28장. 카메라 활용

28.1 SurfaceView vs TextureView

28.1.1. SurfaceView

뷰의 내용을 화면에 출력하기 위해 Surface(LayerBuffer)를 활용하는 뷰
Surface에 그리는 작업은 GPU(Graphic Processing Units)로 빠르게 처리

- SurfaceHolder: Surface에 대한 작업자 클래스
- SurfaceHolder.Callback: Surface의 create, update, destory 상황에 호출될 콜백 함수를 가지는 인터페이스

Callback 클래스

- surfaceCreated: 최초로 Surface가 만들어진 순간에 호출
- surfaceChanged : Surface의 크기 변경이 발생하는 순간마다 호출
- surfaceDestoryed : 최종 Surface 공간이 소멸되는 순간 호출

28.1 SurfaceView vs TextureView

```
public class MySurfaceView extends SurfaceView implements SurfaceHolder.Callback {

    private SurfaceHolder holder;

    public MySurfaceView(Context context) {
        super(context);
        //SurfaceHolder Create
        holder = getHolder();
        //Surface callback 등록
        holder.addCallback(this);
    }

    public void surfaceChanged(SurfaceHolder holder, int format,
                               int width, int height) {        //...
    }

    public void surfaceCreated(SurfaceHolder holder) {
        //...
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        //...
    }
}
```

28.1 SurfaceView vs TextureView

SurfaceHolder 클래스
- addCallback(): Surface에 대한 Callback 클래스를 지정하기 위한 함수
- lockCanvas() : Surface에 그리기 작업을 하기 위한 Canvas를 반환
- unlockCanvasAndPost(): Surface에 그린 내용을 화면에 출력

```
Canvas c = null;
c = holder.lockCanvas(null);

//drawing 작업
holder.unlockCanvasAndPost(c);
```

28.1 SurfaceView vs TextureView

28.1.2. TextureView

SurfaceView는 뷰 자체에 애니메이션, 형태 바꾸기, 스케일 조정 등이 불가능
두 개의 SurfaceView를 겹쳐서 출력하는 것도 불가능
SurfaceView의 단점을 해결하기 위해 TextureView 제공

TextureView를 이용하면 일반 뷰처럼 제어 가능
애니메이션 효과를 적용할 수 있고 이동 및 크기를 조정 가능
TextureView는 SurfaceView보다 훨씬 더 많은 메모리를 소비
퍼퍼먼스상 SurfaceView가 조금 더 유리

SurfaceTextureListener 인터페이스
- onSurfaceTextureAvailable(): TextureView에서 SurfaceTexture를 사용할 준비가 되었을 때 호출
- onSurfaceTextureSizeChanged(): SurfaceTexture의 버퍼 크기가 바뀌었을 때 호출
- onSurfaceTextureDestroyed(): SurfaceTexture가 Destroy 됐을 때 호출
- onSurfaceTextureUpdated(): SurfaceTexture의 내용이 변경되었을 때 호출

28.1 SurfaceView vs TextureView

```
public class MyTextureView extends TextureView implements        TextureView.SurfaceTextureListener {
    public MyTextureView(Context context) {
        super(context);
        //callback 등록
        setSurfaceTextureListener(this);
    }
    //최초에 자동 호출..
    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture surface, int width, int height) {
    }

    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture surface, int width, int height) {
    }

    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {
        return false;
    }

    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture surface) {
        //...
    }

}
```

28.1 SurfaceView vs TextureView

- lockCanvas( ) 함수로 Canvas 객체를 얻고, 그린 내용을 unlockCanvasAndPost( ) 함수를 이용해 화면에 적용

```
Canvas c=lockCanvas(null);
//drawing 작업unlockCanvasAndPost(c);
```

28.2 Camera API

```
<uses-permission android:name="android.permission.CAMERA" />
```

- 카메라에서 넘어오는 영상을 화면에 출력하려면 SurfaceView나 TextureView가 필요

```
<TextureView
   android:id="@+id/lab1_textureview"
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="1"/>
```

28.2 Camera API

```
public class Lab1Activity extends AppCompatActivity implements TextureView.SurfaceTextureListener {
    TextureView textureView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //...
        textureView.setSurfaceTextureListener(this);
        }
    }
    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture surface, int width, int height) {

    }
    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture surface, int width, int height) {

    }
    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {
        return true;
    }
    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture surface) {

    }
}
```

28.2 Camera API

Camera 오픈
- open(): 하드웨어 카메라를 이용할 수 있게 점유
- release(): 카메라 작업이 끝난 후 Camera 객체 자원 반납

- 카메라 설정

```
Camera.Parameters parameters = camera.getParameters();
```

- Preview 크기 설정

```
supportedPreviewSizes = parameters.getSupportedPreviewSizes();
if (supportedPreviewSizes != null) {
    previewSize = CameraUtil.getOptimalPreviewSize(supportedPreviewSizes,                    width,
height);
    parameters.setPreviewSize(previewSize.width, previewSize.height);
}
```

- 영상의 방향 설정

```
int result=CameraUtil.setCameraDisplayOrientation(this, 0);
//사진 촬영 시 획득되는 데이터의 방향
parameters.setRotation(result);
//화면에 출력되는 형상의 방향
camera.setDisplayOrientation(result);
```

28.2 Camera API

- Parameters 값을 Camera에 대입

```
camera.setParameters(parameters);
```

- 카메라로부터 넘어오는 영상을 TextureView에 전달

```
@Override
public void onSurfaceTextureAvailable(SurfaceTexture surface, int width, int height) {
    try {
        camera.setPreviewTexture(surface);
    } catch (IOException t) {
    }
    camera.startPreview();
}
```

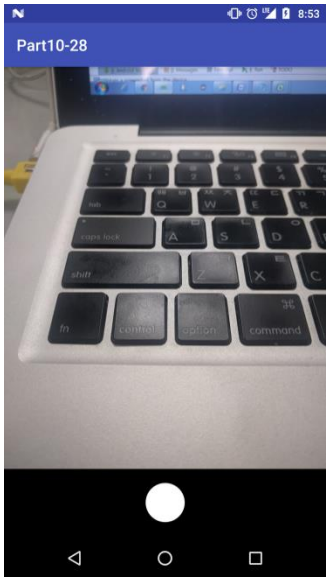
- Preview 시작은 startPreview( ), 그리고 종료는 stopPreview( ) 함수를 이용
- 사진 찍는 기능은 takePicture( ) 함수를 이용

```
camera.takePicture(null, null, new Camera.PictureCallback() {
    @Override
    public void onPictureTaken(byte[] data, Camera camera) {
        camera.startPreview();
    }
});
```

Step by Step 28-1 – Camera API

Camera API을 이용하여 화면에 카메라 영상을 출력하고 사진찍어 파일로 저장하는 테스트

1. Module 생성
2. 파일 복사
3. AndroidManifest.xml 작업
4. MainActivity.java 작성
5. 실행



28.3 Camera2 API

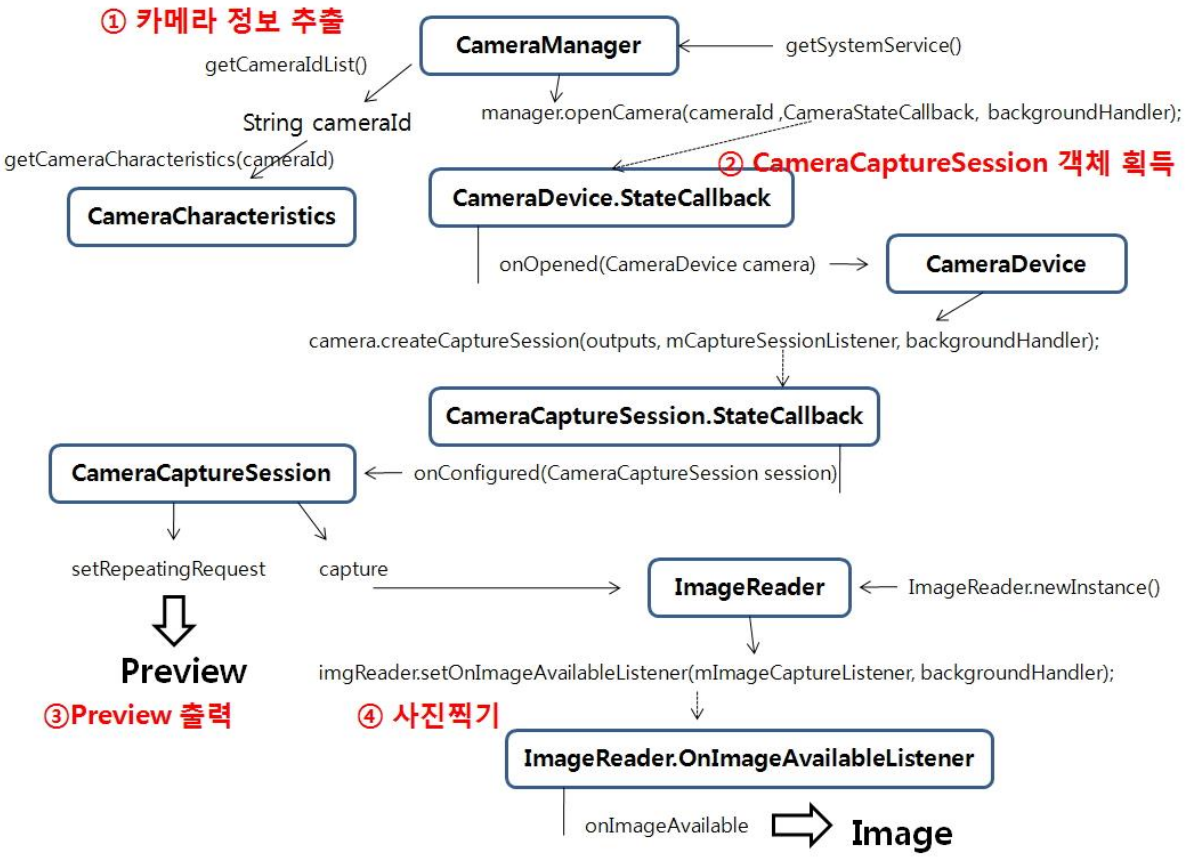
28.3.1. Camera2 API 구조

API Level 21 에서 Camera2 API를 제공
- 무압축 파일 촬영 가능
- 연사속도 향상
- 동영상 프레임 향상
- 동영상 촬영 시 사진촬영
- 셔터 딜레이 단축
- 수동 조작 가능

Camera2 API에서 주요 클래스, 인터페이스

• CameraManager: SystemService, CameraDevice 획득에 사용
• CameraDevice: 카메라 장치를 지칭, CameraCaptureSession create 명령
• CameraCaptureSession: preview 화면 출력 및 이미지 캡처 기능 제공
• ImageReader: 이미지 캡처를 위해 데이터 read 기능 제공
• CameraCharacteristics: 카메라의 각종 정보 획득
•CameraDevice.StateCallback: 카메라 open, disconnection 등의 콜백 제공
• CameraCaptureSession.StateCallback: 카메라 캡처 세션 콜백
• ImageReader.OnImageAvailableListener: 이미지 획득 시 콜백 함수 제공

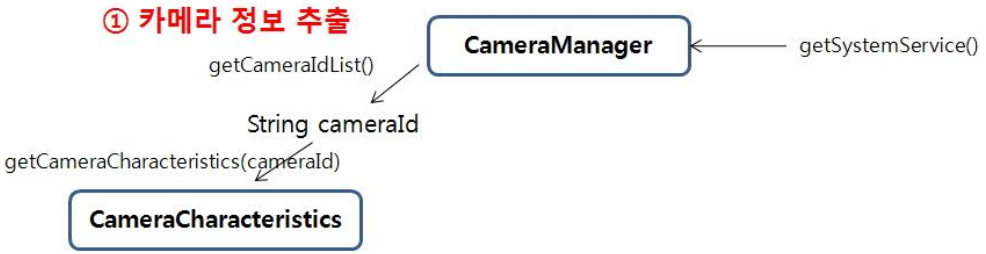
28.3 Camera2 API



28.3 Camera2 API

28.3.2. Camera2 API 활용

- 카메라 정보 추출




```
manager = (CameraManager)getSystemService(CAMERA_SERVICE);
```

- 카메라 식별자 획득

```
for (String cameraId : manager.getCameraIdList()) {
    characteristics = manager.getCameraCharacteristics(cameraId);    //...
}
```

- 정보 추출

```
characteristics.get(characteristics.LENS_FACING)
```

- LENS_FACING_FRONT: 전면 카메라. value : 0
- LENS_FACING_BACK: 후면 카메라. value : 1
- LENS_FACING_EXTERNAL: 기타 카메라. value : 2

28.3 Camera2 API

- 카메라에서 지원하는 크기 목록

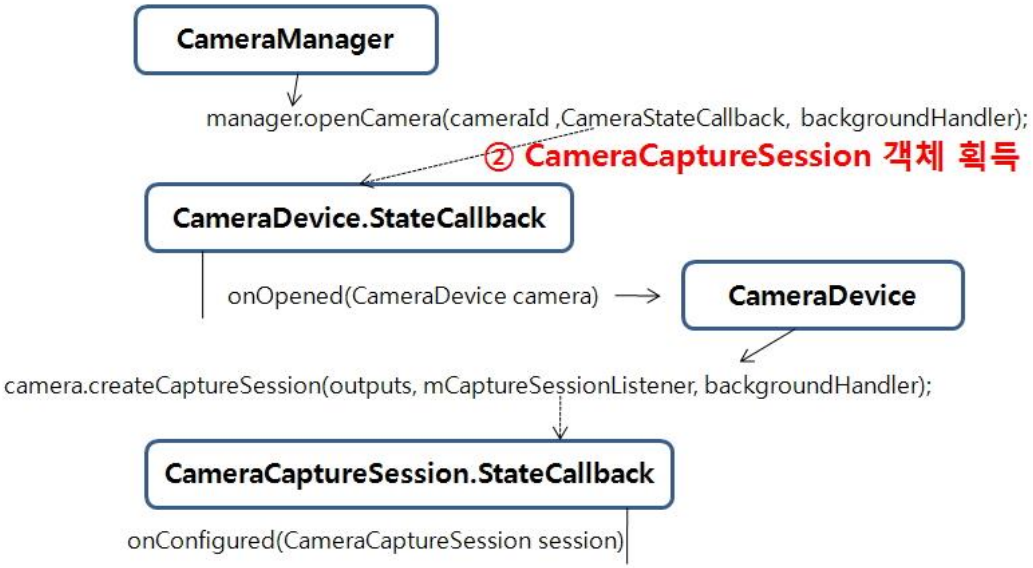
```
StreamConfigurationMap info = cameraCharacteristics
        .get(CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);Size[]
outputSizes=info.getOutputSizes(ImageFormat.JPEG);
```

- 카메라의 방향

```
int mSensorOrientation = cameraCharacteristics
.get(CameraCharacteristics.SENSOR_ORIENTATION);
```

28.3 Camera2 API

- CameraCaptureSession



- 카메라를 점유

```
manager.openCamera(cameraId, stateCallback,handler);
```

28.3 Camera2 API

- CameraDevice.StateCallback

```
CameraDevice.StateCallback stateCallback = new CameraDevice.StateCallback() {
    @Override
    public void onOpened(CameraDevice camera) {

        Lab2Activity.this.camera = camera;
        try {
            List<Surface> outputs = Arrays.asList(
                    holder.getSurface(), reader.getSurface());
            camera.createCaptureSession(outputs, sessionListener,
                    handler);
        } catch (Exception ex) {
        }
    }
    @Override
    public void onDisconnected(CameraDevice camera) {
    }
    @Override
    public void onError(CameraDevice camera, int error) {
    }
};
```

28.3 Camera2 API

- CameraCaptureSession.StateCallback

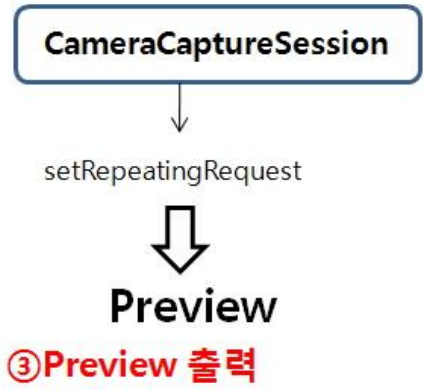
```
CameraCaptureSession.StateCallback sessionListener = new CameraCaptureSession.StateCallback() {
    @Override
    public void onConfigured(CameraCaptureSession session) {
        //...    }

    @Override
    public void onClosed(CameraCaptureSession session) {
        //...
    }

    @Override
    public void onConfigureFailed(CameraCaptureSession session) {
        //...
    }
};
```

28.3 Camera2 API

- Preview 출력

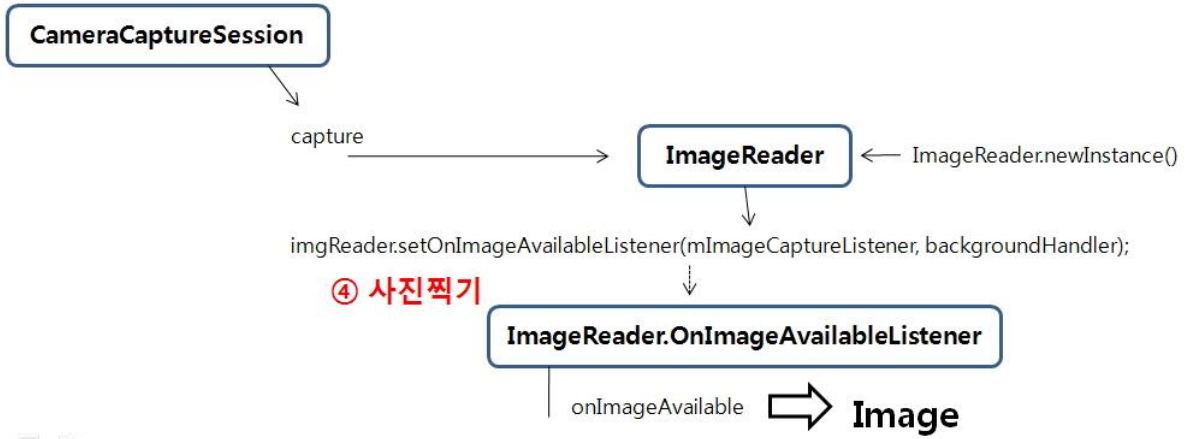


- addTarget( ) 함수를 호출하여 매개변수로 카메라 영상이 출력될 뷰의 surface를 지정

```
CaptureRequest.Builder requestBuilder = camera.createCaptureRequest(camera.TEMPLATE_PREVIEW);
requestBuilder.addTarget(holder.getSurface());
CaptureRequest previewRequest = requestBuilder.build();
//preview 화면 출력..
try {
    session.setRepeatingRequest(previewRequest, /*listener*/null,
                                /*handler*/null);
} catch (CameraAccessException ex) {
    Log.e("kkang", "Failed to make repeating preview request", ex);
}
```

28.3 Camera2 API

- 사진 촬영



- ImageReader

```
ImageReader reader = ImageReader.newInstance(largestSize.getWidth(),
      largestSize.getHeight(), ImageFormat.JPEG, /*maxImages*/2);
reader.setOnImageAvailableListener(captureListener, handler);
```

- capture

```
CaptureRequest.Builder requester =
      camera.createCaptureRequest(camera.TEMPLATE_STILL_CAPTURE);
requester.addTarget(reader.getSurface());

// Orientation.. 처리 하지 않으면 사진 write의 방향이 안 맞음
int rotation = getWindowManager().getDefaultDisplay().getRotation();
requester.set(CaptureRequest.JPEG_ORIENTATION, Camera2Util.getOrientation(rotation,          characteristics));
session.capture(requester.build(), /*listener*/null, /*handler*/null);
```

28.3 Camera2 API

- 이미지 획득

```
ImageReader.OnImageAvailableListener captureListener = new ImageReader.OnImageAvailableListener() {
    @Override
    public void onImageAvailable(ImageReader reader) {
        Log.d("kkang", "onImageAvailable.....");
        //thread가 매게변수의 run 함수를 호출하게 된다
        handler.post(new CapturedImageSaver(reader.acquireNextImage()));
    }
};
```

Step by Step 28-2 – Camera2 API

Camera2 API를 이용하여 카메라 영상을 화면에 출력하고 사진을 촬영하는 기능을 테스트

1. Activity 생성
2. 파일 복사
3. Lab28_2Activity 작성
4. Lab28_2Activity.java 실행