窗口管理是android的一个核心内容。它管理着窗口的创建和销毁，布局和大小，焦点的控制等等。

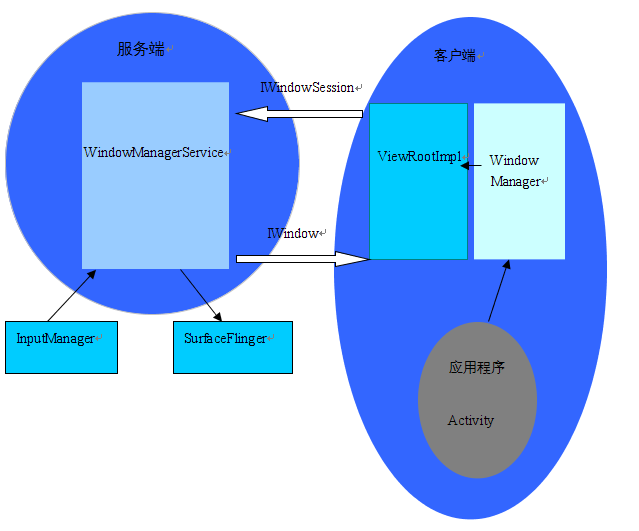
窗口可以分为两类：

一种是应用窗口，即由具体应用创建的窗口，其实其中还可以细分出父窗口和子窗口。窗口一般都会对应一个activity。

一种是系统窗口，如状态栏，这类窗口由系统直接通过windowManager来创建，和activity无关。

在这里，窗口的概念其实可以说由三部分构成，**一部分是用来描述窗口信息的**，由WindowState对象表示。**一个WindowState对象对应一个窗口**，它拥有绘制窗口所需要的信息。但是真正去绘制窗口需要另一部分内容Surface来完成，最终会通过surfaceflinger完成绘图。还有**一部分就是对消息的处理**，windowmanagerService把窗口信息传递给InputManager，这样InputDispatcher就能根据当前窗口的状态进行消息处理。

我们先看下整体的架构图，然后再来看这两种窗口的创建。WindowManager和其他很多android的服务一样，采用C/S的架构。其中windowManagerService跑在System\_server进程，作为服务端，客户端通过ipc调用和它进行交互。



我们通过完整的应用程序窗口创建流程来了解这个结构和整个过程。我们不去纠结其中代码的一些细枝末节的东西，通过整体和重要的东西来看。

一.客户端部分

在客户端，在应用启动的时候，ActivityThread会调用performLaunchActivity方法，去实例化一个activity，同时调用attach方法，并传递很多和activity相关的参数信息。其中有个比较重要的东西是**一个IBinder对象token，这个token成为activity的标识**，windowmanagerService可以**通过这个token获得activity当前的运行状态**。在WindowManager中会通过该token生成一个WindowToken对象，一个父窗口对应一个WindowToken，而具有相同token的所有其子窗口都会被归到一个WindowToken中。即如果token相同，表示他们都会在一个窗口中。还有个用来标识窗口的类AppWindowToken，继承自WindowToken，它由activity传过来的token生成，和Activity一一对应。通过token，就能找到activity和window的对应关系了。

继续往下看，attach方法会通过代码mWindow = PolicyManager.makeNewWindow(this)实例化一个phoneWindow对象，但是这个对象还是比较抽象的东西。在activity开始oncreate调用时，会调用setContentView方法。会去获得之前那个phoneWIndow对象对应的DecorView，最后通过层层窗口修饰（状态栏等）后调用activity的makeVisible方法，在方法中通过addiew方法完成窗口的添加。

windowManager只是提供接口，用了桥接模式，真正实现是WindowManagerImpl类。而调用addiew方法的对象来自另一个类LocalWindowManager，它会做一些简单检查，再通过WindowManagerImp类的addview完成窗口添加。addview大概分三步执行：

1.校验该窗口是否已经添加过了。

2.判断窗口类型如果是子窗口，则找到它附属的父窗口

3.new一个ViewRootImpl对象，最后调用该对象的setView方法。

setView 方法会最终会通过ipc调用IwindowSession的add方法。Session类实现了该方法，并最终给WindowManagerService处理。**客户端的工作至此就完成了**。

这里说明一下ViewRootImpl类，这其实是个handler。自然的，它一部分功能就是对消息进行处理，将用户的一些操作分发到view中。它也是view和WindowManagerService的桥梁。可以看到它通过一个会话将信息传递到了WindowManagerService。而WIndowManagerService也会通过IWindow接口将指令通过消息的方式发送到ViewRootImpl，ViewRootImpl处理这些消息。

二.服务端

WindowManagerService的addWindow方法主要做三部分的处理。

1.做一些合法性校验

2.完成窗口数据的构建

3.完成窗口创建后需要作出的一些调整

我们只看第二部分。首先会new一个WindowState类，该类表示一个窗口。结合WindowToken和AppWindowToken，完整的定义了一个窗口内容。接着创建一个管道，用于处理消息输入。再然后调用attach方法，创建和Surface相关的内容，用于和surfaceFlinger交互。这样，整个窗口就搭建完成了。有了WindowState类对窗口属性的保存以及token对窗口归属的标识，之后就可以通过SurfaceFlinger绘制在屏幕上了。之后通过InputManager，也能处理消息和WindowManagerService之间的传递。保证窗口显示内容和用户操作保持一致性。

当然，WindowManagerService靠近10000行的代码完成了很多功能，因为这篇文章只会了解窗口管理的整个架构，这里不一一详解，以后有时间可能会把一些比较有意思的内容再看下：

1.      窗口的创建和删除

2.      窗口的显示和隐藏控制

3.      Z-order顺序管理

4.      焦点窗口管理

5.      输入法窗口管理和墙纸窗口管理

6.      切换动画

7.      系统消息收集和分发

现在，再来看开始的架构图，应该就比较清晰了。