

**一、Surface、SurfaceView、SurfaceHolder的关系以及概念**

SurfaceView就是一个继承View的组件，这个视图中内嵌了一个专门用于绘制的Surface。用于显示Surface中的内容，而SurfaceHolder是用于控制（监听）Surface的变化。

它和其他的视图组件不同之处在于它拥有独立的绘图表面，即它不与其宿主共享同一个绘制表面，由于拥有独立的绘图表面，因此SurfaceView 的UI可以在一个独立的线程中进行绘制。又由于不会占用主线程的资源，SurfaceView一方面可以实现复制而又高效的UI，另一方面又不会导致用户输入得不到及时响应。

 Surface是纵深排序(Z-ordered)的，这表明它总在自己所在窗口的后面。SurfacevView提供了一个可见区域，只有在这个可见区域内的surface部分内容才可见，可见区域外的部分不可见。Surface的排版显示受到视图层级关系的影响，它的兄弟视图结点会在顶端显示。这意味者 Surface的内容会被它的兄弟视图遮挡，这一特性可以用来放置遮盖物(overlays)(例如，文本和按钮等控件)。

注意，如果Surface上面有透明控件，那么它的每次变化都会引起框架重新计算它和顶层控件的透明效果，这会影响性能。

简单的说Surface、SurfaceView、SurfaceHolder就是设计模式中的MVC模式，Surface就是Model，用于保存要用于显示的数据。SurfaceView就是View，用于显示Surface中的数据、以及surface在屏幕中显示的位置。SurfaceHolder就是Surface中数据内容的监听器，Surface的创建、销毁等。

Surfaceview变得可见时，Surface被创建；Surfaceview隐藏前，Surface被销毁。所以Surface的生命周期是在[surfaceCreated(SurfaceHolder)](http://android.xsoftlab.net/reference/android/view/SurfaceHolder.Callback.html" \l "surfaceCreated(android.view.SurfaceHolder))和[surfaceDestroyed(SurfaceHolder)](http://android.xsoftlab.net/reference/android/view/SurfaceHolder.Callback.html" \l "surfaceDestroyed(android.view.SurfaceHolder))之间，即SurfaceHolder.Callback中的创建Surface函数和销毁Surface函数成为了绘图处理（显示内容）处理代码的边界。

**二、整体实现过程**

SurfaceView的核心在于提供了两个线程，UI线程和渲染线程。

 1> 所有SurfaceView和SurfaceHolder.Callback的方法都应该在UI线程里调用，一般来说就是应用程序主线程。渲染线程所要访问的各种变量应该作同步处理。  
       2> 由于Surface可能被销毁，它只在SurfaceHolder.Callback.surfaceCreated()和 SurfaceHolder.Callback.surfaceDestroyed()之间有效，所以要确保渲染线程访问的是合法有效的Surface。

整个实现流程：

（1）继承SurfaceView并实现SurfaceHolder.Callback接口

（2）通过SurfaceView.getHolder()获得SurfaceHolder对象

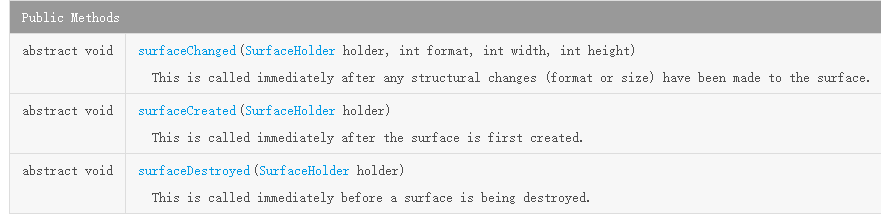
（3）SurfaceHolder.addCallback([SurfaceHolder.Callback](http://android.xsoftlab.net/reference/android/view/SurfaceHolder.Callback.html))添加回调函数

（4）SurfaceHolder.lockCanvas()获取Canvas并锁定画布

（5）Canvas绘画（显示的内容）

（6）SurfaceHolder.unlockCanvasAndPost(Canvas)结束锁定画图并提交图形显示

1、SurfaceHolder.Callback主要是当底层的Surface被创建、销毁或者改变时，提供回调通知，绘制必须要在Surface创建之后才能进行。实现该接口需要重写3个方法



1）SurfaceChanged(…)：当Surface的结构变化时调用

2）SurfaceCreate(SurfaecHolder)：当第一次创建Surface时调用

3）SurfaceDestroyed(SurfaceHolder)：当销毁Surface时调用

2、SurfaceHolder：可以把它看成是一个Surface的控制器，用来操纵Surface在Canvas上画的效果和动画、控制表面、大小等。

SurfaceHolder中的几个常用的函数：

1. [addCallback](http://android.xsoftlab.net/reference/android/view/SurfaceHolder.html#addCallback(android.view.SurfaceHolder.Callback))([SurfaceHolder.Callback](http://android.xsoftlab.net/reference/android/view/SurfaceHolder.Callback.html) callback)
2. [lockCanvas](http://android.xsoftlab.net/reference/android/view/SurfaceHolder.html#lockCanvas())()
3. [unlockCanvasAndPost](http://android.xsoftlab.net/reference/android/view/SurfaceHolder.html#unlockCanvasAndPost(android.graphics.Canvas))([Canvas](http://android.xsoftlab.net/reference/android/graphics/Canvas.html) canvas)

try { //执行的代码，其中可能有异常。一旦发现异常，则立即跳到catch执行。否则不会执行catch里面的内容 }   
  
catch { //除非try里面执行代码发生了异常，否则这里的代码不会执行 }   
  
finally { //不管什么情况都会执行，包括try catch 里面用了return ,可以理解为只要执行了try或者catch，就一定会执行 finally }