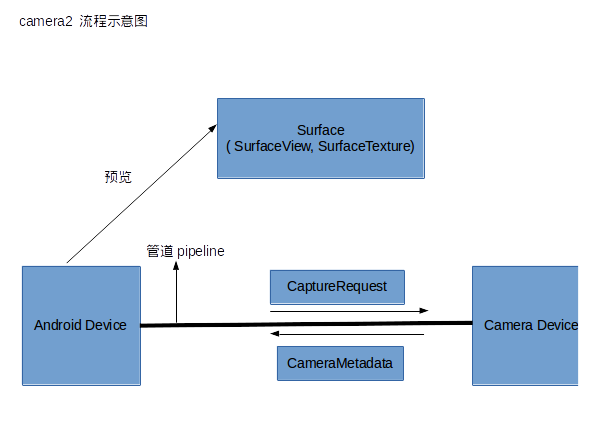
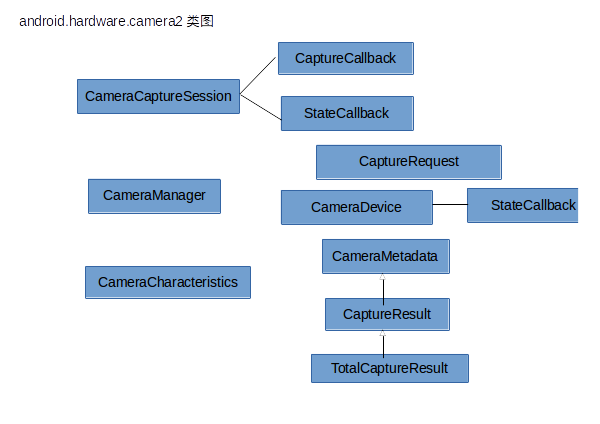
# 1框架



引用了管道的概念将安卓设备和摄像头之间联通起来，系统向摄像头发送 Capture 请求，而摄像头会返回 CameraMetadata。这一切建立在一个叫作 CameraCaptureSession 的会话中。

管道：一种进程通信，对两个文件进行操作，表示管道的两端，一个读，另一个写

# 2核心类介绍



## 2.1 CameraManager

站在高处统管所有摄像头设备，用于检测、打开系统摄像头，并连接CameraDevices

·Context.getSystemService(Context.CAMERA\_SERVICE) 来完成初始化

·通过getCameraIdList()方法获取Android设备的摄像头列表

·getCameraCharacteristics(String cameraId)获取摄像头的详细参数和支持的功能

·openCamera(String cameraId, CameraDevice.StateCallback callback, Handler handler)打开指定Id的摄像头

2.2 CameraDevice

摄像头设备，会负责建立 CameraCaptureSession 以及建立 CaptureRequest

·通过CameraDevice.StateCallback监听摄像头的状态（主要包括onOpened、onClosed、onDisconnected、onErro四种状态）

·管理CameraCaptureSession，-通过方法createCaptureSession()方法创建会话，通常会在CameraDevice.StateCallback中调用对应方法创建预览会话。

·管理CaptureRequest，主要包括通过createCaptureRequest（int templateType）创建捕获请求，在需要预览、拍照、再次预览的时候都需要通过创建请求来完成。

## **2.3 CameraCaptureSession**

系统向摄像头发送 Capture 请求，而摄像头会返回 CameraMetadata，这一切都是在由对应的CameraDevice创建的CameraCaptureSession 会话完成，当程序需要预览、拍照、再次预览时，都需要先通过会话。

CameraCaptureSession一旦被创建，直到对应的CameraDevice关闭才会死掉。虽然CameraCaptureSession会话用于从摄像头中捕获图像，但是只有同一个会话才能再次从同一摄像头中捕获图像。

创建会话是一项耗时的异步操作，可能需要几百毫秒，因为它需要配置相机设备的内部管道并分配内存缓冲区以将图像发送到所需的目标，因而createCaptureSession和createReprocessableCaptureSession会将随时可用的CameraCaptureSession发送到提供的监听器的onConfigured回调中。如果无法完成配置，则触发onConfigureFailed回调，并且会话将不会变为活动状态。 最后需要注意的是，如果摄像头设备创建了一个新的会话，那么上一个会话是被关闭的，并且会回调与其关联的onClosed，如果不处理好，当会话关闭之后再次调用会话的对应方法那么所有方法将会跑出IllegalStateException异常。关闭的会话清除任何重复的请求（和调用了stopRepeating()方法类似），但是在新创建的会话接管并重新配置摄像机设备之前，关闭的会话仍然会正常完成所有正在进行的捕获请求。

·管理CameraCaptureSession.StateCallback状态回调，用于接收有关CameraCaptureSession状态的更新的回调对象，主要回调方法有两个，当CameraDevice 完成配置，对应的会话开始处理捕获请求时触发onConfigured(CameraCaptureSession session)方法，反之配置失败时候触发onConfigureFailed(CameraCaptureSession session)方法。

·管理CameraCaptureSession.CaptureCallback捕获回调，用于接收捕获请求状态的回调，当请求触发捕获已启动时、捕获完成时、在捕获图像时发生错误的情况下都会触发该回调对应的方法。

·通过调用方法capture()提交捕获图像请求，即拍照。 其中该请求定义了捕获单个图像的所有参数，包括传感器，镜头，闪光灯和后处理参数，每一次请求的结果将产生一个CaptureResult，可以为一个或多个Surface生成新的帧，然后通过CaptureRequest.Builder的addTarget(Surface)方法附着到对应的Surface上显示，而且这个参数Surface必须是会话创建时候的一个子集，会话一次可以处理多个常规和重新处理请求。但如果只有常规请求或重新处理请求在进行，则以先进先出的顺序处理它们；如果两者都在进行中则分别以各自的先进先出顺序处理他们；然而，处理常规请求和重新处理请求的顺序并不是特定的，换言之，一个常规请求在下一个常规请求提交前被处理，同理重新处理请求也一样，但是一个常规请求不一定是在下一个重新处理请求提交之前被处理。通过capture方法提交的请求处理优先级比通过其他方式（setRepeatingRequest() 或者setRepeatingBurst()）提交的请求的处理优先级高，一旦当前的repeat / repeatBurst处理完成，就会被处理。最后一点，所有CaptureSession可用于从相机捕获图像，但只有由createReprocessableCaptureSession创建的会话才可以提交重新处理捕获请求，将重新处理请求提交到常规捕获会话将导致IllegalArgumentException。

·通过调用方法setRepeatingRequest()请求不断重复捕获图像，即实现预览

·通过方法调用stopRepeating()实现停止捕获图像，即停止预览。

## 2.4 CameraCharacteristics

是 CameraDevice 的属性描述类，可以通过getCameraCharacteristics（String cameraId）进行查询。

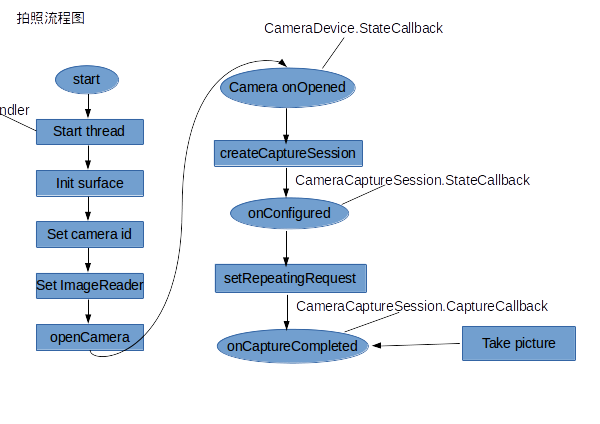
## 2.5 CameraRequest和CameraRequest.Builder

CameraRequest代表了一次捕获请求，而CameraRequest.Builder用于描述捕获图片的各种参数设置，包含捕获硬件（传感器，镜头，闪存），对焦模式、曝光模式，处理流水线，控制算法和输出缓冲区的配置。，然后传递到对应的会话中进行设置，CameraRequest.Builder则负责生成CameraRequest对象。当程序调用setRepeatingRequest()方法进行预览时，或调用capture()方法进行拍照时，都需要传入CameraRequest参数。CameraRequest可以通过CameraRequest.Builder来进行初始化，通过调用createCaptureRequest来获得。

## 2.6 CaptureResult

CaptureRequest描述是从图像传感器捕获单个图像的结果的子集的对象。（CaptureResults are produced by a CameraDevice after processing a CaptureRequest）当CaptureRequest被处理之后由CameraDevice生成

# 3拍照流程

1，获取TextureView，设置监听器 TextureView.SurfaceTextureListener，在该view准备好后回调onSurfaceTextureAvailable

1. 在onSurfaceTextureAvailable里打开摄像头。通过getSystemService获取CameraManager，接着获取camera Id， 获取打开摄像头的属性CameraCharacteristics，通过CameraManager.openCamera初始化摄像头，同时传入了CameraDevice.StateCallback
2. 创建一个ImageReader对象，用于获得摄像头的图像数据，设置监听器ImageReader.OnImageAvailableListener
3. 摄像头打开时回调onOpened，在这里设置预览大小等，接着创建CaptureRequest.Builder，传入ImageReader作为输出目标，这里可以自动曝光、照片方向等属性
4. 调用CameraDevic.createCaptureSession创建会话，成功则回调onConfigured
5. 在onConfigured中执行builder的setRepeatingRequest，正式开启预览
6. 预览完成后，接下来是拍照
7. 依旧是创建CaptureRequest.Builder设定好各参数，接着创建createCaptureSession会话，回调里面CameraCaptureSession.capture传入CaptureRequest.Builder.build()以及CameraCaptureSession.CaptureCallback
8. CameraCaptureSession.CaptureCallback会回调onCaptureCompleted，在这里能拿到CaptureRequest和TotalCaptureResult信息，可以在这里开启下一个预览
9. 同时，在ImageReader.OnImageAvailableListener的onImageAvailable就可以拿到图片数据了