Window和WindowManager

概念：

窗口，日常开发中直接接触Window的机会并不多，但是类似于悬浮窗的需要它

Window是一个抽象类，它的具体实现是 PhoneWindow

Android 中的所有视图都是通过Window来呈现的（Activity、Dialog、Toast……）

创建：

通过WindowManager来完成，WindowManager是外界访问Window的入口

8.1 Window和WindowManager

WindowManager.LayoutParams.Flag参数表示Window的属性，它有很多选项，用来控制Window的显示特性

|  |  |
| --- | --- |
| FLAG\_NOT\_FOCUSABLE | 表示Window不需要获取焦点，也不需要接收各种输入事件，此标记会同时启动 NOT\_TOUCH\_MODAL 标记。  最终事件会直接传递给下层具有焦点的Window |
| FLAG\_NOT\_TOUCH\_MODAL | 系统会将当前Window区域以外的单击事件传递给底层的Window，当前Window区域以内的单击事件则自己处理  即该Window不会拦截屏幕上所有的点击事件，它只处理和它有关的部分，API23是个点注意（好好体会一下） |
| FLAG\_SHOW\_WHEN\_LOCKED | 开启此模式可以让Window显示在锁屏的界面上 |

Type表示Window的类型

应用Window ： Activity （Z-ordered 1-99）

子Window ：如Dialog ( Z-ordered 1000~1999)

系统Window： 需要申明权限才能创建的，如Toast和状态栏 (Z-ordered 2000~2999)

TYPE\_SYSTEM\_OVERLAY | TYPE\_SYSTEM\_ERROR

注意：系统类型的 Window 是需要权限申请的，在注册文件中申请

Window是分层的，每个Window都有对应的 z-ordered,大的覆盖在小的上面

8.2： WindowManager 的用法

常用的只有三个方法： addView , updateView , removeView

拖动效果很简单，设置layoutParams里面的x，y值即可

Window的内部机制

Window是一个抽象的概念，每一个Window都对应一个View和一个ViewRootImpl

这点从WindowManager也可以看出，它定义的三个方法 addView、updateViewLayout以及removeView都是针对View的，这说明View才是Window存在的实体

在实际中无法直接访问 Window，必须通过WindowManager（操控的还是View）

SourceCode

为了分析Window的内部机制，这里从Window的添加、删除以及更新说起

8.2.1 Window 的添加过程

WindowManager的addView来实现，即添加Window但是Window是虚的，所以只能添加它的存在形式View

它的真正实现形式是WindowManagerImpl类