多媒体框架

<https://blog.csdn.net/column/details/12812.html>

# MediaProjection简介

MediaProjection是一个5.0之后给开发者提供的新的截屏或者录屏的新手段。MediaProjection可以用来捕捉屏幕，而MediaProjectionManager可以从命名上面来看到，它是是用来管理MediaProjection的。

MediaProjection是一个准许APP拥有截取屏幕或者是记录系统音频的能力。还有其他的能力依据MediaProjection的类型。通过MediaProjectionManager 类的 createScreenCaptureIntent()方法来开启，可以用来截取屏幕，但是不能够获取系统的音频。

## createVirtualDisplay

fg

公共方法1：

VirtualDisplay createVirtualDisplay (String name,

int width,

int height,

int dpi,

int flags,

Surface surface,

VirtualDisplay.Callback callback,

Handler handler)

创建一个屏幕的截图，保存在VirtualDisplay中，并返回。

参数1：实际的流媒体显示实体名字，不能为null;

参数2：实际的流媒体显示实体的宽度，单位为像素，必须大于0;

参数3：实际的流媒体显示实体的高度，单位为像素，必须大于0;

参数4：实际的流媒体显示实体的像素密度，单位为dp，必须大于0;

参数5：实际的流媒体显示实体标志的结合，更多请查看 DisplayManager里头的标志，取值是{VIRTUAL\_DISPLAY\_FLAG\_AUTO\_MIRROR，VIRTUAL\_DISPLAY\_FLAG\_OWN\_CONTENT\_ONLY，VIRTUAL\_DISPLAY\_FLAG\_PRESENTATION，

VIRTUAL\_DISPLAY\_FLAG\_PUBLIC，

VIRTUAL\_DISPLAY\_FLAG\_SECURE}中的一个

参数6：播放流媒体的surface实例，可为null，如果木有;

参数7：实际的流媒体显示实体状态改变时的回调方法，可能为null;

参数8：调用参数7回调方法的handler;

返回VirtualDisplay实例

if ((flags & DisplayManager.VIRTUAL\_DISPLAY\_FLAG\_SECURE) != 0) {

if (!canProjectSecureVideo(projection)) {

throw new SecurityException("Requires CAPTURE\_SECURE\_VIDEO\_OUTPUT "

+ "or an appropriate MediaProjection token to create a "

+ "secure virtual display.");

}

}

这个是在创建的时候制定的，和我们的目标不一样，不用处理了吧？？？

## registerCallback/unregisterCallback

此方法是注册监听流媒体显示实体状态改变时的回调方法。参数请参照createVirtualDisplay()方法的参数7参数8;

void registerCallback (MediaProjection.Callback callback,

Handler handler)

void unregisterCallback (MediaProjection.Callback callback)

顾名思义：这个方法是注销对流媒体状态改变的监听。

## stop

此方法是停止截屏的方法;

# OverlayDisplayWindow

# WindowManager.LayoutParams.FLAG\_SECURE

F

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ActivityThread.java | (core\java\android\app): | & | WindowManager.LayoutParams.FLAG\_SECURE) | == | 0)) | { |  |
| AssistStructure.java | (core\java\android\app\assist): | if | ((root.getWindowFlags()& | WindowManager.LayoutParams.FLAG\_SECURE) | != | 0) | { |
| ~~OverlayDisplayWi~~ndow.java | (services\core\java\com\android\server\display): | mWindowParams.flags | |= | WindowManager.LayoutParams.FLAG\_SECURE; |  |  |  |
| ~~SurfaceView.java~~ | (core\java\android\view): | mLayout.flags | |= | WindowManager.LayoutParams.FLAG\_SECURE; |  |  |  |
| ~~SurfaceView.java~~ | (core\java\android\view): | mLayout.flags | &= | ~WindowManager.LayoutParams.FLAG\_SECURE; |  |  |  |
| VirtualDisplayTest.java | (core\tests\coretests\src\android\hardware\display): | WindowManager.LayoutParams.FLAG\_SECURE); |  |  |  |  |  |
| VirtualDisplayTest.java | (core\tests\coretests\src\android\hardware\display): | WindowManager.LayoutParams.FLAG\_SECURE); |  |  |  |  |  |
| VirtualDisplayTest.java | (core\tests\coretests\src\android\hardware\display): | WindowManager.LayoutParams.FLAG\_SECURE); |  |  |  |  |  |
| VirtualDisplayTest.java | (core\tests\coretests\src\android\hardware\display): | WindowManager.LayoutParams.FLAG\_SECURE); |  |  |  |  |  |
| VoiceInteractionMain.java | (tests\voiceinteraction\src\com\android\test\voiceinteraction): | getWindow().addFlags(WindowManager.LayoutParams.FLAG\_SECURE); |  |  |  |  |  |
| VoiceInteractionMain.java | (tests\voiceinteraction\src\com\android\test\voiceinteraction): | getWindow().clearFlags(WindowManager.LayoutParams.FLAG\_SECURE); |  |  |  |  |  |
| WindowManagerService.java | (services\core\java\com\android\server\wm):import | static | android.view.WindowManager.LayoutParams.FLAG\_SECURE; |  |  |  |  |
| WindowManagerService.java | (services\core\java\com\android\server\wm): | if | ((w.mAttrs.flags&WindowManager.LayoutParams.FLAG\_SECURE) | != | 0) | { |  |
| WindowState.java | (services\core\java\com\android\server\wm):import | static | android.view.WindowManager.LayoutParams.FLAG\_SECURE; | shouldSaveSurface |  |  |  |