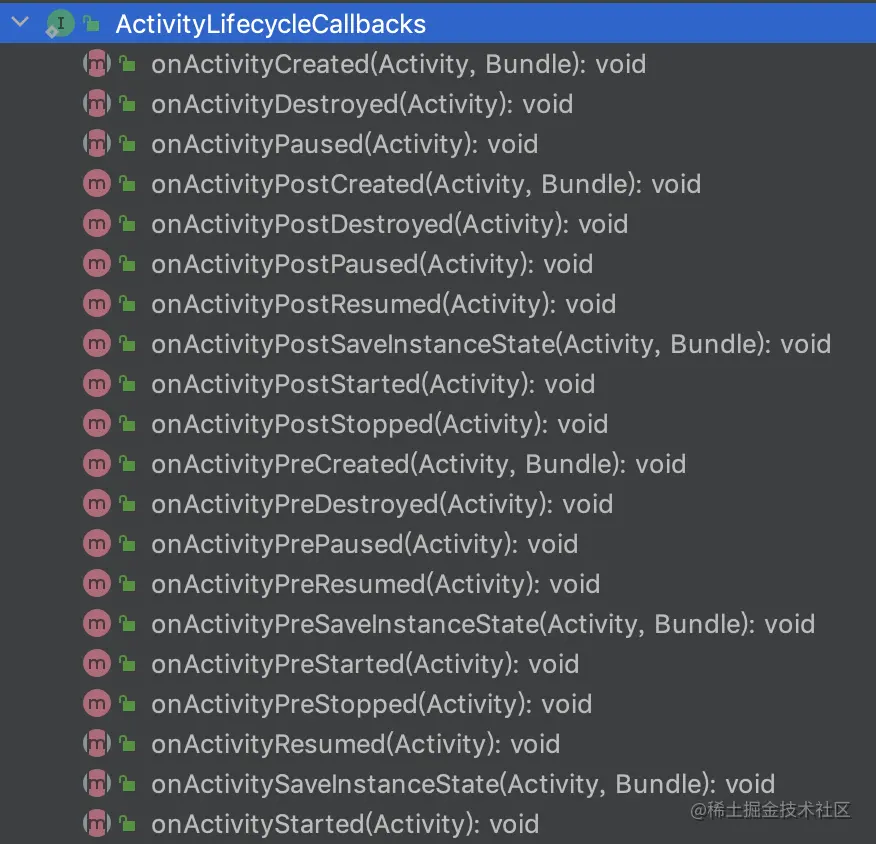
mActivityLifecycleCallbacks.add(callback);

}

}

复制代码

Application.java 里有个方法，顾名思义用来注册Activity 生命周期回调的，这个callback 会加入到回调List：mActivityLifecycleCallbacks 里。  
先看看ActivityLifecycleCallbacks 的构成：



它是个接口，乍一看有很多方法，实际上都是有规律可循的，总结下来就是监听了：

Create、Start、Resume、Pause、Stop、Destroy、SaveInstanceState

各个阶段的回调。  
而每个阶段分为三个小阶段：

此阶段之前、阶段中、此阶段之后

如对于Create阶段，定义了三个回调方法：

#Application.java

public interface ActivityLifecycleCallbacks {

//Create阶段之前

default void onActivityPreCreated(@android.annotation.NonNull Activity activity,

@android.annotation.Nullable Bundle savedInstanceState) {

}

//Create阶段当中

void onActivityCreated(@android.annotation.NonNull Activity activity, @android.annotation.Nullable Bundle savedInstanceState);

//Create阶段之后

default void onActivityPostCreated(@android.annotation.NonNull Activity activity,

@android.annotation.Nullable Bundle savedInstanceState) {

}

//...

}

复制代码

而只有处在当前阶段的方法必须要实现类实现，其它两个方法都是默认的(default 修饰)。

## 生命周期回调原理

注册了监听后，来分析一下该回调是如何被调用的。  
既然是监听Activity 生命周期，那么应当从Activity.java 里入手。  
记得咱们之前分析的Activity 回调方法调用栈，还是以onCreate为例：

#Activity.java

protected void onCreate(@android.annotation.Nullable Bundle savedInstanceState) {

...

mFragments.dispatchCreate();

//分发

dispatchActivityCreated(savedInstanceState);

...

}

private void dispatchActivityCreated(@android.annotation.Nullable Bundle savedInstanceState) {

//调用Application里的方法

getApplication().dispatchActivityCreated(this, savedInstanceState);

}

复制代码

而在Application.java里的实现为：

#Application.java

void dispatchActivityCreated(@android.annotation.NonNull Activity activity,

@android.annotation.Nullable Bundle savedInstanceState) {

//收集注册列表，即是将mActivityLifecycleCallbacks(List)，转为数组

Object[] callbacks = collectActivityLifecycleCallbacks();

if (callbacks != null) {

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

//调用对应的回调方法

((ActivityLifecycleCallbacks)callbacks[i]).onActivityCreated(activity,

savedInstanceState);

}

}

}

复制代码

虽然以上只是分析了onCreate，然而其他方法也是同样的套路，大同小异。

1、每当Activity 处在不同的阶段时，会检测Application里是否有监听它(Activity)的生命周期，若是则告知监听者当前所处的阶段。  
2、若是重写了Activity 对应的方法，并添加了打印，那么Application里的回调方法先执行才会执行到Activity里的打印，因为它是在Activity父类里执行的。

以上即是Activity 生命周期的梳理，下篇将会分析 onSaveInstanceState/onRestoreInstanceState 原理以及应用场景，并引入Jetpack ViewModel与之对比。

透明Activity/生命周期监听 demo 请查看：[github合集](https://link.juejin.cn/?target=https%3A%2F%2Fgithub.com%2Ffishforest%2FJetpackDemo%2Ftree%2Fmaster%2Fapp%2Fsrc%2Fmain%2Fjava%2Fcom%2Ffish%2Fjetpackdemo%2Factivitylife)

本文基于Android 10.0

# 持续更新中，和我一起步步为营系统、深入学习Android

1、[Android各种Context的前世今生](https://juejin.cn/post/7015968660179124238)  
2、[Android DecorView 必知必会](https://juejin.cn/post/7015973616659464206)  
3、[Window/WindowManager 不可不知之事](https://juejin.cn/post/7015978746104512548)  
4、[View Measure/Layout/Draw 真明白了](https://juejin.cn/post/7016245187055878180)  
5、[Android事件分发全套服务](https://juejin.cn/post/7016233922828697608)  
6、[Android invalidate/postInvalidate/requestLayout 彻底厘清](https://juejin.cn/post/7017452765672636446)  
7、[Android Window 如何确定大小/onMeasure()多次执行原因](https://juejin.cn/post/7015980840047869983)  
8、[Android事件驱动Handler-Message-Looper解析](https://juejin.cn/post/7015237933120618504)  
9、[Android 键盘一招搞定](https://juejin.cn/post/7012844100994990087)  
10、[Android 各种坐标彻底明了](https://juejin.cn/post/7017834467175874591)  
11、[Android Activity/Window/View 的background](https://juejin.cn/post/7018044178709872677)  
12、[Android Activity创建到View的显示过](https://juejin.cn/post/7015959719739129869)  
13、[Android IPC 系列](https://juejin.cn/post/7023238726503383076)  
14、[Android 存储系列](https://juejin.cn/post/7012108220982362149)  
15、[Java 并发系列不再疑惑](https://juejin.cn/post/7010305230256488485)  
16、[Java 线程池系列](https://juejin.cn/post/7010622964781547527)  
17、[Android Jetpack 实践与原理系列](https://juejin.cn/post/7035235129479921671)

目录

* [前言](https://juejin.cn/post/7038064327814807589#heading-0)
* [1、什么是生命周期](https://juejin.cn/post/7038064327814807589#heading-1)
  + [Activity 的生命周期](https://juejin.cn/post/7038064327814807589#heading-2)
  + [Activity 生命周期不同阶段](https://juejin.cn/post/7038064327814807589#heading-3)
    - [Activity A 启动Activity B](https://juejin.cn/post/7038064327814807589#heading-4)
    - [从Activity A 按Home回到桌面](https://juejin.cn/post/7038064327814807589#heading-5)
    - [Activity B 按back 键/调用finish](https://juejin.cn/post/7038064327814807589#heading-6)
    - [Pause 阶段](https://juejin.cn/post/7038064327814807589#heading-7)
* [2、谁是生命周期的幕后黑手？](https://juejin.cn/post/7038064327814807589#heading-8)
  + [onCreate 调用堆栈](https://juejin.cn/post/7038064327814807589#heading-9)
  + [其它阶段的调用栈](https://juejin.cn/post/7038064327814807589#heading-10)
* [3、如何感知生命周期](https://juejin.cn/post/7038064327814807589#heading-11)
  + [注册监听](https://juejin.cn/post/7038064327814807589#heading-12)
  + [生命周期回调原理](https://juejin.cn/post/7038064327814807589#heading-13)
* [您若喜欢，请点赞、关注，您的鼓励是我前进的动力](https://juejin.cn/post/7038064327814807589#heading-14)
* [持续更新中，和我一起步步为营系统、深入学习Android](https://juejin.cn/post/7038064327814807589#heading-15)

下一篇

[Android Activity 与View 的互动思考](https://juejin.cn/post/7035235129479921671)