**package** kdt.hello;

기존 Form과비슷하다.

처음에 onCreate를 부른다.

**import** android.app.Activity;

**import** android.os.Bundle;

**public** **class** HiWindow **extends** Activity {

오버라이딩해서 사용

/\*\* Called when the activity is first created. \*/

@Override

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

**super**.onCreate(savedInstanceSt ate);

사용자 레이아웃을 배치한다.

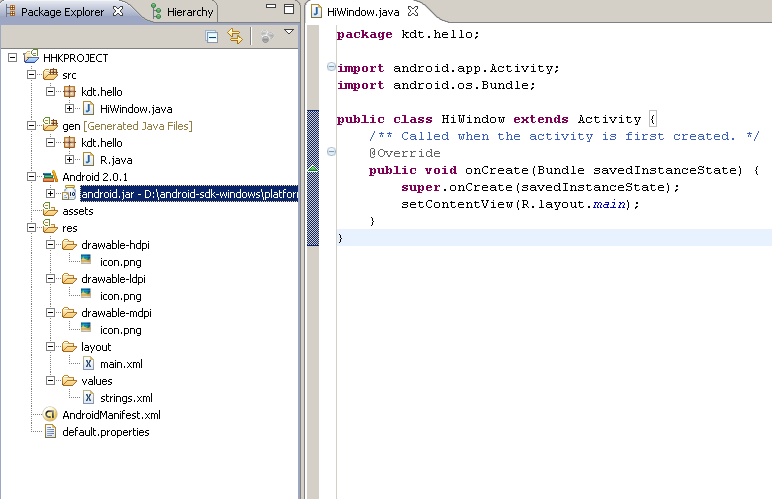
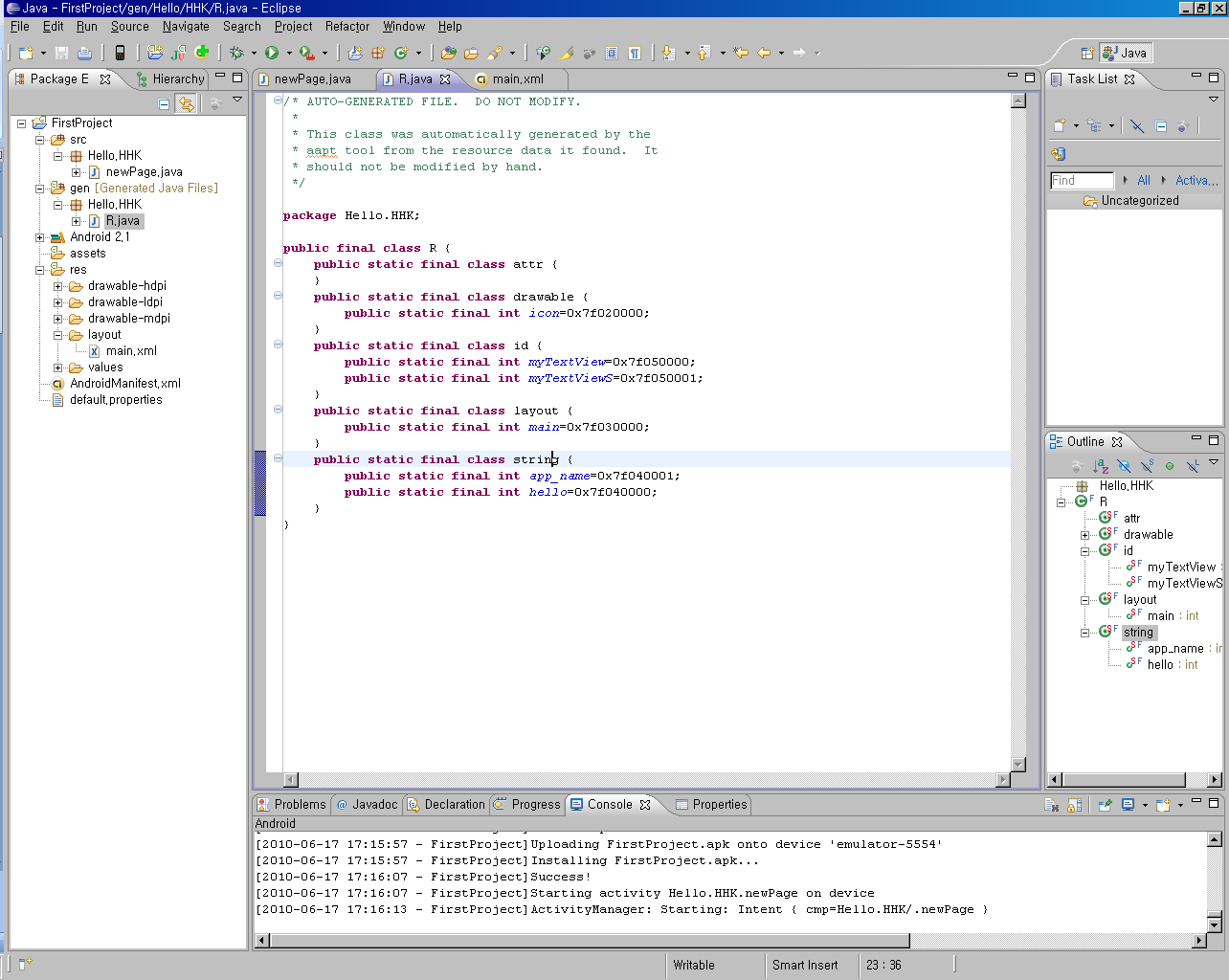
setContentView(R.layout.*main*);

}

}

기본 안드로이드 컴파일 배포 순서: CODE ->Compile -> \*.dex -> \*apk (묵는다압축) ->설치 에뮬

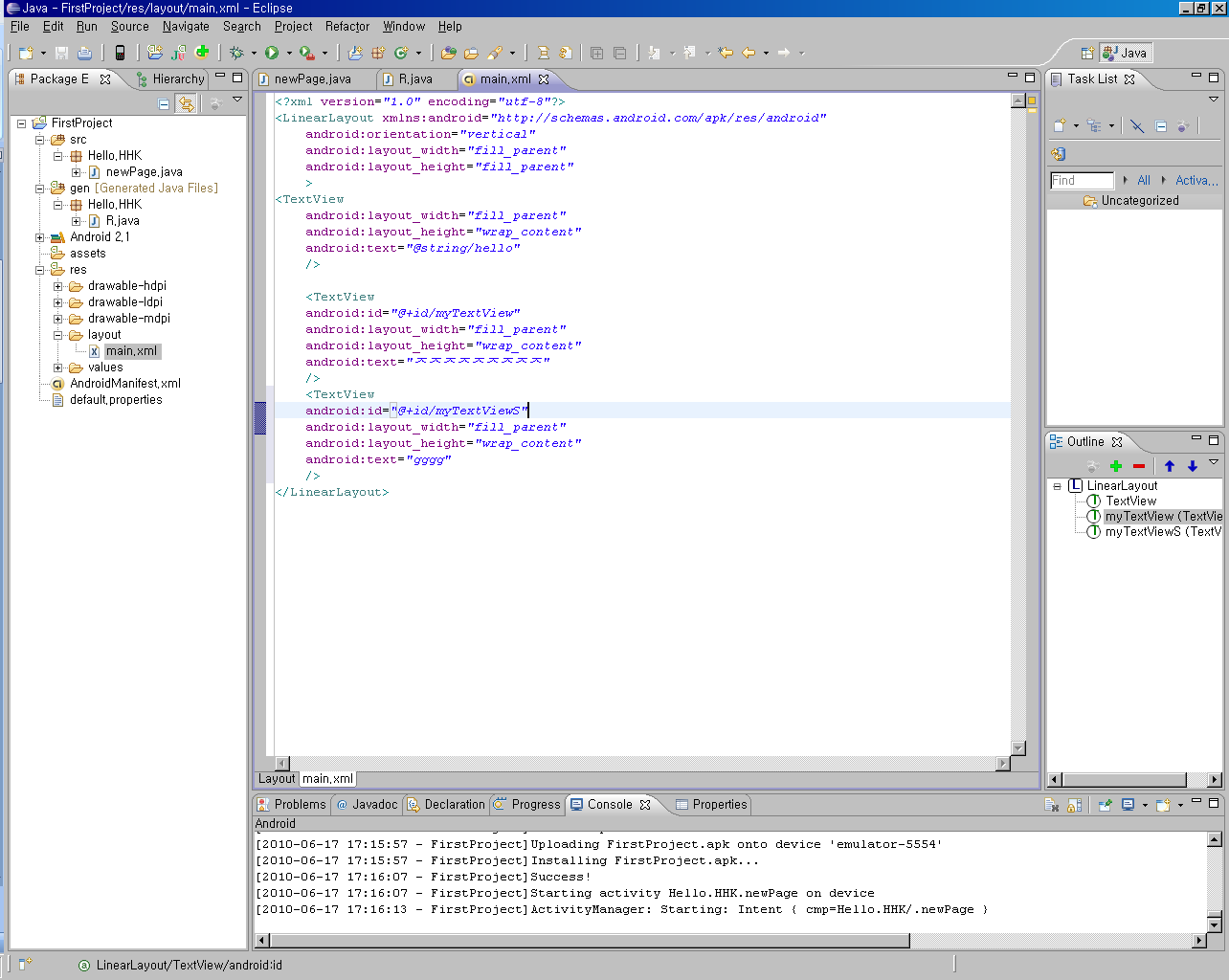
-> open에뮬 -> 푸로그램실행

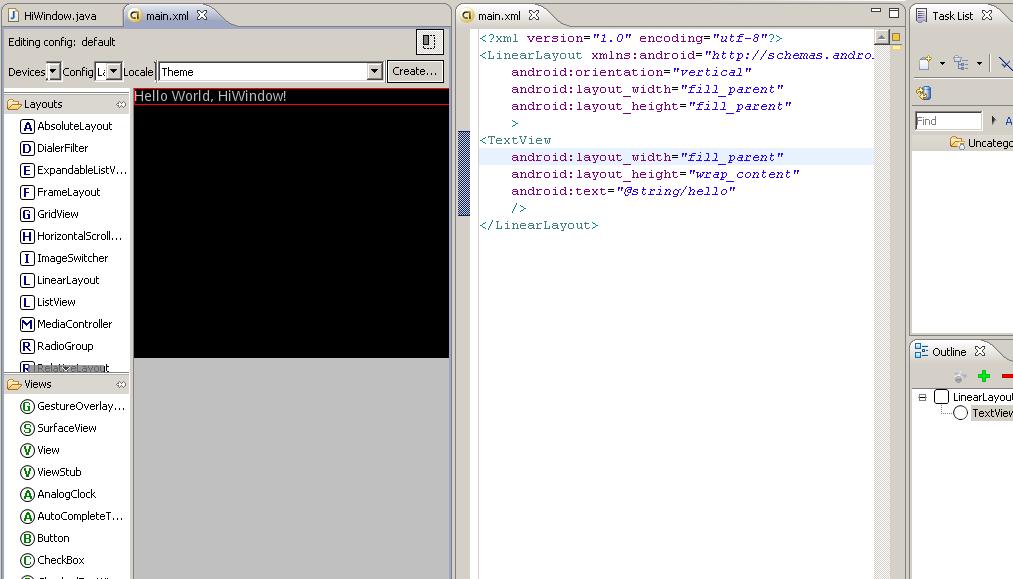


Foreground activity ..있음.

background service 있음sms같은거

intermittent activity 적절할때 사용자에게 통지. 미디어 플레이어 대표적.

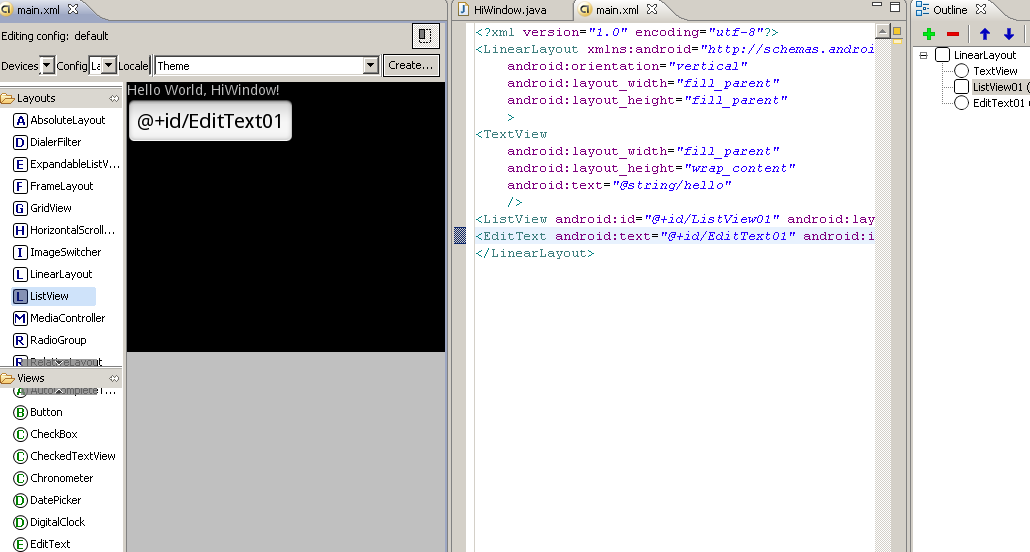
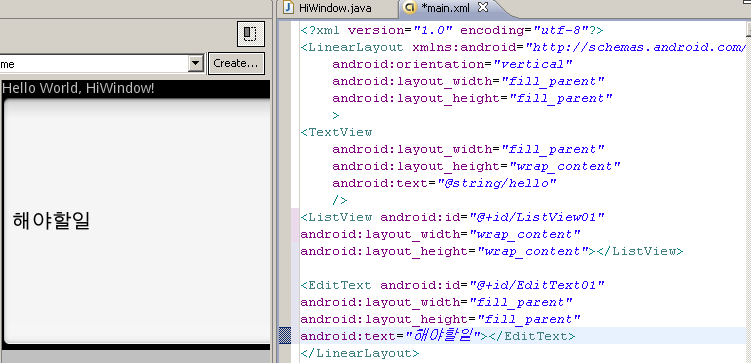
View단을 XML로 간편하게 하자



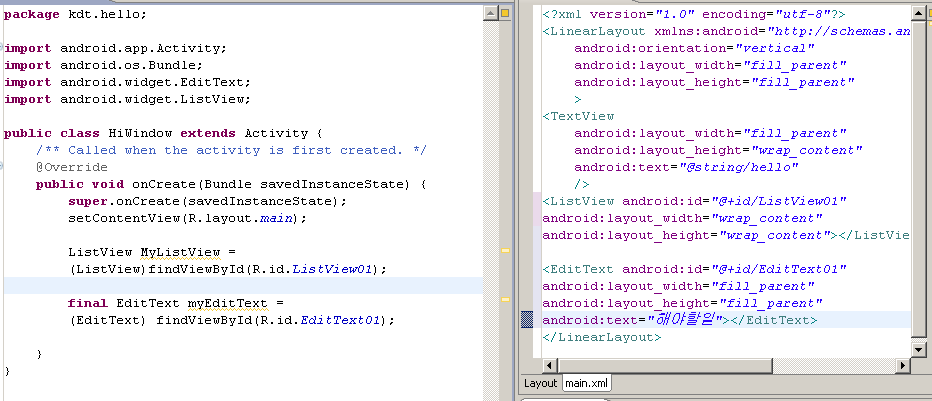
TextView MyTextView = (TextView)findViewById(R.id.myTextView);

필요하다면 레이아웃을 이같이 직접 레퍼런스 가져올수 있다.

화면 디자인



내가 만든 컴포넌트를 선택하자.getElement



**public** **class** **HelloActivity** **extends** **Activity** {

/\*\* Called when the activity is first created. \*/

@Override

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

**super**.onCreate(savedInstanceState);

//뷰를 부풀린다

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

//UI래퍼런스를 가져온다

**ListView** myListview = (**ListView**)findViewById(**R**.**id**.*myListView*);

**final** **EditText** myEditText = (**EditText**)findViewById(**R**.**id**.*myEditText*);

//어레이 리스트

**final** **ArrayList**<**String**> todoItems = **new** **ArrayList**<**String**>();

//어레이 어답터

**final** **ArrayAdapter**<**String**> aa;

aa =**new** **ArrayAdapter**<**String**>(**this**, android.**R**.**layout**.*simple\_list\_item\_1*,todoItems);

myListview.setAdapter(aa);

myEditText.setOnKeyListener(**new** OnKeyListener() {

@Override

**public** **boolean** onKey(**View** v, **int** keyCode, **KeyEvent** event) {

// **TODO** Auto-generated method stub

**if**(event.getAction() == **KeyEvent**.*ACTION\_DOWN*){

**if**(keyCode == **KeyEvent**.*KEYCODE\_DPAD\_CENTER*){

todoItems.add(0,myEditText.getText().toString());

aa.notifyDataSetChanged();

myEditText.setText("");

**return** **true**;

}

}

**return** **false**;

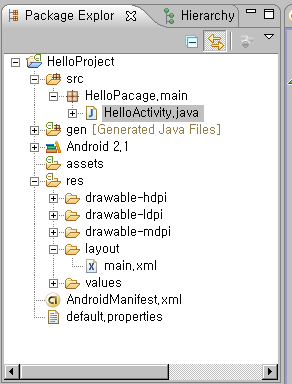
}

});

}

}

Androidmanifest.xml

어플과 어플케션이 가진 컴포넌트의 구조 및 메타 데이터를 정의하도록 도와준다 매니페스트는 여러분의애플리케이션을 구성하는 각각의 컴포넌트 액티비트 서비스 콘텐트 공급자 브로드 캐스트 수신자에 대한 노드를 포함하고 있으며 인텐트 필터와 권환을 사용해 이들이 다른 컴포넌트 및 어플과 어떻게 상호작용하는지 결정짓는다.

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"HelloPacage.main"*

android:versionCode=*"1"*

android:versionName=*"1.0"*>

<application android:icon=*"@drawable/icon"* android:label=*"@string/app\_name"* android:theme=*"@style/ToDoTheme"*>

<activity android:name=*".HelloActivity"*

android:label=*"@string/app\_name"*>

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

</application>

<uses-sdk android:minSdkVersion=*"7"* />

</manifest>

Manifest

는 애플리케이션을 구성하는 애플리케이션 컴포넌트 보안설정 테스트 클래스를 정의하는 노드를 가진다.

Application

오직하나 노드만 존재하며 어플리케이션 메타데이터(타이틀,아이콘,테마등) 지정한다 또한 어플리케이션 노드는 애ㅐ플리케이션 컴포넌트를 지정하는데 사용되는 액티비트 서비스 콘텐츠 공급자 브로드캐스트 수신자 태그를 담는 컨테이너 역할을 한다.

Activity

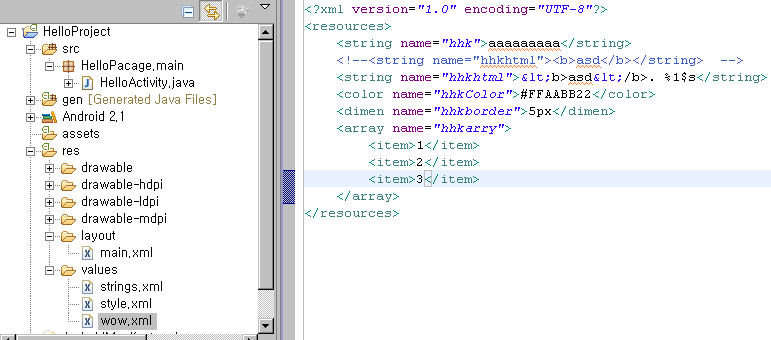
메인 액티비트를 비롯 ㅍ시될수 있는 기타 다른 화면이나 다이얼 로그르 ㄹ반드시 포함해야한다

Service 액티비트와 마찬가지로 애플리케이션에서 사용되는 각 서비스 클래스에 대해 새로운 서비스 태그를 하나씩 생성한다 ….

등등많다. 70p참조

**외부 설정값 (리소스) 사용하기**

1res/values 안에 xml을 생성해놓고 사용하면된다.

****

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<resources>

<string name=*"hhk"*>aaaaaaaaa</string>

<!--<string name="hhkhtml"><b>asd</b></string> -->

<string name=*"hhkhtml"*>&lt;b>asd&lt;/b>. %1$s</string>

<color name=*"hhkColor"*>#FFAABB22</color>16진//컬러 #RGB /RRGGBB // ARGB //ARRGGBB

<dimen name=*"hhkborder"*>5px</dimen> //px in pt mm dp, sp,

<array name=*"hhkarry"*>

<item>1</item>

<item>2</item>

<item>3</item>

</array>

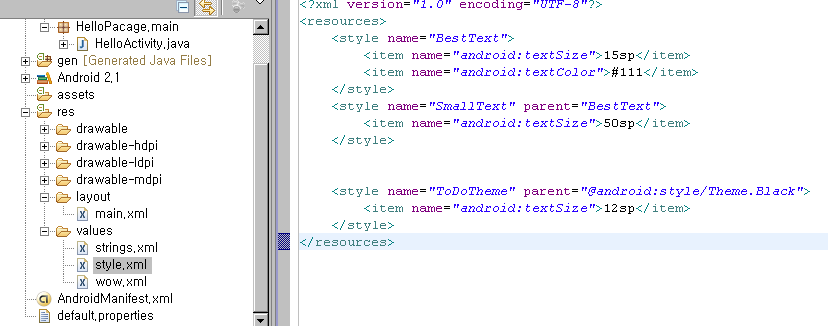
</resources>

**Html도 넣을수 있지만 이스케이프를 해줘야한다 가져다 사용할때는**

String rString =getString(r.string.stop\_message);

String fString =String.format(rString,”Collaborate and listen.”);

CharSequence styledString = Html.formHtml(fString);

**Stlye 지정가능. ** <?xml version=*"1.0"* encoding=*"UTF-8"*?>

<resources>

<style name=*"BestText"*>

<item name=*"android:textSize"*>15sp</item>

<item name=*"android:textColor"*>#111</item>

</style>

<style name=*"SmallText"* parent=*"BestText"*>

<item name=*"android:textSize"*>50sp</item>

</style>

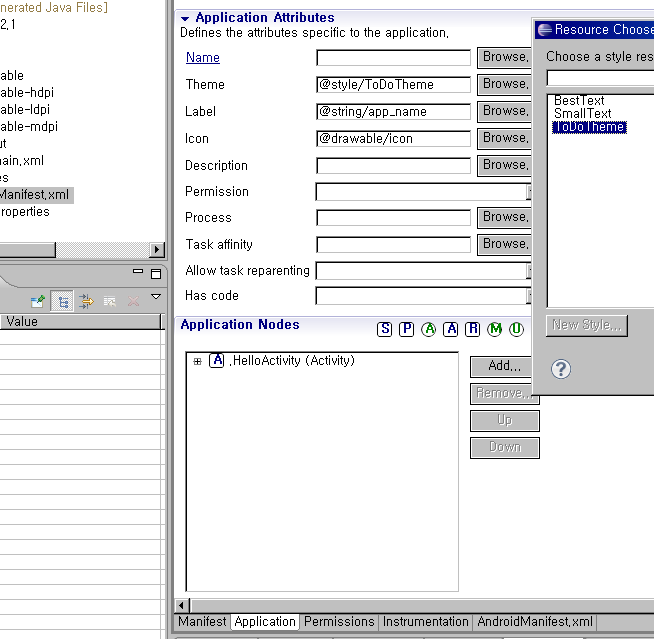
<style name=*"ToDoTheme"* parent=*"@android:style/Theme.Black"*>

<item name=*"android:textSize"*>12sp</item>

</style>

</resources>

스타일 적용

**레이아웃**

화면에 보는 레이아웃은 xml로 되어있다 이걸 부풀리는거는

onCreate 와 setContentView를 통해 부풀려진다. 참조도 다른 레이아웃을 할수있다.

Res/layout 에저장된다

**애니메이션**

트윈드 애니메이션(늘리고 비틀고 이동하고) 과 프레임바이 프레임 (그림 그리고 또그림그리고)에니메이션이있다.

Res/anim에 저장된다 p80 참고.

**그외 리소스 참조해서 사용하기**

코드상에서 static Class R를 통해 참조 할것이다 리소스접근 R은 외부리소스 기반으로 생성되는 클래스로서 프로젝트를 컴파일 하면 만들어진다 87p 참고. 리소스의 리소스 참조가능.

**런타임 구성 변경**

안드로이드는 각 애플리케이션을 종료하고 재시작한두ㅠㅣ 리소스 값을 다시 읽어 들임으로써 위치 하드웨어 에대한 런타임 변경을 지원한다   
**configChanges 속성으로는.**

Onrientation 화면이 세로 방향이나 가로 방향으로 회전됐다.

keyboardHidden 키보드가 보여졌거나 숨겨졌다

fontScale 사용자가 선호하는 글꼴 크기변경했다

locale 사용자가 다른언어 설정을 선택했다

keyboard 키보드 종류가 바뀌었다 예컨대 폰은 간단한 조작으로 풀키보드를 꺼내기위한 12키패트를 가질 것

touchscreen 또는 navigation 키보드 종류나 탐색방법이 바뀌었다 보통 발생안함

위옵션은 중복 가능 합니다 |연결



<?xml version=*"1.0"* encoding=*"utf-8"*?>

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"HelloPacage.main"*

android:versionCode=*"1"*

android:versionName=*"1.0"*>

<application

android:icon=*"@drawable/icon"*

android:label=*"@string/app\_name"*

android:theme=*"@style/ToDoTheme"*>

<activity android:name=*".HelloActivity"*

android:label=*"@string/app\_name"* **android:configChanges=*"orientation|keyboard"***

>

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

</application>

<uses-sdk android:minSdkVersion=*"7"* />

</manifest>

**onConfigurationChanged 메서드를 호출한다 ……. 94p 참조.**

**액티비티 변경 및 그릅.**

**P96참조**

**우선순위**

1활성 프로세스 -> 2화면에 보이는 프로세스 ->3 시작된 서비스 프로세스-> 백그라운드 프로세스 -> 빈 프로세스

액티비트 상태

1활성. 2 일시중지 3 중지 4 비활성 p100 참조 함수 p100 참조!!

**Layout**

ViewGroup 에서 파생된 다양한 레이아웃 관리자를 사용해 위에 배열된다.

뷰(View) : 비주얼 인터페이스 사용자 인터페이스와 레이아웃 클래스는 모두 뷰에서 파생된다

뷰 그릅 : 여러 개의 뷰 자식들을 담을수 있다. 뷰클래스의 확장입니다. 인터페이스 구성을 돕는 LinearLayout 같은 레이아웃 관리자를 제공하기 위해 확장되기도 한다.

액티비티 : 사용자의 화면이나 윈도우를 나타낸다 폼에 상응하는 안드로이드 요소이다. 사용자 인터페이스를 나타내려면 뷰나 레이아웃을 액티비티에 지정한다.

**관계**

뷰

뷰그릅

레이아웃

위젯

컨트롤 ….컨트롤 ….컨트롤 ….

컨트롤

**레퍼런스 얻기**

findViewById 를 이용하여 레이아웃 내에 사용된 뷰의 레퍼런스를 얻을수 있다.

**public** **class** **wow** **extends** **Activity** {

/\*\* Called when the activity is first created. \*/

@Override

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

**super**.onCreate(savedInstanceState);

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

**TextView** textview = (**TextView**)findViewById(**R**.**id**.*TextView01*);

textview.setText("wowwowwowowo");

}

}

고전적인 방법으론. 약간 프로그래밍적 관점으로 접근하면

**public** **class** **wow** **extends** **Activity** {

/\*\* Called when the activity is first created. \*/

@Override

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

**super**.onCreate(savedInstanceState);

**TextView** textview = **new** **TextView**(**this**);

textview.setText("wwwwwww");

setContentView(textview);

}

}

**setContentView 는 단일 뷰 인스턴스를 받아들인다**

**그렇기 때문에 뷰그릅화를 시켜야 한다.**

**위젯 도구상자**

TextView

EditText

ListView

Spinner

ListView

Button

CheckBox

RadioButton

등등 http://developer .android.com/guide/tutorials/views/index.html 참고 하면 더많다

**레이아웃**

|  |  |
| --- | --- |
| FrameLayout | 최상단 모서리에 고정시킨다. 자식추가하면 새로운 자식이 이전 것 위에 쌓이게 되며 각각 새로운 뷰는 맨마지막것을 가르키게된다. |
| LinearLayout | 선형 레이아웃은 각각 자식뷰를 수직 혹은 슈평으로 일직선상에 추가한다 수직렝아웃은 행마다 하나씩 자식뷰를 갖는 반면 수평 레아웃은은 하나의 행으로 이루어진 뷰를 갖는다 선형 레이아웃 관리자는 각각 자식뷰마다 사용가능한 공간 안에서 각가의상대적은 크기를 제어하는 가중치를 지정가능하다 |
| RelativeLayout | 상대 레이아웃 자식뷰 각각 위치를 서로 대해 그리고 화면 경게에 대해 상대적으로 정의할수있다. |
| TableLayout | 테이블 레이아웃 행과 열의 격자로 사용해서 뷰 배치 가능하다 |
| AbsoluteLayout | 절대 레이아웃 각 자식뷰의 위치가 절대 좌표로 정의 동적 조절안된다 |
| <http://developer.android.com/huide/topics/ui/index.html> 참고. | |

**레이아웃 xml 로 만들 때..**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

>

<TextView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/hello"*

/>

<TextView android:text=*"@+id/TextView01"* android:id=*"@+id/TextView01"* android:layout\_width=*"wrap\_content"* android:layout\_height=*"wrap\_content"*></TextView>

</LinearLayout>

**이런식으로 LinearLayout 지정가능**

**또한 코딩으로는**

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

**super**.onCreate(savedInstanceState);

**LinearLayout** ll =**new** **LinearLayout**(**this**);

ll.setOrientation(**LinearLayout**.*VERTICAL*);

**TextView** myTextView = **new** **TextView**(**this**);

**EditText** myEditText = **new** **EditText**(**this**);

myTextView.setText("aaaaa");

myEditText.setText("comon");

**int** lHeight = **LinearLayout**.**LayoutParams**.*FILL\_PARENT*;

**int** lWidth = **LinearLayout**.**LayoutParams**.*WRAP\_CONTENT*;

ll.addView(myTextView,**new** **LinearLayout**.**LayoutParams**(lHeight,lWidth));

ll.addView(myEditText,**new** **LinearLayout**.**LayoutParams**(lHeight,lWidth));

setContentView(ll);

}

View제정의

onDraw 메서드 는 그릴 때… 오버라이딩해서 사용해라

onKeyDown 또한… 키눌렸을때..D패드,키보드, 전화끊기 버튼 통화버튼 뒤로가기버튼 캄메라버튼등…

onKeyUp 사용자가 눌렀던 키를 놓을 때

onTrackballEvent 트랙볼움직일때

onTouchEvent 터치스크린눌렀을때 땔 때 이동감지될 때

**커스텀 위젯과 컨트롤만들기**

기본 View클래스는 100\*100 텅빈 정사각 제공 비쥬얼 인터페이스를 만들려면 onMeasure와 onDraw메서드를 재정의할필요있다.

onMeasure은 자신이 차지할 높이와 폭 계산

onDraw메서드는 비쥬얼인터페이스 만들기위한 캔버스 위에 그리기 수행

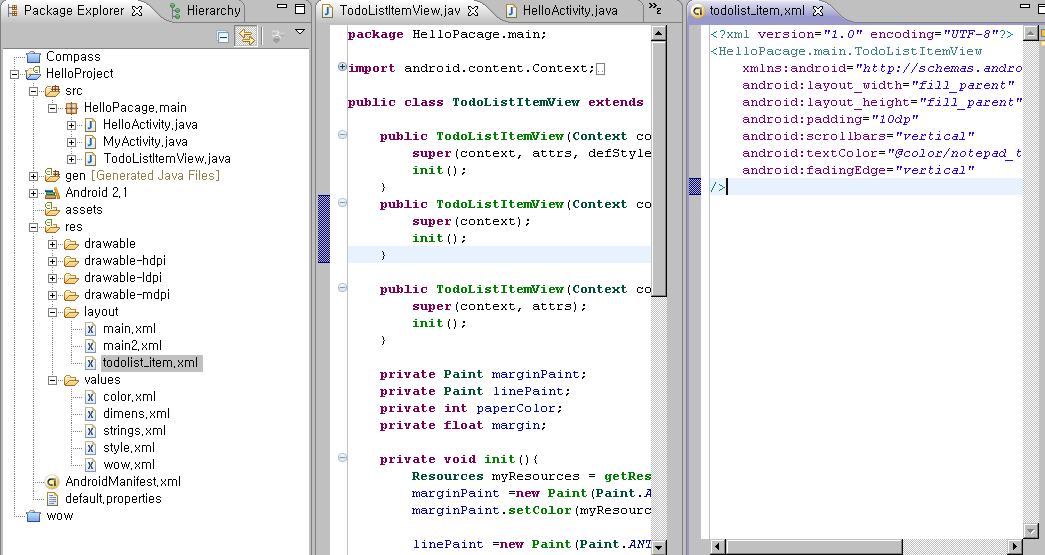
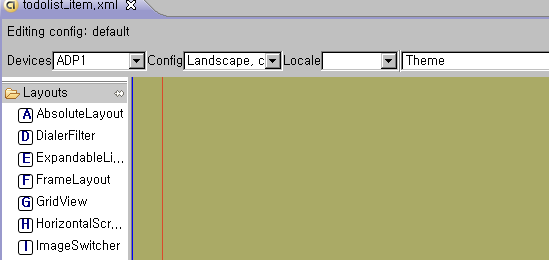
**기존뷰 수정**

View 를 상속받고 생성자를 모두 오버라이딩시킨다. 마지막에 super 생성자 호출해야겠고.

겉모습을 수정하기위해서 onDraw()오버라이딩 해서 메서드를 수정하면된다. 인자 canvas로 그림된다

KeyEvent 함수또한 오버라이딩해서 사용하면된다

뷰에 getMeasuredHeight() 와 getMeasuredWidth()를불러서 위치 파악가능



바로 보인다. 좋다. 이렇게 리소스에 등록을 한다음에 사용할수 있다.

**public** **class** **HelloActivity** **extends** **Activity** {

/\*\* Called when the activity is first created. \*/

@Override

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

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

//뷰를 부풀린다

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

// Log.d("aaaaaaaa","bbbbbbbbbb");

//UI래퍼런스를 가져온다

**ListView** myListview = (**ListView**)**findViewById**(**R**.**id**.*myListView*);

**final** **EditText** myEditText = (**EditText**)**findViewById**(**R**.**id**.*myEditText*);

//어레이 리스트

**final** **ArrayList**<**String**> todoItems = **new** **ArrayList**<**String**>();

**int** resId =**R**.**layout**.*todolist\_item*;

//어레이 어답터

**final** **ArrayAdapter**<**String**> aa;

aa =**new** **ArrayAdapter**<**String**>(**this**, resId,todoItems);

myListview.**setAdapter**(aa);

myEditText.**setOnKeyListener**(**new** **OnKeyListener**() {

@Override

**public** **boolean** **onKey**(**View** v, **int** keyCode, **KeyEvent** event) {

// **TODO** Auto-generated method stub

**if**(event.**getAction**() == **KeyEvent**.*ACTION\_DOWN*){

**if**(keyCode == **KeyEvent**.*KEYCODE\_DPAD\_CENTER*){

todoItems.**add**(0,myEditText.**getText**().**toString**());

aa.**notifyDataSetChanged**();

myEditText.**setText**("");

**return** **true**;

}

}

**return** **false**;

}

});

}

}

**복합컨트롤 만들기**

가장 적합한 layout 을 확장시켜 사용한다.

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"* android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

<EditText android:id=*"@+id/myEditText"* android:text=*"해야할일"* android:layout\_width=*"fill\_parent"* android:layout\_height=*"wrap\_content"*></EditText>

<ListView android:id=*"@+id/myListView"* android:layout\_height=*"wrap\_content"* android:layout\_width=*"fill\_parent"*></ListView>

<ImageView android:id=*"@+id/ImageView01"* android:layout\_width=*"wrap\_content"* android:layout\_height=*"wrap\_content"* android:src=*"@drawable/sa"*></ImageView>

</LinearLayout>

이런 xml 을…가능하지만.훔............위젯을 해당위젯에서 사용하기위해서는

LayoutInflate 시스템 서비스의 inflate 메서드를 사용해서 레이아웃 리소스를 부풀린다.

Inflate 메서드는 레이아웃 리소스를 취하고 부풀려진 뷰를 리턴한다

**public** **class** **layoutmanager** **extends** **LinearLayout** {

**EditText** editText;

**Button** clearButton;

**public** **layoutmanager**(**Context** context) {

**super**(context);

**String** infService = **Context**.*LAYOUT\_INFLATER\_SERVICE*; //context

**LayoutInflater** li;

li = (**LayoutInflater**)**getContext**().**getSystemService**(infService);

li.**inflate**(**R**.**layout**.*main*,**this**,**true**);

//자식컨트롤 레퍼런스 얻어온다

editText = (**EditText**)**findViewById**(**R**.**id**.*Button01*);

clearButton = (**Button**)**findViewById**(**R**.**id**.*Button02*);

**hookupButton**();

// **TODO** Auto-generated constructor stub

}

//각각 컨트롤에 이벤트 리스너를 등록시킬수 있다.

**private** **void** **hookupButton**() {

clearButton.**setOnClickListener**(**new** **Button**.**OnClickListener**(){

@Override

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

// **TODO** Auto-generated method stub

editText.**setText**("");

}

});

}

}

**커스텀 위잿고 컨트롤만들기 아에 처음부터 만들기**

View나 SurfaceView클래스를 하나 확장 상속한다

* View는 Canvas객채 하나를 비롯하여 일련의 그리기 메서드와 Paint 클래스를 제공
* SurfaceView 는 백그라운드 스레드로부터 그리기를 지원하고 3D 그래픽 위해OpenGL 사용하는 캔버스 제공

기본적 View는 100\*100 제공

* onMeasure 내에서 자신이 차지할 높이와 폭을 계산할 것..
* onDraw 메서드는 비쥬얼 인터페이스를 만들기위해 캔버스 위에 그리고 수행되는곳

골격을보자

**public** **class** **myView** **extends** **View** {

//클래스 객체를 코드로 생성할때 필요한 생성자

**public** **myView**(**Context** context) {

**super**(context);

// **TODO** Auto-generated constructor stub

}

//리소스파일로부터 부풀릴때 필요한 생성자

**public** **myView**(**Context** context, AttributeSet attrs) {

**super**(context, attrs);

// **TODO** Auto-generated constructor stub

}

//리소스파일로부터 부풀릴때 필요한 생성자

**public** **myView**(**Context** context, AttributeSet attrs, **int** defStyle) {

**super**(context, attrs, defStyle);

// **TODO** Auto-generated constructor stub

}

/\*

\* **onMeasure 에서 꼭 setMeasuredDimension 를 호출해야한다 안그러면 예외를 던진다**

\*/

@Override

**protected** **void** **onMeasure**(**int** widthMeasureSpec, **int** heightMeasureSpec) {

**int** h=0;

**int** w=0;

h = **measureHeight**(heightMeasureSpec);

w = **measureWidth**(widthMeasureSpec);

**setMeasuredDimension**(w, h);

}

**private** **int** **measureWidth**(**int** measureSpec) {

// 길이를 구한다

//디코더

**int** specMode = **MeasureSpec**.*getMode*(measureSpec);

**int** specSize = **MeasureSpec**.*getSize*(measureSpec);

**return** 0;

}

**private** **int** **measureHeight**(**int** measureSpec) {

// 높이를구한다

//디코더

**int** specMode = MeasureSpec.getMode(measureSpec);

**int** spec

**return** 0;

}

/\*

\* 컨테이너를 디자인한다. 그린다.

\*/

@Override

**protected** **void** **onDraw**(**Canvas** canvas) {

//인터페이스를 그린다

**int** h = **getMeasuredHeight**();

**int** w = **getMeasuredWidth**();

//중시점

**int** px = w/2;

**int** py = h/2;

//새로운 페인트 브러쉬 만든다.

**Paint** mTextPaint = **new** **Paint**(**Paint**.*ANTI\_ALIAS\_FLAG*);

mTextPaint.**setColor**(**Color**.*WHITE*);

**String** displayText="헬로월드";

//텍스트문자열폭측정

**float** textWidth = mTextPaint.**measureText**(displayText);

canvas.**drawText**(displayText, px-textWidth/2, py, mTextPaint);

}

}

**100\*100 을 원치않는다면 onMeasure () 를 재정의해서 사용해야한다**

이메서드는 니가 사용할 공간이 얼마나되? 라고 물으면서 두개으 laoroqustn 를 전달한다

@Override

**protected** **void** **onMeasure**(**int** widthMeasureSpec, **int** heightMeasureSpec) {

**int** h= widthMeasureSpec;

**int** w= heightMeasureSpec;

**setMeasuredDimension**(w, h);

}

그러면 여러분은 setMeasureDimension 에 전달한다. 리턴값은없다.

**재정의할 때**

@Override

**protected** **void** **onMeasure**(**int** widthMeasureSpec, **int** heightMeasureSpec) {

**int** h=0;

**int** w=0;

h = **measureHeight**(heightMeasureSpec);

w = **measureWidth**(widthMeasureSpec);

**setMeasuredDimension**(w, h);

}

**private** **int** **measureWidth**(**int** measureSpec) {

// 길이를 구한다

//디코더

**int** specMode = **MeasureSpec**.*getMode*(measureSpec);

**int** specSize = **MeasureSpec**.*getSize*(measureSpec);

**return** 0;

}

**private** **int** **measureHeight**(**int** measureSpec) {

// 높이를구한다

//디코더

**int** specMode = MeasureSpec.getMode(measureSpec);

**int** spec

**return** 0;

}

크기는 모드값에 따라 컨트롤을 위해 사용 가능한 최대공간(AT\_MOST의경우)이나 여러분의 컨트롤이

차지할 정확한 크기 (EXACTLY 의경우) 를 나타낸다 UNSPECIFIED의 경우에는 크기를 나타내는 어떤 레퍼런스도없다.

쉽게말해 모드값에따라 최대공간이나 컨트롤이 차지할 정확한크기를 나타낸다.

**이렇게해야한다.**

**private** **int** **measure**(**int** measureSpec){

**int** specMode = **MeasureSpec**.*getMode*(measureSpec);

**int** specSize = **MeasureSpec**.*getSize*(measureSpec);

//지정된 제한이 없는경우의 기본크기

**int** result = 500;

**if**(specMode == **MeasureSpec**.*AT\_MOST*){

//최대 크기 범위내에서 컨트롤의 이상적은 크기를 계산한다

result =specSize;

}**else** **if**(specMode == **MeasureSpec**.*EXACTLY*){

//컨트롤이 이들경계내에 맞을수 있다면 그값을 리턴한다

result =specSize;

}

**return** result;

}

**나침반 껍데기만들기**

**package** com.app.compass;

**import** java.util.Iterator;

**import** com.app.compass.R;

**import** android.content.Context;

**import** android.content.res.Resources;

**import** android.graphics.Canvas;

**import** android.graphics.Paint;

**import** android.util.AttributeSet;

**import** android.view.View;

**import** android.view.View.MeasureSpec;

**public** **class** **CompassView** **extends** **View** {

// Paints used to draw the Compass

**private** **Paint** markerPaint;

**private** **Paint** textPaint;

**private** **Paint** circlePaint;

// Cardinal point Strings

**private** **String** northString;

**private** **String** eastString;

**private** **String** southString;

**private** **String** westString;

// Height of text

**private** **int** textHeight;

/\*\* Get or set the bearing displayed by the compass \*\*/

**public** **void** **setBearing**(**float** \_bearing) {

bearing = \_bearing;

}

**public** **float** **getBearing**() {

**return** bearing;

}

**private** **float** bearing;

/\*\* Constructors \*\*/

**public** **CompassView**(**Context** context) {

**super**(context);

**initCompassView**();

}

**public** **CompassView**(**Context** context, AttributeSet attrs) {

**super**(context, attrs);

**initCompassView**();

}

**public** **CompassView**(**Context** context, AttributeSet attrs, **int** defaultStyle) {

**super**(context, attrs, defaultStyle);

**initCompassView**();

}

/\*\* Initialize the Class variables \*\*/

**protected** **void** **initCompassView**() {

**setFocusable**(**true**);

// Get a reference to the external resources

**Resources** r = **this**.**getResources**();

northString = r.**getString**(**R**.**string**.*cardinal\_north*);

eastString = r.**getString**(**R**.**string**.*cardinal\_east*);

southString = r.**getString**(**R**.**string**.*cardinal\_south*);

westString = r.**getString**(**R**.**string**.*cardinal\_west*);

// Create the paints

circlePaint = **new** **Paint**(**Paint**.*ANTI\_ALIAS\_FLAG*);

circlePaint.**setColor**(**R**.**color**.*background\_color*);

circlePaint.**setStrokeWidth**(1);

circlePaint.**setStyle**(**Paint**.Style.*FILL\_AND\_STROKE*);

markerPaint = **new** **Paint**(**Paint**.*ANTI\_ALIAS\_FLAG*);

markerPaint.**setColor**(r.**getColor**(**R**.**color**.*marker\_color*));

textPaint = **new** **Paint**(**Paint**.*ANTI\_ALIAS\_FLAG*);

textPaint.**setColor**(r.**getColor**(**R**.**color**.*text\_color*));

textHeight = (**int**) textPaint.**measureText**("yY");

}

@Override

**protected** **void** **onMeasure**(**int** widthMeasureSpec, **int** heightMeasureSpec) {

// The compass is a circle that fills as much space as possible.

// Set the measured dimensions by figuring out the shortest boundary,

// height or width.

**int** measuredWidth = **measure**(widthMeasureSpec);

**int** measuredHeight = **measure**(heightMeasureSpec);

**int** d = **Math**.*min*(measuredWidth, measuredHeight);

**setMeasuredDimension**(d, d);

}

//

**private** **int** **measure**(**int** measureSpec) {

**int** result = 0;

// Decode the measurement specifications.

**int** specMode = **MeasureSpec**.*getMode*(measureSpec);

**int** specSize = **MeasureSpec**.*getSize*(measureSpec);

**if** (specMode == **MeasureSpec**.*UNSPECIFIED*) {

// Return a default size of 200 if no bounds are specified.

result = 200;

} **else** {

// As you want to fill the available space

// always return the full available bounds.

result = specSize;

}

**return** result;

}

@Override

**protected** **void** **onDraw**(**Canvas** canvas) {

**int** px = **getMeasuredWidth**() / 2;

**int** py = **getMeasuredHeight**() / 2;

**int** radius = **Math**.*min*(px, py);

// Draw the background

canvas.**drawCircle**(px, py, radius, circlePaint);

// Rotate our perspective so that the 'top' is facing the current

// bearing.

canvas.**save**();

canvas.**rotate**(-bearing, px, py);

**int** textWidth = (**int**) textPaint.**measureText**("W");

**int** cardinalX = px - textWidth / 2;

**int** cardinalY = py - radius + textHeight;

// Draw the marker every 15 degrees and a text every 45.

**for** (**int** i = 0; i < 24; i++) {

// Draw a marker.

canvas.**drawLine**(px, py - radius, px, py - radius + 10, markerPaint);

canvas.**save**();

canvas.**translate**(0, textHeight);

// Draw the cardinal points

**if** (i % 6 == 0) {

**String** dirString = "";

**switch** (i) {

**case** (0): {

dirString = northString;

**int** arrowY = 2 \* textHeight;

canvas.**drawLine**(px, arrowY, px - 5, 3 \* textHeight,

markerPaint);

canvas.**drawLine**(px, arrowY, px + 5, 3 \* textHeight,

markerPaint);

**break**;

}

**case** (6):

dirString = eastString;

**break**;

**case** (12):

dirString = southString;

**break**;

**case** (18):

dirString = westString;

**break**;

}

canvas.**drawText**(dirString, cardinalX, cardinalY, textPaint);

}

**else** **if** (i % 3 == 0) {

// Draw the text every alternate 45deg

**String** angle = **String**.*valueOf*(i \* 15);

**float** angleTextWidth = textPaint.**measureText**(angle);

**int** angleTextX = (**int**) (px - angleTextWidth / 2);

**int** angleTextY = py - radius + textHeight;

canvas.**drawText**(angle, angleTextX, angleTextY, textPaint);

}

canvas.**restore**();

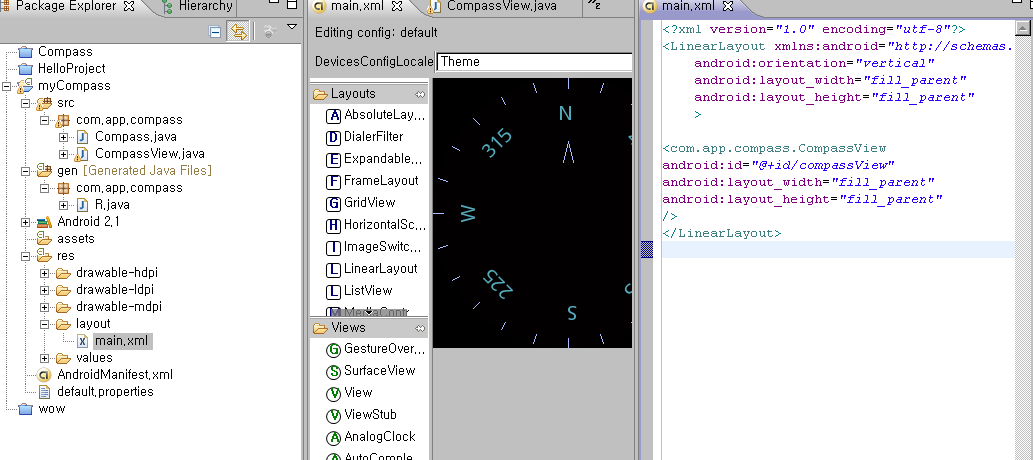
canvas.**rotate**(15, px, py);

}

canvas.**restore**();

}

}



<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

>

이런식으로 페키지부터 명까지

다적어줘야한다.

<com.app.compass.CompassView

android:id=*"@+id/compassView"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

/>

</LinearLayout>

**public** **class** **Compass** **extends** **Activity** {

/\*\* Called when the activity is first created. \*/

@Override

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

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

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

**CompassView** cv =(**CompassView**)**this**.**findViewById**(**R**.**id**.*compassView*);

cv.**setBearing**(500);

}

}

/////////////////////

**public** **class** **Compass** **extends** **Activity** {

/\*\* Called when the activity is first created. \*/

@Override

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

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

// setContentView(R.layout.main);

**CompassView** cv =**new** **CompassView**(**this**);

**setContentView**(cv);

cv.**setBearing**(500);

}

}

이렇게 사용가능하다

코딩으로도 저런걸 정의할수 있다 124page

**메뉴p144참고**

해당뷰가 포커스를 가졌을 때 3초간 터치스크린을 길게 누르면 나타난다.

액티비티를 위한 메뉴를 정의하려면 onCreateOptionsMenu 메서드를 재정의한다

이메서드는 액티비티의 메뉴가 맨처음 나타날 때 호출된다. 위메서드는 menu객체를 변수로받는다.

Menu객체의 add메서드를 사용하여 채운다

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.view.ContextMenu;

**import** android.widget.AdapterView;

엑티브티 안에서 **onCreateOptionsMenu( 부른다.재정의**

@Override

**public** **boolean** **onCreateOptionsMenu**(Menu menu) {

**super**.**onCreateOptionsMenu**(menu);

// Create and add new menu items.

MenuItem itemAdd = menu.**add**(0, *ADD\_NEW\_TODO*, Menu.*NONE*, **R**.**string**.*add\_new*);

MenuItem itemRem = menu.**add**(0, *REMOVE\_TODO*, Menu.*NONE*, **R**.**string**.*remove*);

// Assign icons

itemAdd.**setIcon**(**R**.**drawable**.*add\_new\_item*);

itemRem.**setIcon**(**R**.**drawable**.*remove\_item*);

// Allocate shortcuts to each of them.

itemAdd.**setShortcut**('0', 'a');

itemRem.**setShortcut**('1', 'r');

**return** **true**;

}

설명:

menu.**add**(0, *ADD\_NEW\_TODO*, Menu.*NONE*, **R**.**string**.*add\_new*);

그릅값, 이벤트를위한 유일한값 Menu.*FIRST*; Static으로 선언되어있다. 이걸 사용후 ++시켜 유일값유지.

메뉴항목이 출력되는 순서를 정의하는 순서값 , 문자열이나 문자열리소스된 메뉴텍스트

마지막에 return true로 하자

체크박스로 바꿀땐 .setCheckable 메서드를 사용하자

menu.**add**(0, *ADD\_NEW\_TODO*, Menu.*NONE*, **R**.**string**.*add\_new*).setCheckable(true);

MenuItem itemRem = menu.**add**(1, *REMOVE\_TODO*, Menu.*NONE*, "체크박스").**setCheckable**(**true**);

MenuItem itemRem = menu.**add**(2, *REMOVE\_TODO*, Menu.*NONE*, "라디오1").**setCheckable**(**true**);

MenuItem itemRem = menu.**add**(2, *REMOVE\_TODO*, Menu.*NONE*, " 라디오2").**setCheckable**(**true**);

MenuItem itemRem = menu.**add**(2, *REMOVE\_TODO*, Menu.*NONE*, " 라디오3").**setCheckable**(**true**);

그릅이 동일하면 라디오로 바뀐다.

**단축키**

itemAdd.**setShortcut**('0', 'a');

itemRem.**setShortcut**('1', 'r');

**아이콘**

itemAdd.**setIcon**(**R**.**drawable**.*add\_new\_item*);

itemRem.**setIcon**(**R**.**drawable**.*remove\_item*);

간결한제목

menuItem.**setHeaderTitle**("선택해야할일");

**메뉴 이벤트리스너.146p**

안드로이드는 액티비티의 모든 메뉴 아이템 선택을 onOptionItemSelected 메서드라는 단일 이벤트 핸들러를 사용해서 다룬다 선택된 메뉴 항목은 MenuItem 매개변수로 이메서드에 전달된다

@Override

**public** **boolean** **onOptionsItemSelected**(MenuItem item) {

**Log**.*d*("=======Menu=", item.**getTitle**().**toString**());

**return** **super**.**onOptionsItemSelected**(item);

}

**동적으로 메뉴 항목 업데이트**

액티비티의 **onPrepareOptionsMenu**(Menu menu) 메서드를 재정의하면 메뉴가 표시될때마다 애플리케이션 상태에 기반해 메뉴를 수정할수 있다.

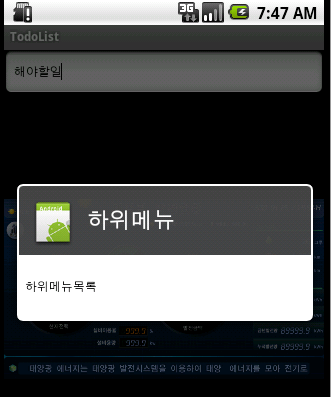
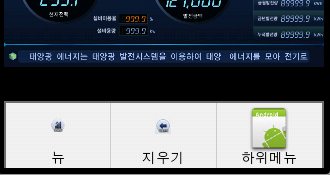
**하위메뉴**

SubMenu sub = menu.**addSubMenu**(0,0,Menu.*NONE*,"하위메뉴");

sub.**setHeaderIcon**(**R**.**drawable**.*icon*);

sub.**setIcon**(**R**.**drawable**.*icon*);

MenuItem submenuItem = sub.**add**(0,0,Menu.*NONE*,"하위메뉴목록");



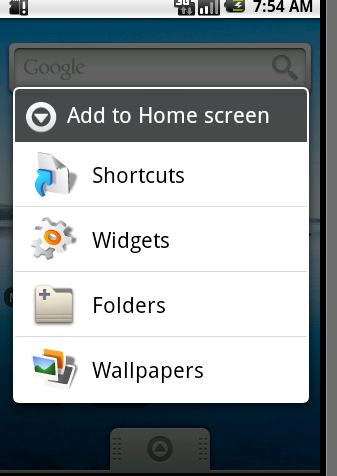
**/////////**

**컨텍스트 메뉴사용하기 (3초간 누르면..)**

현재 포커스를 가진 뷰에 으ㅢ해 맥락화되며 트랙볼이나 가운데 D패드 버튼또는 뷰를 3초간 누르면 나타난다

옵션이 2가지있다.

* 첫번째옵션 onCreateContextMenu 핸들러를 재정해 View 클래스에 대한 일반적인 컨텍스트 메뉴만듬

 @Override

**public** **void** **onCreateContextMenu**(ContextMenu menu, **View** v,

ContextMenuInfo menuInfo) {

**super**.**onCreateContextMenu**(menu, v, menuInfo);

menu.**setHeaderTitle**("선택해야할일");

menu.**add**(0,*REMOVE\_TODO*,Menu.*NONE*,"aaaaaaaaaaaaaaa");

}

* 두번째는 onCreate쪽에서 하는게있다 P149참고.

**선택다루기**

**onContextItemSelected**(MenuItem item) 메서드를 재정의하는 기법을 사용할수있다.

선택될때마다 호출딘다.

모든걸 다사용한 것.

**package** HelloPacage.main;

**import** java.util.ArrayList;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.util.Log;

**import** android.view.KeyEvent;

**import** android.view.View;

**import** android.view.ContextMenu.ContextMenuInfo;

**import** android.view.View.OnKeyListener;

**import** android.widget.ArrayAdapter;

**import** android.widget.EditText;

**import** android.widget.ListView;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.view.ContextMenu;

**import** android.view.SubMenu;

**import** android.widget.AdapterView;

**public** **class** **ToDoList** **extends** **Activity** {

/\*\* Called when the activity is first created. \*/

**static** **final** **private** **int** *ADD\_NEW\_TODO* = Menu.*FIRST*;

**static** **final** **private** **int** *REMOVE\_TODO* = Menu.*FIRST* + 1;

**private** **ArrayList**<**String**> todoItems;

**private** **ListView** myListView;

**private** **EditText** myEditText;

**private** **ArrayAdapter**<**String**> aa;

**private** **boolean** addingNew = **false**;

@Override

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

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

//뷰를 부풀린다

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

//UI래퍼런스를 가져온다

myListView = (**ListView**)**findViewById**(**R**.**id**.*myListView*);

myEditText = (**EditText**)**findViewById**(**R**.**id**.*myEditText*);

//어레이 리스트

todoItems = **new** **ArrayList**<**String**>();

**int** resId =**R**.**layout**.*todolist\_item*;

//어레이 어답터

aa =**new** **ArrayAdapter**<**String**>(**this**, resId,todoItems);

myListView.**setAdapter**(aa);

myEditText.**setOnKeyListener**(**new** **OnKeyListener**() {

@Override

**public** **boolean** **onKey**(**View** v, **int** keyCode, **KeyEvent** event) {

// **TODO** Auto-generated method stub

**if**(event.**getAction**() == **KeyEvent**.*ACTION\_DOWN*){

**if**(keyCode == **KeyEvent**.*KEYCODE\_DPAD\_CENTER*){

todoItems.**add**(0,myEditText.**getText**().**toString**());

aa.**notifyDataSetChanged**();

myEditText.**setText**("");

**return** **true**;

}

}

**return** **false**;

}

});

////menu...............................

**registerForContextMenu**(myListView); //메뉴를 받아들이도록 등록

}

@Override

**public** **boolean** **onCreateOptionsMenu**(Menu menu) {

**super**.**onCreateOptionsMenu**(menu);

// Create and add new menu items.

MenuItem itemAdd = menu.**add**(0, *ADD\_NEW\_TODO*, Menu.*NONE*, **R**.**string**.*add\_new*);

MenuItem itemRem = menu.**add**(0, *REMOVE\_TODO*, Menu.*NONE*, **R**.**string**.*remove*).**setCheckable**(**true**);

// MenuItem itemRem1 = menu.add(0, REMOVE\_TODO, Menu.NONE, R.string.remove);

// MenuItem itemRem2 = menu.add(0, REMOVE\_TODO, Menu.NONE, R.string.remove);

// MenuItem itemRem3 = menu.add(0, REMOVE\_TODO, Menu.NONE, R.string.remove);

// MenuItem itemRem4 = menu.add(0, REMOVE\_TODO, Menu.NONE, R.string.remove);

// MenuItem itemRem5 = menu.add(0, REMOVE\_TODO, Menu.NONE, R.string.remove);

// MenuItem itemRem6 = menu.add(0, REMOVE\_TODO, Menu.NONE, R.string.remove);

// MenuItem itemRem7 = menu.add(0, REMOVE\_TODO, Menu.NONE, R.string.remove);

// Assign icons

itemAdd.**setIcon**(**R**.**drawable**.*add\_new\_item*);

itemRem.**setIcon**(**R**.**drawable**.*remove\_item*);

// itemRem1.setIcon(R.drawable.remove\_item);

// itemRem2.setIcon(R.drawable.remove\_item);

// itemRem3.setIcon(R.drawable.remove\_item);

// itemRem4.setIcon(R.drawable.remove\_item);

// itemRem5.setIcon(R.drawable.remove\_item);

// itemRem6.setIcon(R.drawable.remove\_item);

// itemRem7.setIcon(R.drawable.remove\_item);

// SubMenu sub = menu.addSubMenu(0,0,Menu.NONE,"하위메뉴");

// sub.setHeaderIcon(R.drawable.icon);

// sub.setIcon(R.drawable.icon);

// MenuItem submenuItem = sub.add(0,0,Menu.NONE,"하위메뉴목록");

// Allocate shortcuts to each of them.

itemAdd.**setShortcut**('0', 'a');

itemRem.**setShortcut**('1', 'r');

**return** **true**;

}

//컨텍스트 메뉴 ......

@Override

**public** **void** **onCreateContextMenu**(ContextMenu menu, **View** v,

ContextMenuInfo menuInfo) {

**super**.**onCreateContextMenu**(menu, v, menuInfo);

menu.**setHeaderTitle**("선택해야할일");

menu.**add**(0,*REMOVE\_TODO*,Menu.*NONE*,**R**.**string**.*remove*);

}

//동적인..상황에맞게 할려고

@Override

**public** **boolean** **onPrepareOptionsMenu**(Menu menu) {

**super**.**onPrepareOptionsMenu**(menu);

**int** idx = myListView.**getSelectedItemPosition**();

**String** removeTitle = **getString**(addingNew ? **R**.**string**.*cancel* : **R**.**string**.*remove*);

MenuItem removeItem = menu.**findItem**(*REMOVE\_TODO*);

removeItem.**setTitle**(removeTitle);

removeItem.**setVisible**(addingNew || idx > -1);

**return** **true**;

}

/\*\* Process the options menu item selection \*/

@Override

**public** **boolean** **onOptionsItemSelected**(MenuItem item) {

**super**.**onOptionsItemSelected**(item);

//현재 하일라이트 된항목찾을수있다

**int** index = myListView.**getSelectedItemPosition**();

**switch** (item.**getItemId**()) {

**case** (*REMOVE\_TODO*): {

**if** (addingNew) {

// If adding a new item, cancel this action

**cancelAdd**();

}

**else** {

// If not adding a new item, remove the currently selected one.

**removeItem**(index);

}

**return** **true**;

}

**case** (*ADD\_NEW\_TODO*): {

// Add a new item

**addNewItem**();

**return** **true**;

}

}

**return** **false**;

}

/\* \*/

@Override

**public** **boolean** **onContextItemSelected**(MenuItem item) {

**super**.**onContextItemSelected**(item);

**switch** (item.**getItemId**()) {

**case** (*REMOVE\_TODO*): {

**AdapterView**.**AdapterContextMenuInfo** menuInfo = (**AdapterView**.**AdapterContextMenuInfo**)item.**getMenuInfo**();

**int** index = menuInfo.position;

**removeItem**(index);

**return** **true**;

}

}

**return** **false**;

}

/\*\* Cancel the action of adding a new todo list item \*/

**private** **void** **cancelAdd**() {

addingNew = **false**;

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

}

/\*\* Present the text entry box for adding a new todo item \*/

**private** **void** **addNewItem**() {

addingNew = **true**;

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

myEditText.**requestFocus**();

}

/\*\* Remove the specified item from the todo list \*/

**private** **void** **removeItem**(**int** \_index) {

todoItems.**remove**(\_index);

aa.**notifyDataSetChanged**();

}

}

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"* android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

<EditText android:id=*"@+id/myEditText"* android:text=*"해야할일"* android:layout\_width=*"fill\_parent"* android:layout\_height=*"wrap\_content"*

android:visibility =*"gone"*></EditText>

<ListView android:id=*"@+id/myListView"* android:layout\_height=*"wrap\_content"* android:layout\_width=*"fill\_parent"*></ListView>

<ImageView android:id=*"@+id/ImageView01"* android:layout\_width=*"wrap\_content"* android:layout\_height=*"wrap\_content"* android:src=*"@drawable/sa"*></ImageView>

</LinearLayout>

**인텐트**

어떤 액션이 수행되어야 한다는 여러분의 의향을 선언할 수있도록 해주는 메시지 전달 메커니즘으로 사용된다.

* **인텐트는 가장 많이 사용하는곳은 엑티브티를 명시적으로/암시적으로 불러들이는 기능이다.**
* **인텐트는 시스템 전역에 메시지를 방송하는데에도 사용될수 있다.**
  + 브로드캐스트 수신 자를 등록해 이들 브로드 캐스트 인텐트에 귀기울이고 반응할수 있다.
  + 브로드캐스트 인텐트를 사용해 인터넷 연결상태나 배터리 충전 수준의 변화 같은 시스템 이벤트를 린다.
  + sms관리자 같은 네이티브 안드로이드 애플리케이션은 간단한 특정 브로드캐스트 인텐트(걸려오오는 전화 나 SMS 메시지 수신 같은) 에 귀기울이는 컴포넌트를 등록하고 적절희 반응한다
* **명시적으로 액티비트를 실행시킬수있다**
* **런타임 바인딩으로 시작하는 애티비트를 실행시킬수있다.**

**인텐트를 이용해 액티비트 뛰우기**

* 인덴트 이용해서 애플리케이션 컴포넌트간의 상호작용가능
* 인텐트는 애플리케이션 내에있는 액티비트들 간의 시작, 중지 , 전이 에 사용된다
* 새로운 애티브티를 오픈하려면 **startActivity(myIntent);** 전달해 호출한다.
* 인덴트는 오픈할 클래스를 명시적으로 지정하거나 수행되어야 하는 대상인 액션을 포함할수 있다. 후자인경우 런타임은 인텐트 확인 이라는 과정을 통해 오픈할 액티비트를 고를것이다.
  + startActivity()메서드는 인텐트와 갖아 필적한 애티비트 하나를 찾아 시작시킨다.
  + startActivity()를 사용할경우 새로 띄어진 액티비티가 종료할 때 아무런 통지도 받지 못할것이다 오픈된 폼으로부터 피드백을 추적하려면 startActivityForResult() 메서드를 이용하자.

**명시적으로 액티비트 시작시키기**

* 어플은 화면(애티비트) 들로 구성되어있으며 이들 화면은 반드시 애플리케이션 매니페스트 안에 포함되어야 한다
* 이들을 연결하기위해서 오픈 액티비트르 명시적으로 지정할수 있다.

**Intent intent = new Intent(MyActivity.this,MyOtherActivity.class);**

**startActivity(intent);**

위코드가 실행이되면 MyActivity.this 가 생성되어 화면에 보이지고 활성상태가 되며 액티비트 스택 맨위로 올라갈것이다.

암시적 인텐트와 늦은 런타임 바인딩

If(somethingWeird && itDontLookGood){

Intent intent = new Intent (Intent.ACTION\_DIAL, Uri.parse(<tel:444-4444)>);

startActivity(intent);

전화걸기를 요청하는 암시적 인텐트를 사용할수 있다.

**Linkify**

지정된 REGEX 패턴과 일치하는 텍스트는 그일치한 텍스트를 대상 URI로 사용해 암시적으로 startActivity(new Intent(Intent.ACTION\_VIEW,uri))를 호출해 클릭가능 한 링크 만들수있다.

**네임티브 링크타입**

….164 많이있으니보도록.. 다른것들도.

**커스텀 링크 문자열 만들기**

**…**

**매치필터 사용하기**

**…**

**변형필터 사용하기**

**…**

**액티브티 결과 리턴하기**

자식 액티브티는 종류시 부모의에 연결된 서브 액티비티로 시작시킬수있다.

즉종류시 부모의 액티브티 이벤트 핸들러를 호출한다….

**서브 액티브티 뛰우기**

센서관리자.

**tring** service\_name = **Context**.*SENSOR\_SERVICE*;

**SensorManager** sensormanager =(**SensorManager**)**getSystemService**(service\_name);

센서 불러온다

매너저로 접근가능하다

정확도 구분

|  |  |
| --- | --- |
| SensorManager.SENSOR\_STATUS\_ACCURACY\_HIGH | 좋아. |
| SensorManager.SENSOR\_STATUS\_ACCURACY\_LOW | 낮아 |
| SensorManager.SENSOR\_STATUS\_ACCURACY\_MEDIUM | 중간. |
| SensorManager.SENSOR\_STATUS\_ACCURACY\_UNRELIABLE | 믿지마 |

센서구분

|  |  |
| --- | --- |
| SensorManager.SENSOR\_ACCELEROMETER | 현재 가속도를 세축을 따라 제곱초당 미터(m/s2)로 리턴하는 가속센서 |
| SensorManager.SENSOR\_ORIENTATION | 세축에 대한 현재 방향을 도 단위로 리턴하는 방향센서 |
| SensorManager.SENSOR\_LIGHT | 조도를 럭스로 기술한 단일 값 리턴 |
| SensorManager.SENSOR\_MAGNETIC\_FIELD | 세축을 따라 마이크로테슬라 로 측정하는데 사용되는 센서 |
| SensorManager.SENSOR\_PROXIMITY | 장치와 대한 물체간의 거리를 미터로 기술 한 단일값 리턴 근접센서 |
| SensorManager.SENSOR\_TEMPERATURE | 주변온도 도시로 리턴 |

업데이트 주기 구분

|  |  |
| --- | --- |
| SensorManager.SENSOR\_DELAY\_FASTEST | 가장빨러 |
| SensorManager.SENSOR\_DELAY\_GAME | 게임에서 적절한정도 |
| SensorManager.SENSOR\_DELAY\_NOMAL | 기본 |
| SensorManager.SENSOR\_DELAY\_UI | 유아이에서 적절한정도 |

센서 구분되서 불러올 때

**Sensor** sensor = sensormanager.**getDefaultSensor**(**Sensor**.*TYPE\_ORIENTATION*);

<http://developer.android.com/reference/android/hardware/Sensor.html>

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Constants** | | | | | | | | | | | |
| int | [TYPE\_ACCELEROMETER](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_ACCELEROMETER) | A constant describing an accelerometer sensor type. |  |  |  |  |  |  |  |  |  |
| int | [TYPE\_ALL](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_ALL) | A constant describing all sensor types. |  |  |  |  |  |  |  |  |  |
| int | [TYPE\_GYROSCOPE](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_GYROSCOPE) | A constant describing a gyroscope sensor type |  |  |  |  |  |  |  |  |  |
| int | [TYPE\_LIGHT](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_LIGHT) | A constant describing an light sensor type. |  |  |  |  |  |  |  |  |  |
| int | [TYPE\_MAGNETIC\_FIELD](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_MAGNETIC_FIELD) | A constant describing a magnetic field sensor type. |  |  |  |  |  |  |  |  |  |
| int | [TYPE\_ORIENTATION](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_ORIENTATION) | This constant is deprecated. use[*SensorManager.getOrientation()*](http://developer.android.com/reference/android/hardware/SensorManager.html#getOrientation(float[], float[]))instead. |  |  |  |  |  |  |  |  |  |
| int | [TYPE\_PRESSURE](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_PRESSURE) | A constant describing a pressure sensor type |  |  |  |  |  |  |  |  |  |
| int | [TYPE\_PROXIMITY](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_PROXIMITY) | A constant describing an proximity sensor type. |  |  |  |  |  |  |  |  |  |
| int | [TYPE\_TEMPERATURE](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_TEMPERATURE) | A constant describing a temperature sensor type |  |  |  |  |  |  |  |  |  |

#### [안드로이드 가속도 센서 좌표계 및 값 정리 (accelerometer sensor of android)](http://www.androidpub.com/1305)

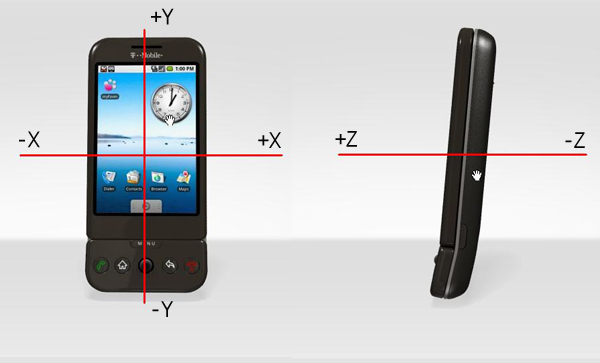
id: 회색회색

<http://www.androidpub.com/1305>

**2008.12.18** 01:33:09

7553

[**앱개발**](http://www.androidpub.com/?mid=android_dev_info&category=108970)

회색입니다. 안드로이드 센서(Android Sensor)의 종류(Orientation, Accelorometer, Magnetic field)와 좌표계 그리고 값에 대해 간단히 정리했습니다. 가속도 센서를 이용하시려고 하는 분은 참고하세요.   
   
**좌표 시스템**  
센서는 X, Y, Z 의 좌표에 대해서 각각에 해당하는 값을 배열로 받아와서 처리할 수 있게 되어있습니다.   
  
좌표 축 정의  
X축 : 화면에 수평축(portrait 모드에서 짧은 에지를 landscape 모드에서 긴 에지)을 나타내며 오른쪽을 가리킨다.  
Y축 : 화면에 수직축을 나타내며 화면 위쪽을 가리킨다.(원점은 왼쪽 밑 코너)  
Z축 : 단말이 화면을 위로 해서 테이블에 올려져 있다고 생각했을 때 하늘을 가리킨다.  
  
  
  
\* Orientation (Portrait, Landscape)이 바뀌었을 때 좌표값 역시 바뀐다는 것을 기억해야 합니다. 바뀌지 않은 값을 사용하기 위해서는 배열의 3, 4, 5번째 값을 가져올 수 있습니다.  
  
  
  
  
**Orientation**  
SENSOR\_ORIENTATION (방향 센서)  
각 배열의 값은 각도를 나타냅니다.    
단말기 화면이 하늘을 향한 상태로 테이블 위에 수평으로 놓여있는 상태를 기준으로 각 축을 중심으로 회전시키는 것을 생각하면 됩니다.  
  
values[0] : Z 축을 중심으로 회전 (0 <= azimuth < 360)   
                    0  = 북, 90 = 동, 180 = 남, 270 = 서  
  
values[1] : X축을 중심으로 회전 (-180 <= pitch <= 180)   
                   Z축이 Y축 방향으로 향하면 0보다 큰값  
                   화면이 하늘을 향하고 테이블위에 수평으로 놓여있는  상태 0, 화면이 아래를 향하면 -180 or 180,    
                   똑바로 세우면 -90, 거꾸로 세우면 +90  
                     
values[2] : Y축을 중심으로 회전 (-90 <= roll <= 90)   
                   Z축이 X축 방향으로 향하면 0보다 큰값  
   
  
**Accelerometer**  
SENSOR\_ACCELOROMETER (가속도 센서)  
각 배열의 값은 (m/s2) 단위로 되어있으며, 접촉힘(Contact Force)을 측정합니다.   
  
values[0] : X축에 적용되는 힘  
values[1] : Y축에 적용되는 힘  
values[2] : Z축에 적용되는 힘  
  
예를 들어

 단말의 왼쪽 측면을 오른쪽 방향으로 눌렀을때 X 가속도 값은 음수를 나타냅니다.

 단말에 테이블위에 평평하게 놓여있을때 -STANDARD\_GRAVITY 값, 즉 -9.8 (m/s2) 을 가집니다. 단말이 테이블에서 중력에 대한 반작용으로 적용되는 힘을 나타냅니다.   
  
**Mgnetic field**  
SENSOR\_MAGNETIC\_FIELD (자기장 센서)  
모든 값은 micro-Tesla (uT) 단위로 되어있으며, X, Y, -Z 축 주변 자기장을 측정합니다.  
  
\* Z축의 값이 바뀐 것 주의  
  
관련링크 : <http://code.google.com/android/reference/android/hardware/SensorListener.html>