[Activity、Window、View三者关系](https://segmentfault.com/a/1190000019327597)

目录介绍

* 01.Window，View，子Window
* 02.什么是Activity
* 03.什么是Window
* 04.什么是DecorView
* 05.什么是View
* 06.关系结构图
* 07.Window创建过程
* 08.创建机制分析
  + 8.1 Activity实例的创建
  + 8.2 Activity中Window的创建
  + 8.3 DecorView的创建

弹窗系列博客

* [01.Activity、Window、View三者关系](https://link.segmentfault.com/?enc=Au7V2K7lhTS%2BjetadtLeqg%3D%3D.CBrT8x1BpxxOHZPP3rEfZQZXczQnsAmwLEu54AZI705lGAZ%2FM4FJN%2FYwXoP50e5EsM%2B0MBpDN2v2pC1wIlluUMQOxCzPm1ecDjRFWgrtztIizrVlM9FoPjznd2zj3EK%2FVpZP5O%2F%2FoDCvzr3Q6Qc9%2FWPK7bZmE0ib%2FiNTIjaCHljZMmqkp98Vc4gQR5xwJvC0WGqLGiW2rKY6Ztio9n71rg%3D%3D)
  + 深入分析Activity、Window、View三者之间的关系
* [02.Toast源码深度分析](https://link.segmentfault.com/?enc=gtdY7Vrc5F0DIUU379Dd%2Bw%3D%3D.0cIYoEYGF0n2WbAXnIhiIHw%2BO9inbLtdppP5Xld2a0O0ZKV2ocT08XqYEpsP%2BgNuxPmrsqHNuR1v4WmAaHTQYmvUIJ0WusllKKdOW%2FPCCCtkIdDjMJ0BSdDui7RZ4iEY8bnIEzrfIydLKTSl1%2BtjR%2FHcn4A9YRd7GiefonkMhpmh%2FNhP32TLFlZIWRXqEzlS)
  + 最简单的创建，简单改造避免重复创建，show()方法源码分析，scheduleTimeoutLocked吐司如何自动销毁的，TN类中的消息机制是如何执行的，普通应用的Toast显示数量是有限制的，用代码解释为何Activity销毁后Toast仍会显示，Toast偶尔报错Unable to add window是如何产生的，Toast运行在子线程问题，Toast如何添加系统窗口的权限等等
* [03.DialogFragment源码分析](https://link.segmentfault.com/?enc=5qa%2FtWntV%2FL9tqsSKDqX2Q%3D%3D.p8FqaDTwMOAgkEQFlMqAkwyftyZcM4xPmZbP2e26ROksIAg3Nmn4p%2Bfn0YbZcrwPsZxsP45OWYcv9WlNJK4PFfh7Zcspma%2BfDA1Nxf55SIWIJ7YoEzrSR3FGSxmwQ5%2B705Y%2FHpwVZJHQqHqGiu4a33%2B4gaJjn9dTgqMjJmQxcKU%3D)
  + 最简单的使用方法，onCreate(@Nullable Bundle savedInstanceState)源码分析，重点分析弹窗展示和销毁源码，使用中show()方法遇到的IllegalStateException分析
* [04.Dialog源码分析](https://link.segmentfault.com/?enc=pNUjtEoPUhm5%2F4zje9%2FI2Q%3D%3D.q6PrvjatDSyBXcOrMaR2W27CJ3asT2A6YSZCd%2BfbssTb0ZuwUP8KnTrgIqyRFuv594CBRi1Hgb86T2WnzxpnI3RYvo2%2FhSj8IzkgnPbuRHr16Xa9OgxGDfoOfDVIxjZ46LLqT53AVNeMDK2RUZBndYPIOznKfuVgGWZbsVmP6rU%3D)
  + AlertDialog源码分析，通过AlertDialog.Builder对象设置属性，Dialog生命周期，Dialog中show方法展示弹窗分析，Dialog的dismiss销毁弹窗，Dialog弹窗问题分析等等
* [05.PopupWindow源码分析](https://link.segmentfault.com/?enc=u5B04c%2ByYGMs3D%2FxY2kuGQ%3D%3D.h%2FVlPcpJ1aKLK1lTSc%2BwnCgfw1Xpql96SZtj1a4T5oC4%2FPSWUQk4u2Wus3ElO7Ug%2BVZ4Js9cagiC79Rhtk5vVyuVcJ17Q3OCetL%2BtK5RREVaqSkF1egx4zmTXW1%2BJ5y5%2B0F8rRActHH%2F79gpj4ofbyI75em7Oo6%2BXmjIIoZvdWo%3D)
  + 显示PopupWindow，注意问题宽和高属性，showAsDropDown()源码，dismiss()源码分析，PopupWindow和Dialog有什么区别？为何弹窗点击一下就dismiss呢？
* [06.Snackbar源码分析](https://link.segmentfault.com/?enc=CN0k2KfxnRu7jGNl9v%2Fk1w%3D%3D.9zRVjTh87Fsdsuhd693LxhOhDoGb1E6rxrRp7V4AWXCxsTAZDQ1lJ5y6xFQY3QOgqmmr8EbsJFRwAyBeEGr4UXhfJSY7STkosTWPKK7SYePE%2BP16y8j1aWny0zEcvuOzf%2FOqKDTg1rWA5sNo4YMYc%2BsQB0xkfW1dFT9ITDKK6lY%3D)
  + 最简单的创建，Snackbar的make方法源码分析，Snackbar的show显示与点击消失源码分析，显示和隐藏中动画源码分析，Snackbar的设计思路，为什么Snackbar总是显示在最下面
* [07.弹窗常见问题](https://link.segmentfault.com/?enc=774dI2jVKoR%2BZuCzC28y%2Fg%3D%3D.NmZd1GJ0WAvKjcKUR3xFaca5zDko%2BMNrExaOvFFlFMMzmPCZqDkjeWD%2B7bqJkImq3OEHlVPDR5sXcCEx6t96gLPWkJnLYtI%2BKQDq05iryMFKI0uV4ZDMWfshoXXt01U%2F%2FMoj4%2BPDp0ABoIKy7lbUp%2FCuiFSjSmBj%2BkzBLbrNjrc%3D)
  + DialogFragment使用中show()方法遇到的IllegalStateException,什么常见产生的？Toast偶尔报错Unable to add window，Toast运行在子线程导致崩溃如何解决？
* [09.onAttachedToWindow和onDetachedFromWindow](https://link.segmentfault.com/?enc=6Z4TxzhmIMEfvnmYyh3LYA%3D%3D.bptuC%2Bwbd3ROZkf%2Fnt0jci6BtJU769s3RhWbhvAFqjNawcJMUvkysUonkvu8S3R3bUNqqQsKSLEsQHkRDj5yYJZocxfEluasOQKo3SJQFjdQGTOB9myxcCTEd4w5eFLSrqqpRV7jkHvMQD5K3SR5CTzjdiRfRTIPZcg2mOjUeLM%3D)
  + onAttachedToWindow的调用过程，onDetachedFromWindow可以做什么？
* [10.DecorView介绍](https://link.segmentfault.com/?enc=fkrTypA12AlmfRD93kD8Jw%3D%3D.3hnTXnXwq7P%2FD4lp5PWHI%2FjcWHtlOoCL5hEgy7REDqin45Brk894opxKy4ldUQxe6rI981mBSMmt6HCCdE%2BWisfL9q25okIpR4j0AW7w6yhTlbU8Svo5gwXDYxBYRc4ItOg3MZgBbnY4DQSOOIUfIg%3D%3D)
  + 什么是DecorView，DecorView的创建，DecorView的显示，深度解析

01.Window，View，子Window

* 弹窗有哪些类型
  + 使用子窗口：在 Android 进程内，我们可以直接使用类型为子窗口类型的窗口。在 Android 代码中的直接应用是 PopupWindow 或者是 Dialog 。这当然可以，不过这种窗口依赖于它的宿主窗口，它可用的条件是你的宿主窗口可用
  + 采用View系统：使用 View 系统去模拟一个窗口行为，且能更加快速的实现动画效果，比如SnackBar 就是采用这套方案
  + 使用系统窗口：比如吐司Toast

02.什么是Activity

* Activity并不负责视图控制，它只是控制生命周期和处理事件。真正控制视图的是Window。一个Activity包含了一个Window，Window才是真正代表一个窗口。
* **Activity就像一个控制器，统筹视图的添加与显示，以及通过其他回调方法，来与Window、以及View进行交互。**

03.什么是Window

* Window是什么？
  + 表示一个窗口的概念，是所有View的直接管理者，任何视图都通过Window呈现(点击事件由Window->DecorView->View; Activity的setContentView底层通过Window完成)
  + Window是一个抽象类，具体实现是PhoneWindow。PhoneWindow中有个内部类DecorView，通过创建DecorView来加载Activity中设置的布局R.layout.activity\_main。
  + 创建Window需要通过WindowManager创建，通过WindowManager将DecorView加载其中，并将DecorView交给ViewRoot，进行视图绘制以及其他交互。
  + WindowManager是外界访问Window的入口
  + Window具体实现位于WindowManagerService中
  + WindowManager和WindowManagerService的交互是通过IPC完成
* 如何通过WindowManager添加Window(代码实现)？
  + 如下所示

|  |
| --- |
| * //1. 控件 * Button button = new Button(this); * button.setText("Window Button"); * //2. 布局参数 * WindowManager.LayoutParams layoutParams = new WindowManager.LayoutParams(WindowManager.LayoutParams.WRAP\_CONTENT, WindowManager.LayoutParams.WRAP\_CONTENT, 0, 0, PixelFormat.TRANSPARENT); * layoutParams.flags = WindowManager.LayoutParams.FLAG\_NOT\_TOUCH\_MODAL | WindowManager.LayoutParams.FLAG\_NOT\_FOCUSABLE | WindowManager.LayoutParams.FLAG\_SHOW\_WHEN\_LOCKED; * layoutParams.gravity = Gravity.LEFT | Gravity.TOP; * layoutParams.x = 100; * layoutParams.y = 300; * // 必须要有type不然会异常: the specified window type 0 is not valid * layoutParams.type = WindowManager.LayoutParams.TYPE\_SYSTEM\_ERROR; * //3. 获取WindowManager并添加控件到Window中 * WindowManager windowManager = getWindowManager(); * windowManager.addView(button, layoutParams); |

WindowManager的主要功能是什么？

* 添加、更新、删除View

|  |
| --- |
| * public interface ViewManager{ * public void addView(View view, ViewGroup.LayoutParams params); * //添加View * public void updateViewLayout(View view, ViewGroup.LayoutParams params); * //更新View * public void removeView(View view); * //删除View * } |

04.什么是DecorView

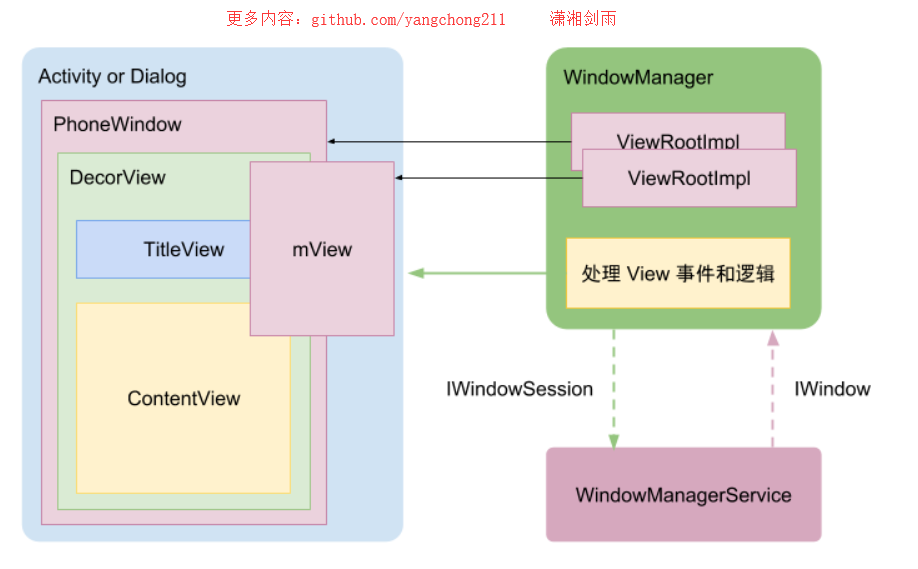
* DecorView是FrameLayout的子类，它可以被认为是Android视图树的根节点视图。
  + DecorView作为顶级View，一般情况下它内部包含一个竖直方向的LinearLayout，**在这个LinearLayout里面有上下三个部分，上面是个ViewStub，延迟加载的视图（应该是设置ActionBar，根据Theme设置），中间的是标题栏(根据Theme设置，有的布局没有)，下面的是内容栏。**
  + 具体情况和Android版本及主体有关，以其中一个布局为例，如下所示：

|  |
| --- |
| * <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" * android:fitsSystemWindows="true" * android:orientation="vertical"> * <!-- Popout bar for action modes --> * <ViewStub * android:id="@+id/action\_mode\_bar\_stub" * android:layout\_width="match\_parent" * android:layout\_height="wrap\_content" * android:inflatedId="@+id/action\_mode\_bar" * android:layout="@layout/action\_mode\_bar" * android:theme="?attr/actionBarTheme" /> * <FrameLayout * style="?android:attr/windowTitleBackgroundStyle" * android:layout\_width="match\_parent" * android:layout\_height="?android:attr/windowTitleSize"> * <TextView * android:id="@android:id/title" * style="?android:attr/windowTitleStyle" * android:layout\_width="match\_parent" * android:layout\_height="match\_parent" * android:background="@null" * android:fadingEdge="horizontal" * android:gravity="center\_vertical" /> * </FrameLayout> * <FrameLayout * android:id="@android:id/content" * android:layout\_width="match\_parent" * android:layout\_height="0dip" * android:layout\_weight="1" * android:foreground="?android:attr/windowContentOverlay" * android:foregroundGravity="fill\_horizontal|top" /> * </LinearLayout> |

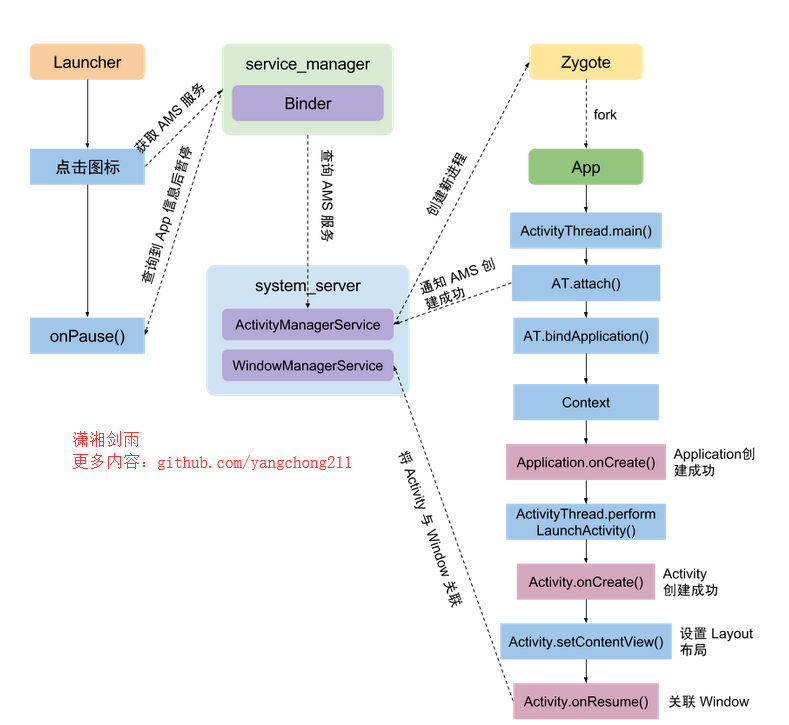
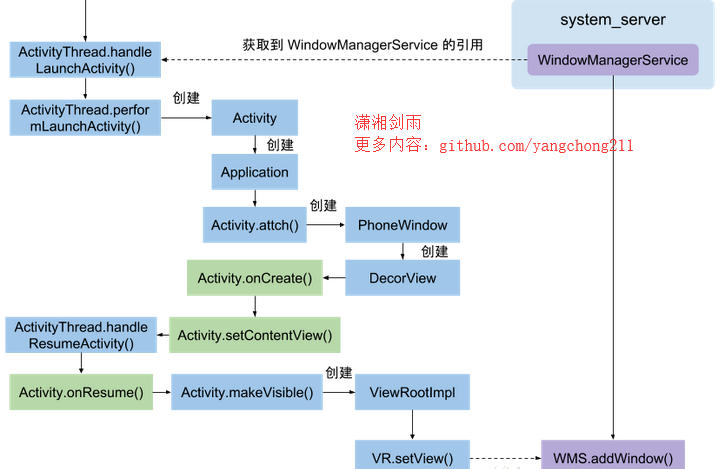
* 在Activity中通过setContentView所设置的布局文件其实就是被加到内容栏之中的，成为其唯一子View，就是上面的id为content的FrameLayout中，在代码中可以通过content来得到对应加载的布局。
* ViewGroup content = (ViewGroup)findViewById(android.R.id.content);

ViewGroup rootView = (ViewGroup) content.getChildAt(0);

06.关系结构图

* Activity 与 PhoneWindow 与 DecorView 关系图
  + 

07.Window创建过程

* App点击桌面图片启动过程
  + 
* window启动流程
  + 
* Activity 与 PhoneWindow 与 DecorView 之间什么关系？
  + 一个 Activity 对应一个 Window 也就是 PhoneWindow，一个 PhoneWindow 持有一个 DecorView 的实例，DecorView 本身是一个 FrameLayout。

08.创建机制分析

8.1 Activity实例的创建

* ActivityThread中执行performLaunchActivity，从而生成了Activity的实例。源码如下所示，ActivityThread类中源码

|  |
| --- |
| * private Activity performLaunchActivity(ActivityClientRecord r, Intent customIntent) { * ... * Activity activity = null; * try { * java.lang.ClassLoader cl = r.packageInfo.getClassLoader(); * activity = mInstrumentation.newActivity( * cl, component.getClassName(), r.intent); * ... * } catch (Exception e) { * ... * } * try { * ... * if (activity != null) { * ... * activity.attach(appContext, this, getInstrumentation(), r.token, * r.ident, app, r.intent, r.activityInfo, title, r.parent, * r.embeddedID, r.lastNonConfigurationInstances, config, * r.referrer, r.voiceInteractor); * ... * } * ... * } catch (SuperNotCalledException e) { * throw e; * } catch (Exception e) { * ... * } * return activity; * } |

8.2 Activity中Window的创建

* 从上面的performLaunchActivity可以看出，在创建Activity实例的同时，会调用Activity的内部方法attach
* 在attach该方法中完成window的初始化。源码如下所示，Activity类中源码

|  |
| --- |
| * final void attach(Context context, ActivityThread aThread, * Instrumentation instr, IBinder token, int ident, * Application application, Intent intent, ActivityInfo info, * CharSequence title, Activity parent, String id, * NonConfigurationInstances lastNonConfigurationInstances, * Configuration config, String referrer, IVoiceInteractor voiceInteractor, * Window window, ActivityConfigCallback activityConfigCallback) { * mWindow = new PhoneWindow(this, window, activityConfigCallback); * mWindow.setWindowControllerCallback(this); * mWindow.setCallback(this); * mWindow.setOnWindowDismissedCallback(this); * mWindow.getLayoutInflater().setPrivateFactory(this); * if (info.softInputMode != WindowManager.LayoutParams.SOFT\_INPUT\_STATE\_UNSPECIFIED) { * mWindow.setSoftInputMode(info.softInputMode); * } * if (info.uiOptions != 0) { * mWindow.setUiOptions(info.uiOptions); * } * } |

8.3 DecorView的创建

* 用户执行Activity的setContentView方法，内部是调用PhoneWindow的setContentView方法，在PhoneWindow中完成DecorView的创建。流程
  + 1.Activity中的setContentView
  + 2.PhoneWindow中的setContentView
  + 3.PhoneWindow中的installDecor

|  |
| --- |
| * public void setContentView(@LayoutRes int layoutResID) { * getWindow().setContentView(layoutResID); * initWindowDecorActionBar(); * } * @Override * public void setContentView(int layoutResID) { * ... * if (mContentParent == null) { * installDecor(); * } else if (!hasFeature(FEATURE\_CONTENT\_TRANSITIONS)) { * mContentParent.removeAllViews(); * } * ... * } * private void installDecor() { * if (mDecor == null) { * mDecor = generateDecor(); * mDecor.setDescendantFocusability(ViewGroup.FOCUS\_AFTER\_DESCENDANTS); * mDecor.setIsRootNamespace(true); * if (!mInvalidatePanelMenuPosted && mInvalidatePanelMenuFeatures != 0) { * mDecor.postOnAnimation(mInvalidatePanelMenuRunnable); * } * } * ... * } |

02.关于我的博客

* github：[https://github.com/yangchong211](https://link.segmentfault.com/?enc=n9ed3SD0ht01xTkvi6xzYQ%3D%3D.0Tcjp4qWQm2GObKOrvUyAR4iBXRO033%2FP49E8IFLW0A%3D)
* 知乎：[https://www.zhihu.com/people/...](https://link.segmentfault.com/?enc=aWCbQNjGuod6bvnE8mclOA%3D%3D.3EZeFWtA%2BhbcxUFgv%2Fv26rpCidr3LtGWrQiNHEjgoQ6FDEY5fAZV5HjpjCh%2FgA7d)
* 简书：[http://www.jianshu.com/u/b7b2...](https://link.segmentfault.com/?enc=y9honGjxw7k%2FS%2BKMT1zpOA%3D%3D.NHg8AtM3%2FuoYqyn4O4HK4w2R%2FbhexSiZAK%2BszS1skzfArl%2BT3K2lKHWziNWgrhDX)
* csdn：[http://my.csdn.net/m0\_37700275](https://link.segmentfault.com/?enc=XVomDp6mM%2B1Q7ghiRaTqKw%3D%3D.QiBDlMktueTVmufx9Q0mZDRXTn4oiPQes2Oaduxq7mk%3D)
* 喜马拉雅听书：[http://www.ximalaya.com/zhubo...](https://link.segmentfault.com/?enc=koCz9lb1sL8ZE1qr5YJLwg%3D%3D.yCU6x7bcA0hvYr801fH0Rt6wEjWAOPtolufT54dFQTHRmLvWL98SNTCqqf%2Bzq780)
* 开源中国：[https://my.oschina.net/zbj161...](https://link.segmentfault.com/?enc=ppHP6XgHsANPe3NFKkdM%2Bw%3D%3D.rsmD23ABUzq5dARa4XoH6EU34XtctqW34UvYM3ZXO7uzB%2B7ivXqPH3C92PVT%2F2r0)
* 泡在网上的日子：[http://www.jcodecraeer.com/me...](https://link.segmentfault.com/?enc=gmkQqAbiRMgH5%2FnoYzSMkw%3D%3D.L3%2Fgi%2Bt3k6CjhMKKx%2F7aHB25baI9fEfr9kEjlUPGNV0lDUOXA%2BG%2BaUEZhL0ZfkvVDhdlZOnk0F0voTmHqbw3GQ%3D%3D)
* 邮箱：yangchong211@163.com
* 阿里云博客：[https://yq.aliyun.com/users/a...](https://link.segmentfault.com/?enc=N10NBy06C4Wa%2BEyUbulWlQ%3D%3D.y24QVSVFlv5LbXnGVfFkjHV1a5mDJqwnWxGGdLbZDHHxU8%2FeDjdk3bIIOaX9dIilU3297i3mL0BNIKSdQFuPRQ%3D%3D) 239.headeruserinfo.3.dT4bcV
* segmentfault头条：[https://segmentfault.com/u/xi...](https://segmentfault.com/u/xiangjianyu/articles)
* 掘金：[https://juejin.im/user/593943...](https://link.segmentfault.com/?enc=WR8KFHBUDtxZdLOEWGs4Vg%3D%3D.QkVrSFQoKUE2U3csWFlmqjdxyhZUUrfViBW6NuEzSlNJS%2B3hTcGBySe8XlVQJ%2Fxu)

GitHub链接：[https://github.com/yangchong211](https://link.segmentfault.com/?enc=ePvsy0W%2Fr76rZXLWcJBkYg%3D%3D.cWUsXX1URS2Jq8KgR0lRNQytOkCBth85DqfGPCtmhmM%3D)

[[https://avatar-static.segmentfault.com/868/271/868271510-54cb382abb7a1_small](https://segmentfault.com/t/java)java](https://segmentfault.com/t/java)