**GPS 위치기반 서비스 API**

**<uses-permission**

**android:name=*"android.permission.ACCESS\_FINE\_LOCATION"* />**

**<uses-permission**

**android:name=*"android.permission.ACCESS\_COARSE\_LOCATION"* />**

<uses-permission

android:name=*"android.permission.INTERNET"* />

<uses-sdk

android:minSdkVersion=*"3"*></uses-sdk>

1. LOCATION\_SERVICE로 **getSystemService 호출 LocationManager 얻음**
2. getBestProvider() getAllProviders() 메서드이용 위치정보 제공자 찾음
3. criteria 객체를 설정해서 그존건들에 가장 근접한 제공자 선택
4. LocationListener 클래스구현.
5. requestLocationUpdates() 메서드를 호출된다 이시점부터 응용프로그램 위치정보 제공된다

**private** **boolean** **getmyLocation**() {

**LocationManager** locationManager;

locationManager = (**LocationManager**)**getSystemService**(**Context**.*LOCATION\_SERVICE*);

**Location** location = locationManager.**getLastKnownLocation**(**LocationManager**.*GPS\_PROVIDER*);

**boolean** sw=**false**;

**if** (location != **null**) {

lat=location.**getLatitude**();

lon=location.**getLongitude**();

sw = **true**;

} **else** {

lat = 36;

lon = 127.7;

}

**CustInfo** myInfo = **new** **CustInfo**("mylocal","위치","접속위치",lat,lon,999);

**addCustInfo**(myInfo);

**return** sw;

}

**public** **class** **GPS** **extends** **Activity** **implements** LocationListener{

**LocationManager** location = **null**;

@Override

**protected** **void** **onCreate**(**Bundle** savedInstanceState) {

**적절한 공급자 찾고**

**LocationManager**

**객체 얻어오고**

**super**.**onCreate**(savedInstanceState);

**setContentView**(**R**.**layout**.*gps*);

location = (**LocationManager**) **getSystemService**(**Context**.*LOCATION\_SERVICE*);

**final** **Button** start = (**Button**)**findViewById**(**R**.**id**.*start*);

**final** **Button** stop = (**Button**)**findViewById**(**R**.**id**.*stop*);

**final** **TextView** status = (**TextView**)**findViewById**(**R**.**id**.*status*);

start.**setOnClickListener**(**new** **View**.**OnClickListener**() {

**public** **void** **onClick**(**View** v) {

**Toast**.*makeText*(**GPS**.**this**, "Starting....", **Toast**.*LENGTH\_SHORT*).**show**();

Iterator<**String**> providers = location.**getAllProviders**().**iterator**();

**while**(providers.**hasNext**()) {

**Log**.*v*("Location", providers.**next**());

}

**Criteria** criteria = **new** **Criteria**();

criteria.**setAccuracy**(**Criteria**.*NO\_REQUIREMENT*);

criteria.**setPowerRequirement**(**Criteria**.*NO\_REQUIREMENT*);

**String** best = location.**getBestProvider**(criteria, **true**);

status.**setText**("Best provider: " + best);

location.**requestLocationUpdates**(best, 1000, 0, GPS.this);

**리스너 등록한거 탄다!**

**이시점부터 시작이다**

**4가지의 함수 오버라이딩해야함**

start.**setVisibility**(**View**.*GONE*);

stop.**setVisibility**(**View**.*VISIBLE*);

}

});

stop.**setOnClickListener**(**new** **View**.**OnClickListener**() {

2개 제공자의 활성화/비활성화 체크

1나 제공자 상태 알려주는거

1나 onLocationChaned가 바로 위치정보 응용프로그램에게 제공하는 메서드

**public** **void** **onClick**(**View** v) {

stop.**setVisibility**(**View**.*GONE*);

start.**setVisibility**(**View**.*VISIBLE*);

}

});

}

**Location** lastLocation = **null**;

**public** **void** **onLocationChanged**(**Location** location) {

**String** locInfo = **String**.*format*("Current loc = (%f, %f) @ (%f meters up)", location.**getLatitude**(), location.**getLongitude**(), location.**getAltitude**() );

**if** (lastLocation != **null**) {

**float** distance = location.**distanceTo**(lastLocation);

locInfo += **String**.*format*("\n Distance from last = %f meters", distance);

}

lastLocation = location;

**Geocoder** coder = **new** **Geocoder**(**this**);

**try** {

Iterator<**Address**> addresses = coder.**getFromLocation**(location.**getLatitude**(), location.**getLongitude**(), 3).**iterator**();

**if** (addresses != **null**) {

**while** (addresses.**hasNext**()) {

**Address** namedLoc = addresses.**next**();

**String** placeName = namedLoc.**getLocality**();

**String** featureName = namedLoc.**getFeatureName**();

**String** country = namedLoc.**getCountryName**();

**String** road = namedLoc.**getThoroughfare**();

locInfo += **String**.*format*("\n[%s][%s][%s][%s]", placeName, featureName, road, country);

**int** addIdx = namedLoc.**getMaxAddressLineIndex**();

**for** (**int** idx = 0; idx <= addIdx; idx++){

**String** addLine = namedLoc.**getAddressLine**(idx);

locInfo += **String**.*format*("\nLine %d: %s", idx, addLine);

}

}

}

} **catch** (**IOException** e) {

**Log**.*e*("GPS", "Failed to get address", e);

}

**TextView** status = (**TextView**)**findViewById**(**R**.**id**.*status*);

status.**setText**(locInfo);

**final** **String** geoURI = **String**.*format*("geo: %f,%f", location.**getLatitude**(), location.**getLongitude**());

**Button** show = (**Button**) **findViewById**(**R**.**id**.*show\_map*);

show.**setOnClickListener**(**new** **View**.**OnClickListener**() {

**public** **void** **onClick**(**View** v) {

**Intent** map = **new** **Intent**(**Intent**.*ACTION\_VIEW*, **Uri**.*parse*(geoURI));

**startActivity**(map);

}

});

show.**setVisibility**(**View**.*VISIBLE*);

}

**public** **void** **onProviderDisabled**(**String** provider) {

**Log**.*v*("GPS", "Provider disabled "+ provider);

}

**public** **void** **onProviderEnabled**(**String** provider) {

**Log**.*v*("GPS", "Provider enabled "+ provider);

}

**private** **static** **final** Map<**Integer**, **String**> *providerStatusMap* = **new** **HashMap**<**Integer**, **String**>() {

{

**put**(**LocationProvider**.*AVAILABLE*, "Available");

**put**(**LocationProvider**.*OUT\_OF\_SERVICE*, "Out of Service");

**put**(**LocationProvider**.*TEMPORARILY\_UNAVAILABLE*, "Temporarily Unavailable");

**put**(-1, "Not Reported");

}

};

**public** **void** **onStatusChanged**(**String** provider, **int** status, **Bundle** extras) {

**int** satellites = extras.**getInt**("satellites", -1);

**String** statusInfo = **String**.*format*("Provider: %s, status: %s, satellites: %d", provider, *providerStatusMap*.**get**(status), satellites);

**Log**.*v*("GPS", statusInfo);

**TextView** statusText = (**TextView**)**findViewById**(**R**.**id**.*status*);

statusText.**setText**(statusInfo);

}

}

**LayoutParam**

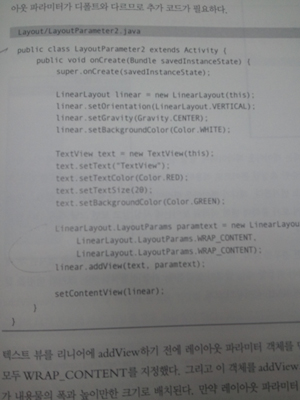
일반 속성들은 뷰자체 속성을 지정한다 뷰혼자만의 속성이다.

이에비해 레이아웃 파라미터는 뷰의 속성이 아니라 뷰가 배치되는 부모 즉 레이아웃에 소석되는 속성이며 부몽[ㄱ[ 차일드 뷰를 배치할 방법을 지시한다

뷰자체의 성질을 규정하는 것이 아니라 뷰외부의 관계를 지정하는거에 봐서 특수성을 가진다.

뷰가 레이아웃 파라미터를 조작하지 못하는 근본적인 이유는 자신이 어떤 레잉웃에 놓일것인지 미리알수 없기때문이다.

만약 모든 위젯이 모든 레이아웃에 배치정보를 가진다면 엄청난 용량압박이 올거다.



**CAMERA 카메라**

1. SurfaceView 클래스를 확장 하여 SurfaceHolder.Callback 인터페이스 구현
2. surfaceCreated() 메서드에서 Camera인스턴스 얻는다.
3. surfaceChanged() 메서드에서 Camera.Paremeters를 이용해 관련 속성 설정후 startPreview()메서드 호출
4. 촬영 사건에 반응할 메서드를 CameraSurfaceView에 추가
5. CameraSurfaceView를 적절한 레이아웃추가
6. 사용자가 사진을 찍을수 있게 하는 위젯(버튼등)을 화면에 추가
7. 찍은 이미지의 저장을 위해 pictureCallback클래스 구현
8. Android.permission.CAMERA권한 AndroideManifast.xml 파일에 추가
9. surfaceDestroyed()메서드에서 camera 객체 해제.

**class** **CameraSurfaceView** **extends** **SurfaceView** **implements** SurfaceHolder.Callback {

**private** SurfaceHolder mHolder;

**private** **Camera** camera = **null**;

**public** **CameraSurfaceView**(**Context** context) {

**super**(context);

// Install a SurfaceHolder.Callback so we get notified when the

// underlying surface is created and destroyed

mHolder = **getHolder**();

mHolder.**addCallback**(**this**);

mHolder.**setType**(SurfaceHolder.*SURFACE\_TYPE\_PUSH\_BUFFERS*);

}

**public** **void** **surfaceChanged**(SurfaceHolder holder, **int** format, **int** width,

**int** height) {

**Camera**.**Parameters** params = camera.**getParameters**();

**int** w = width;

**int** h = height;

params.**setPreviewSize**(800, 480);

camera.**setParameters**(params);

camera.**startPreview**();

}

**public** **void** **surfaceCreated**(SurfaceHolder holder) {

camera = **Camera**.*open*();

**try** {

camera.**setPreviewDisplay**(mHolder);

} **catch** (**Exception** e) {

camera.**release**();

camera = **null**;

**Log**.*e*("Camera", "Failed to set camera preview display", e);

}

}

**public** **void** **surfaceDestroyed**(SurfaceHolder holder) {

camera.**stopPreview**();

camera.**release**();

camera = **null**;

}

**public** **boolean** **capture**(**Camera**.PictureCallback jpegHandler) {

**if** (camera != **null**) {

camera.**takePicture**(**null**, **null**, jpegHandler);

**return** **true**;

} **else** {

**return** **false**;

}

}

}

class Preview extends SurfaceView implements SurfaceHolder.Callback {

02.SurfaceHolder mHolder;

03.Camera mCamera;

04.

05.Preview(Context context) {

06.super(context);

07.

08.// SurfaceHolder.Callback을 설정함으로써 Surface가 생성/소멸되었음을

09.// 알 수 있습니다.

10.mHolder = getHolder();

11.mHolder.addCallback(this);

12.mHolder.setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS);

13.}

14.

15.public void surfaceCreated(SurfaceHolder holder) {

16.// Surface가 생성되었다면, 카메라의 인스턴스를 받아온 후 카메라의

17.// Preview 를 표시할 위치를 설정합니다.

18.mCamera = Camera.open();

19.try {

20.mCamera.setPreviewDisplay(holder);

21.} catch (IOException exception) {

22.mCamera.release();

23.mCamera = null;

24.// TODO: add more exception handling logic here

25.}

26.}

27.

28.public void surfaceDestroyed(SurfaceHolder holder) {

29.// 다른 화면으로 돌아가면, Surface가 소멸됩니다. 따라서 카메라의 Preview도

30.// 중지해야 합니다. 카메라는 공유할 수 있는 자원이 아니기에, 사용하지 않을

31.// 경우 -액티비티가 일시정지 상태가 된 경우 등 - 자원을 반환해야합니다.

32.mCamera.stopPreview();

33.mCamera = null;

34.}

35.

36.public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {

37.// 표시할 영역의 크기를 알았으므로 해당 크기로 Preview를 시작합니다.

38.Camera.Parameters parameters = mCamera.getParameters();

39.parameters.setPreviewSize(w, h);

40.mCamera.setParameters(parameters);

41.mCamera.startPreview();

42.}

43.

44.}

이전 글에서는 아직 surfaceCreated, surfaceDestroyed, surfaceChanged 메소드가 구현되어있지 않았는데, 여기에서 이 메소드들까지 구현되어 있는 것을 확인할 수 있습니다. 하나하나씩 차근차근 살펴보도록 하겠습니다.

[view source](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#viewSource)[print](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#printSource)[?](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#about)

1.class Preview extends SurfaceView implements SurfaceHolder.Callback {

2.SurfaceHolder mHolder;

3.Camera mCamera; // Camera객체 추가

우선, SurfaceView에 카메라에서 받은 영상을 표시하기 위해 Camera객체가 추가된 것을 확인할 수 있습니다.

[view source](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#viewSource)[print](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#printSource)[?](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#about)

01.public void surfaceCreated(SurfaceHolder holder) {

02.// Surface가 생성되었다면, 카메라의 인스턴스를 받아온 후 카메라의

03.// Preview 를 표시할 위치를 설정합니다.

04.mCamera = Camera.open();

05.try {

06.mCamera.setPreviewDisplay(holder);

07.} catch (IOException exception) {

08.mCamera.release();

09.mCamera = null;

10.// TODO: add more exception handling logic here

11.}

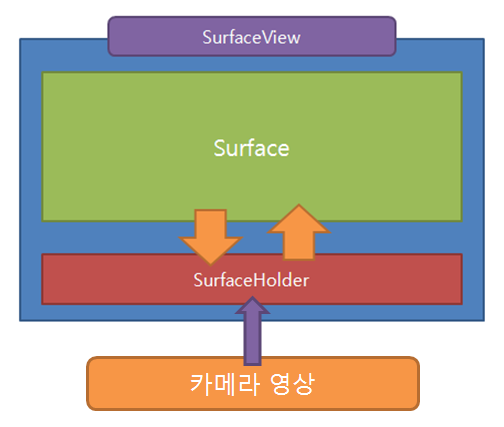
12.}

그 다음, surfaceCreated 메소드의 구현부입니다.

마치 액티비티의 생애주기 메소드 중 하나인 onCreate(Bundle)의 형태와 유사합니다. 이 메소드를 구현함으로써 SurfaceView가 생성되었을 때, 즉 "화면에 표시" 될 때 해야 할 동작을 처리할 수 있습니다.

여기에서는 카메라 객체를 받아와 카메라로부터 영상을 받을 수 있도록 초기화해주는 동작을 수행하고 있습니다. (Camera.open()) 그 다음, setPreviewDisplay(holder)메소드를 통해 카메라로부터 받은 프리뷰 영상을 어디에 표시해줄지 지정하고 있습니다.

앞의 글에서 SurfaceView는 SurfaceView 자체가 내용을 표시하는 것이 아니라 내부의 SurfaceHolder를 통해 최종적으로는 Surface라는 객체 위에 그 내용을 표시한다고 하였는데, 위의 경우는 아래와 같이 카메라의 영상이 처리된다고 볼 수 있습니다.

[](http://cfile1.uf.tistory.com/original/155002104B37FA3248877D)

그 다음을 한번 보도록 하죠. 예외처리 부분이군요.

[view source](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#viewSource)[print](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#printSource)[?](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#about)

1.} catch (IOException exception) {

2.mCamera.release();

3.mCamera = null;

4.// TODO: add more exception handling logic here

5.}

오류가 발생해서 프리뷰 화면을 제데로 표시하지 못했다면, release() 메소드를 사용하여 카메라의 자원을 다시 반환하는 것을 확인할 수 있습니다. 카메라는 여러 곳에서 공유할 수 있는 자원이 아니기에 카메라 객체의 사용이 끝났다면 위와 같이 release() 메소드를 사용하여 자원을 반환해야 합니다. 그렇지 않으면 다른 문제가 발생할 수 있습니다. :(

[view source](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#viewSource)[print](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#printSource)[?](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#about)

1.public void surfaceDestroyed(SurfaceHolder holder) {

2.// 다른 화면으로 돌아가면, Surface가 소멸됩니다. 따라서 카메라의 Preview도

3.// 중지해야 합니다. 카메라는 공유할 수 있는 자원이 아니기에, 사용하지 않을

4.// 경우 -액티비티가 일시정지 상태가 된 경우 등 - 자원을 반환해야합니다.

5.mCamera.stopPreview();

6.mCamera = null;

7.}

그 다음, surfaceDestroyed 메소드 구현부입니다. 이 메소드는 SurfaceView가 더이상 화면에 표시되지 않을 때 호출됩니다. 일반적인 View는 액티비티가 일시정지 상태가 된 경우에도 계속 화면에 그 내용을 표시하지만, SurfaceVIew의 경우 표시하는 내용이 다른 뷰에 비해 복잡하기에 굳이 액티비티가 비활성 상태일때 그 내용을 표시할 이유가 없겠죠.

따라서, stopPreview() 메소드를 사용하여 카메라의 프리뷰 영상을 표시하는 것을 중단하고, 카메라 객체를 소멸시킵니다.

[view source](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#viewSource)[print](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#printSource)[?](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#about)

1.public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {

2.// 표시할 영역의 크기를 알았으므로 해당 크기로 Preview를 시작합니다.

3.Camera.Parameters parameters = mCamera.getParameters(); // (1)

4.parameters.setPreviewSize(w, h); // (2)

5.mCamera.setParameters(parameters); // (3)

6.mCamera.startPreview();

7.}

다음은 surfaceChanged메소드입니다. 이 메소드는 surfaceCreated()메소드 호출 이후에 호출되는 메소드로, Surface의 크기에 따라 실질적으로 어떻게 내용을 표시할지를 처리해주는 메소드입니다.

여기에서는 카메라의 여러 설정값들을 담고 있는 parameters를 받아온 후(1), 프리뷰의 크기를 지정합니다.(2) 그 후, setParameters()에 방금 카메라 프리뷰 크기를 수정한 parameter 객체를 인자로 넘겨줌으로써 카메라 객체에서 프리뷰 영상을 표시할 영역의 크기를 설정해주게 됩니다. (3)

이 과정이 모두 끝났다면, startPreview() 메소드를 통해 최종적으로 카메라 프리뷰 영상을 표시해주게 됩니다.

여기까지 해서 SurfaceView 부분의 구현이 모두 끝났습니다. 실질적으로 카메라의 영상을 처리하는 부분은 여기에서 끝났으므로, 남은 것들은 액티비티를 구성하고, 이 뷰의 객체를 생성한 후 액티비티의 화면으로 표시해주는 과정이 남았습니다.

액티비티 구현을 보도록 하죠.

[view source](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#viewSource)[print](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#printSource)[?](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0#about)

01.public class CameraPreview extends Activity {

02.private Preview mPreview;

03.

04.@Override

05.protected void onCreate(Bundle savedInstanceState) {

06.super.onCreate(savedInstanceState);

07.

08.// Hide the window title.

09.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

10.

11.// Create our Preview view and set it as the content of our activity.

12.mPreview = new Preview(this);

13.setContentView(mPreview);

14.}

15.

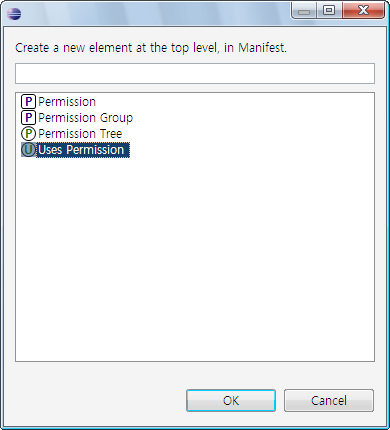
16.}

다른 특이한 것들은 별로 없고, 액티비티의 이름을 표시해주는 타이틀바를 표시하지 않기 위해 requestWindowFeature(Window.FEATURE\_NO\_TITLE) 를 사용하는 것을 확인할 수 있습니다.

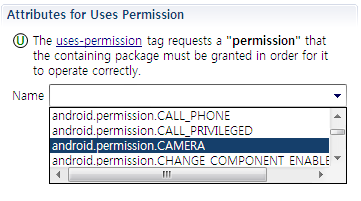
자 이제 거의 다 끝났습니다. 카메라를 사용하기 위해서는 권한이 필요하므로, 메니페스트 파일에 권한을 추가합니다.

[](http://cfile23.uf.tistory.com/original/16200F034B38042E829C55)

위와 같이 메니페스트 파일의 Permission 탭으로 이동한 후, Add..버튼을 누릅니다.

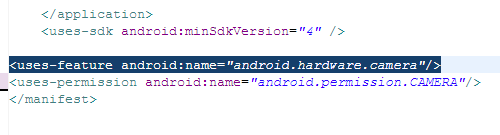
[](http://cfile24.uf.tistory.com/original/18200F034B38042F83FCD1)

Uses Permission을 선택한 후, OK 버튼을 클릭합니다.

[](http://cfile3.uf.tistory.com/original/19200F034B38042F8424F0)

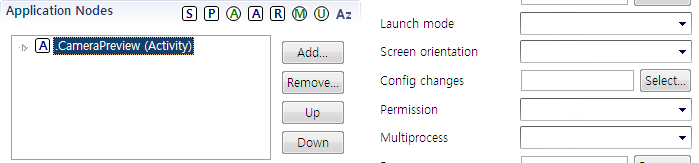
새로 Uses Permission 항목에 카메라 사용 권한인 android.permission.CAMERA 를 선택해주면 권한 추가가 완료됩니다.

그리고, 메니페스트 파일을 열어 수동으로 <uses-feature android:name="android.hardware.camera"/> 를 추가합니다.

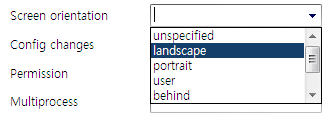
[](http://cfile2.uf.tistory.com/original/17759E164B3804CA975C50)

이 기능은 SDK버전을 선언하는 것과 비슷하게 카메라가 없는 장치에서는 아예 어플리케이션이 설치가 되지 않도록 하는 옵션입니다. 안드로이드를 사용하는 장치가 한두가지가 아니므로 이런 식으로 각 장치의 특성에 맞도록 적절히 조치를 해 주는 것이죠.

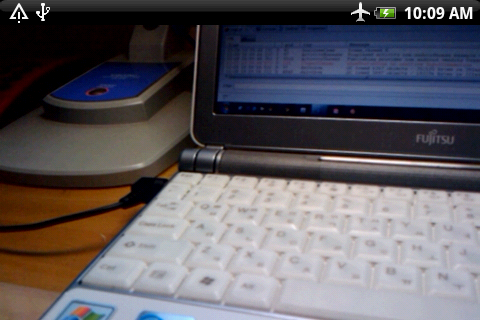
마지막으로, 액티비티를 landscape 모드로 표시하기 위해 메니페스트 파일을 열어 Application 탭으로 간후,

[](http://cfile30.uf.tistory.com/original/177407264B3806EB5CB60C)

Screen orientation을 landscape로 바꾸어줍니다. 이렇게 하면 이 액티비티는 항상 landscape 모드로 표시되게 됩니다.

[](http://cfile8.uf.tistory.com/original/187407264B3806EB5DB2D5)

이 과정이 모두 끝났다면, 어플리케이션을 실행시켜봅시다. 카메라 미리보기가 잘 표시되는 것을 확인할 수 있습니다 :)

[](http://cfile4.uf.tistory.com/original/1658F6274B380762AF5064)

소스파일 첨부합니다.

[[http://cfs.tistory.com/blog/image/extension/zip.gif](http://androidhuman.tistory.com/attachment/cfile4.uf@1718A1254B3808087B3C3E.zip) CameraPreview.zip](http://androidhuman.tistory.com/attachment/cfile4.uf@1718A1254B3808087B3C3E.zip)

[http://androidhuman.tistory.com/entry/카메라를-이용하자-카메라-프리뷰-띄우기](http://androidhuman.tistory.com/entry/%EC%B9%B4%EB%A9%94%EB%9D%BC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%98%EC%9E%90-%EC%B9%B4%EB%A9%94%EB%9D%BC-%ED%94%84%EB%A6%AC%EB%B7%B0-%EB%9D%84%EC%9A%B0%EA%B8%B0)

센서.