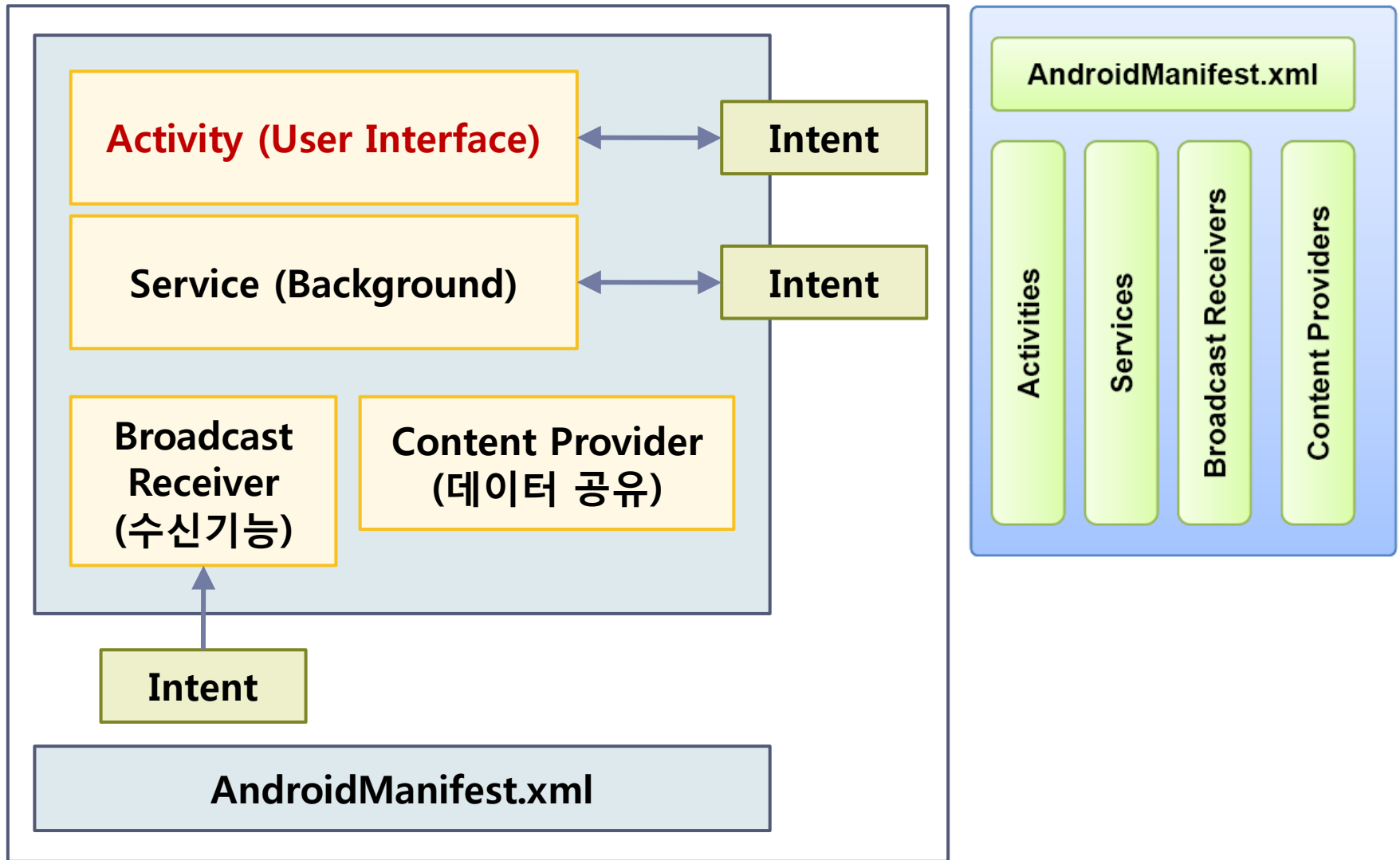




Android 구성 요소

Android Application 구성 요소



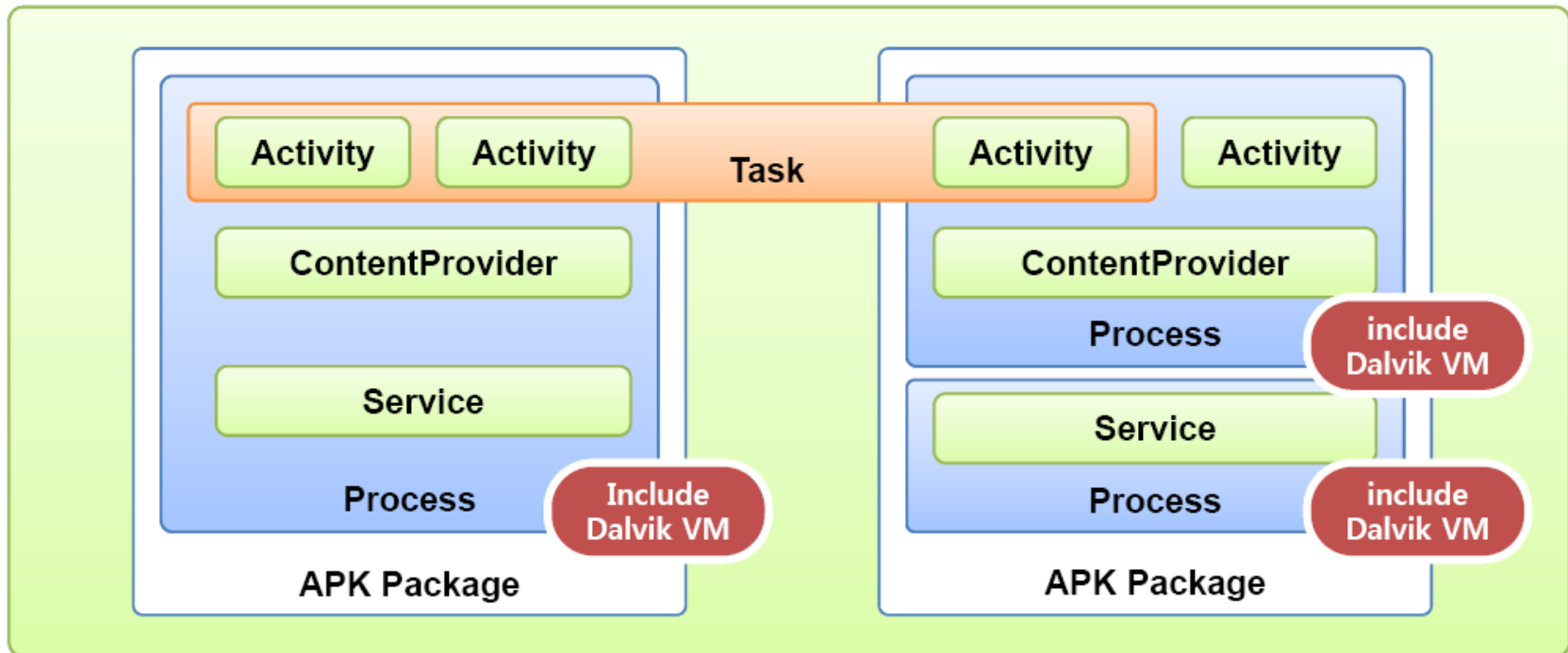
Android Application 구성

Application Building Block

- AndroidManifest.xml
- Activity [User Interaction]
- ContentProvider [Data Provider]
- Service [Service Provider]
- BroadcastReceiver

Intent : Component Activation Method

- Explicit Method : Call Class
- Implicit Method : IntentFilter
 - Action, Data, Category
 - Declared at AndroidManifest.xml

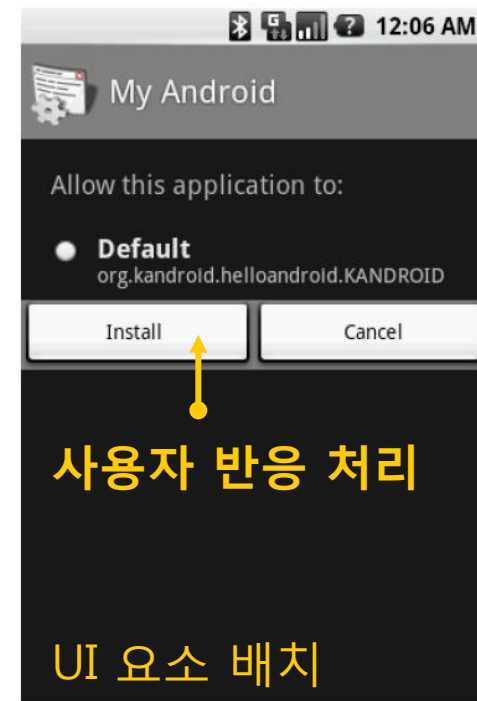
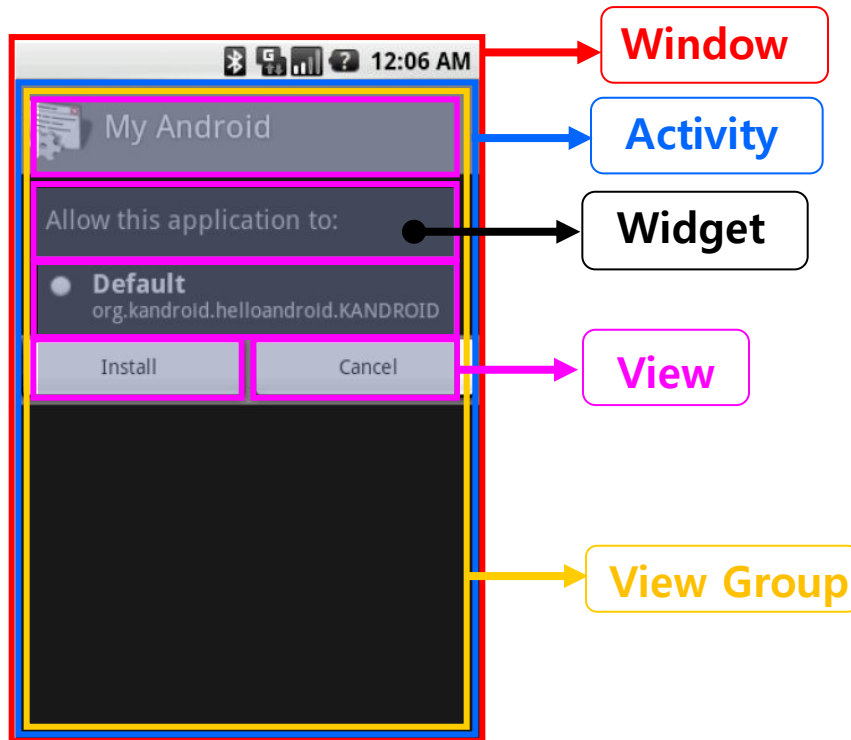




Activity & UI Elements - View, View Group, Widget

Activity

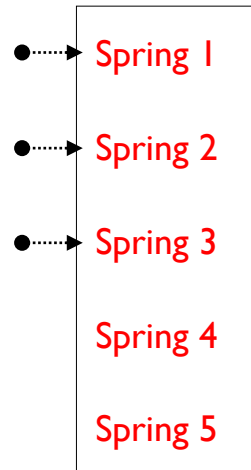
- 애플리케이션에서 하나의 화면을 일컫는 말
- UI element를 화면에 표시하고, 시스템이나 사용자의 반응을 처리할 수 있음
- 윈도우의 창의 개념과 유사하지만, Status Bar 영역이 포함되지 않으며 크기 변경, 화면 최대/최소 등을 할 수 없음



AdapterView - ListView



Cursor



데이터 처리 용이

List View



Spring 1

Spring 2

Spring 3

Spring 4

Spring 5

ListAdapter

UI Elements

Activity – UI 표시 및 사용자 이벤트 처리, 안드로이드 애플리케이션의 가장 기본적인 구성 단위로 한 화면을 나타내지만 그 자체로는 아무 것도 보여줄 수 없고, Activity에 View와 ViewGroup 클래스를 사용해야 비로소 화면에 무엇인가를 표시할 수 있음

View – android.view 패키지의 클래스, UI를 표현하는 단위, 사각형 영역으로 상호 작용 이벤트를 처리함

ViewGroup – 직접적으로 보이지 않으며 다른 View를 담는 컨테이너 역할을 함, 여러 개의 뷰를 유기적으로 모아 놓은 것 (FrameLayout, RelativeLayout, LinearLayout, ListView, Spinner ...)

Widget – android.widget 패키지의 클래스, 직접적으로 보이며, 사용자로부터 입력되는 이벤트를 처리하는 기능을 가지고 있고, custom widget을 만들어 사용하기도 함 (TextView, EditText, Button, ImageView, CheckBox, RadioButton...)

AdapterView – ViewGroup으로부터 파생되었으며, 배치만 담당하는 레이아웃과 달리 사용자와 상호 작용도 처리 하므로 터치나 키패드로 항목 선택이 가능함. 표시할 항목 데이터를 어댑터(Adapter) 객체로부터 공급 받으며, 항목의 개수는 무한대이고, 실행 중 목록이 바뀔 수 있고, 데이터의 원본(자료구조, XML, DB)도 다양함. 어댑터 뷰의 대표 위젯으로 리스트 뷰가 있음

Adapter(데이터 관리) + AdapterView (어댑터가 전달한 데이터를 화면에 표시)



View Group

파생
순서

java.lang.Object

android.view.View

android.view.ViewGroup

android.widget.AdapterView

android.widget.AbsListView

android.widget.ListView

java.lang.Object

android.view.View

android.view.ViewGroup

android.widget.RelativeLayout



widget

파생
순서

java.lang.Object

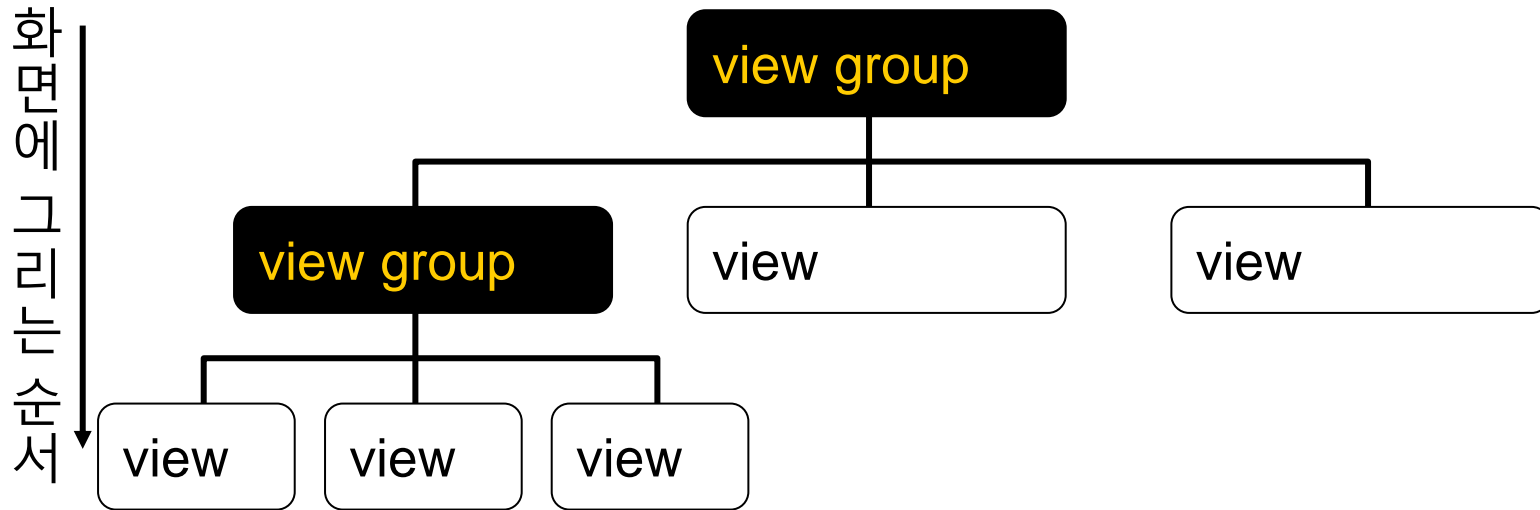
adnroid.view.View

android.widget.TextView

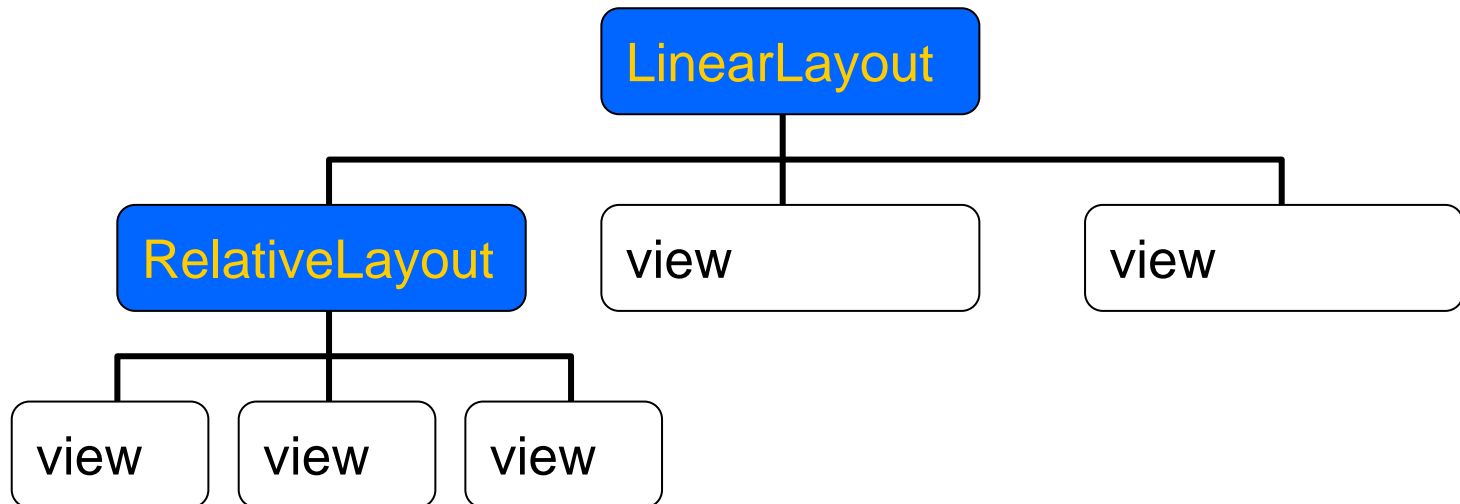
android.widget.Button



View와 View Group



view group은 아래와 같이 구성됨





Widget의 사용

View Component 사용

1. XML 형식으로 사용 (권장 방법, 정적인 요소와 동적인 요소의 분리관리)

```
<TextView  
    android:id="@+id=name"  
    android:layout_width="fill_parent"  
    android:layout_height="100dp"  
    android:textSize="20sp"  
    android:text="@string/hello" />
```

XML file **main.xml**

```
setContentView(R.layout.main);  
TextView t = (TextView) findViewById(R.id.name);
```

java file

2. Method 형식으로 사용 (Android에서는 비추천, Java에서 사용하는 방식)

```
TextView t = new TextView(this); // 메모리에 만들기
```

```
t.setText("Finish");
```

```
t.setWidth(30);
```

```
t.setHeight(20);
```

```
setContentView(t); // 화면에 보여주는 것
```

View Method는 대응하는 XML attribute가 있음
(**android.widget.TextView** 참조)

View Component 주요 속성

id : 식별자, 뷰의 이름

android:id="@+id/btn_finish"

android:id="@id/btn_finish"

android:id="@android:id/empty"

android:text="@string/hello"

android:id : Resource의 종류

@ : resource(R.java)에 정의 또는 참조함을 의미

+ : 새로 정의함

+가 없으면 이미 만들어진 것을 사용 (참조)

findViewById(R.id.name)로 뷰를 찾을 수 있음

android package, R class

R

R.anim

R.animator

R.array

R.attr

R.bool

R.color

R.dimen

R.drawable

R.fraction

R.id

R.integer

R.interpolator

R.layout

R.menu

R.mipmap

R.plurals

R.raw

R.string

R.style

View Component 속성

layout_width: 컴포넌트의 폭

layout_height: 컴포넌트의 높이

- : fill_parent - 부모 영역 전부다 사용

- : wrap_content - 해당 view component의 최소 크기

- : 숫자 - 숫자 크기에 따라 (사용 단위 - px(해상도 영향을 받게 됨), mm, in, pt, dp(해상도 비례 크기), sp)

android:background : 색상 (#RRGGBB, #AARRGGBB)

android:padding : 배치할 때 사용하는 attribute, 4곳의 padding을 한꺼번에 부여할 때 사용 (따로 부여-paddingLeft, paddingTop, paddingRight, paddingBottom)

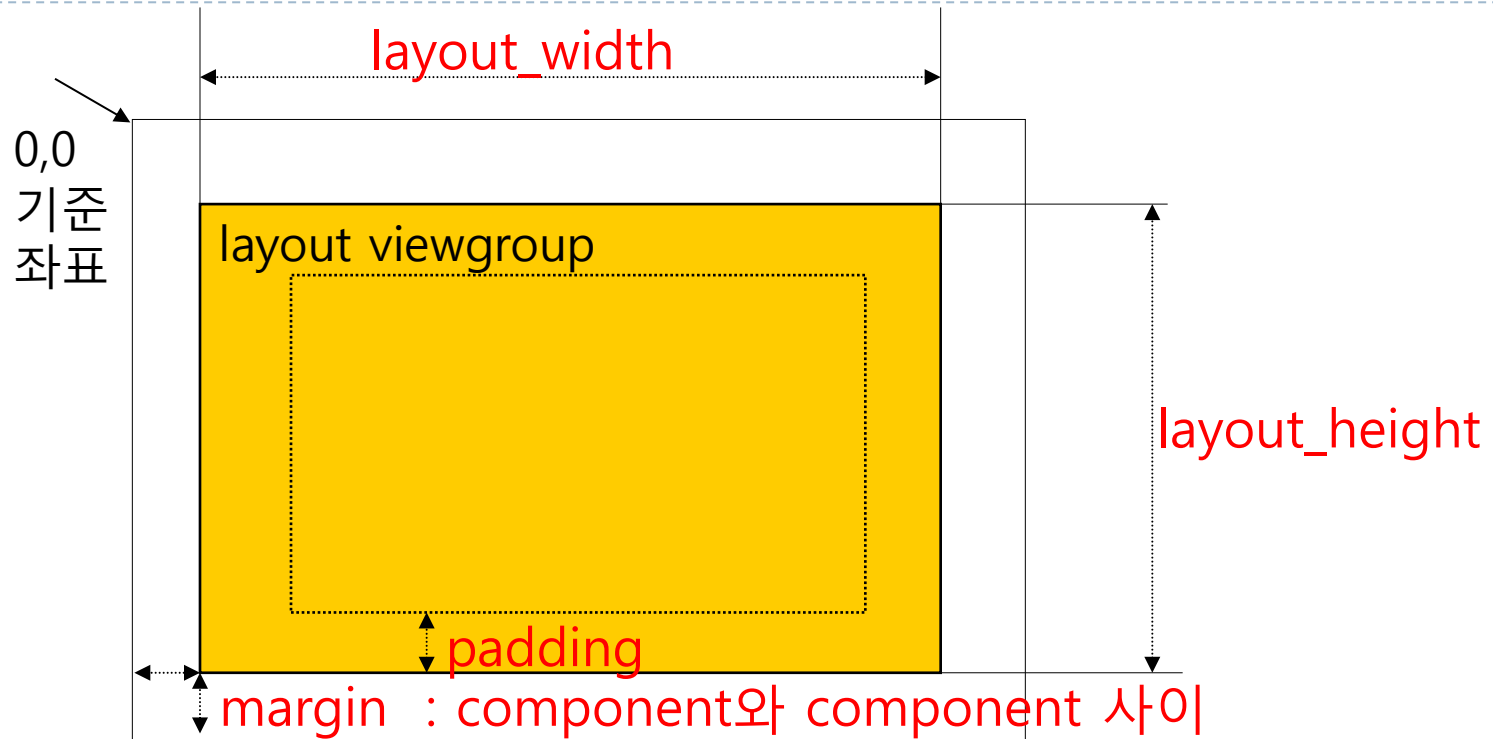
android:visibility : 보였다가 보이지 않았다가 하는 동작에 사용 (visible, invisible, gone)

android:clickable : 마우스 클릭 이벤트를 받을 것인지 롱 클릭 이벤트를 받을지 지정 (true/false)

focusable: 키보드 포커스를 받을 수 있을지 지정 (default-false)



Layout의 좌표와 용어



문자열을 표시하는 라벨 view

Hello

wrap_content

컨텐츠 표시하기 충분한 크기

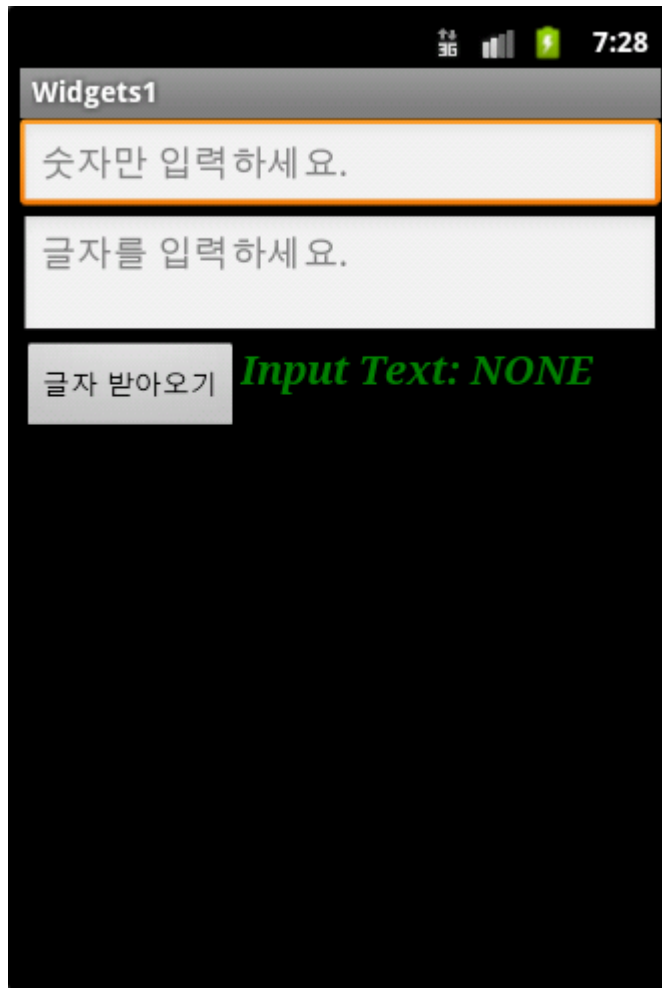
문자열을 표시하는 라벨 view

Hello

fill_parent









부모 객체와의 패딩(여백)을 제외한
나머지 공간을 차지

Widget1 (EditText, Button, Spinner, TextView)



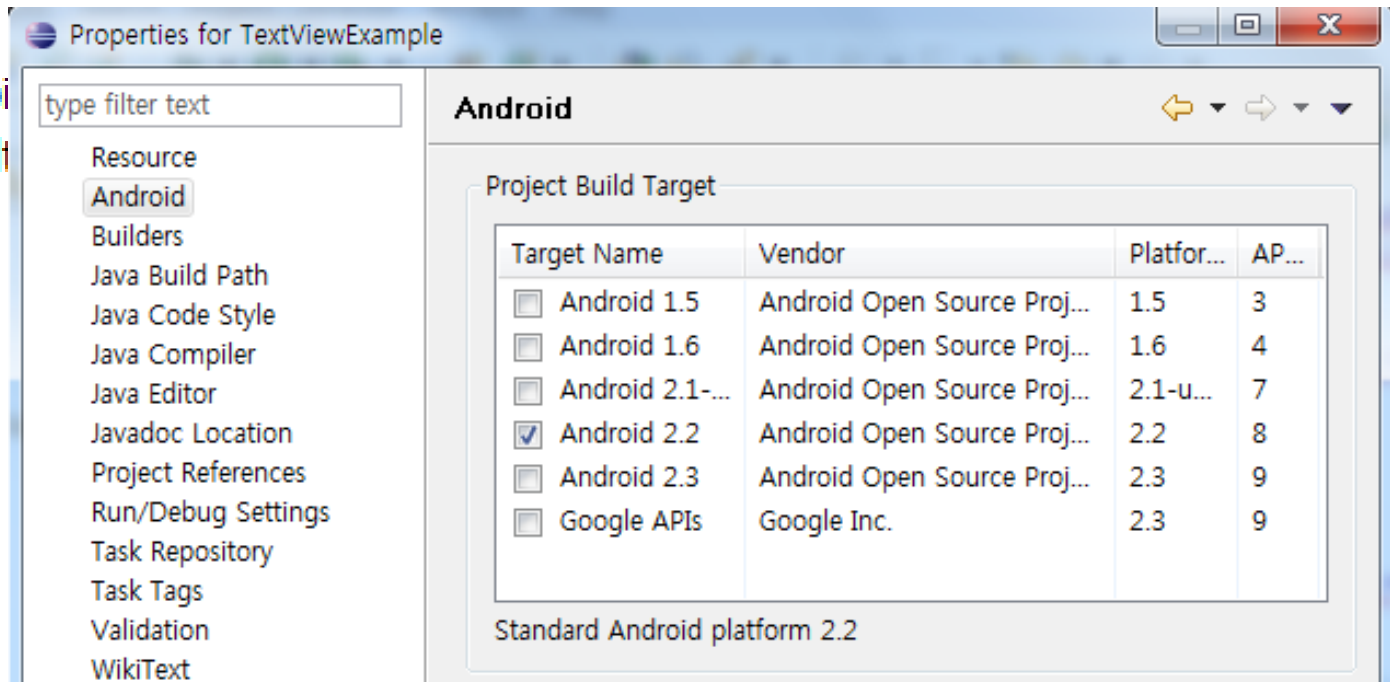
- Widget1, Widget2, Widget3 파일을 다운로드 받아 Import 하여 사용
- Eclipse에서 Import 하면 Android 2.3.3으로 작성이 되어 있으며, 그대로 사용함

Import 오류 대처 방법

- ▼  TextViewExample
 - ▶  src
 - ▶  gen [Generated Java Files]
 - ▶  Android 2.1-update1
 - ▶  assets
 - ▶  res
 - ▶  AndroidManifest.xml
 - ▶  default.properties

<<ADK 버전이 다른 경우>>

- [Properties]창에서
Android – Build Target을 조정함



Widget1 - string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="default_text">Input Text: NONE</string>
    <string name="app_name">Widgets1</string>
</resources>
```



Widget1 - main.xml

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

```
<LinearLayout xmlns:android="http://~ ~"  
    android:orientation="vertical"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
>
```

View Group에서 설명

```
<EditText android:layout_height="wrap_content"  
    android:inputType="number"  
    android:layout_width="fill_parent"  
    android:hint="숫자만 입력하세요."/>
```

inputType: none, text,
number, phone,
datetime ... etc

```
<EditText android:layout_height="wrap_content"  
    android:id="@+id/editText"  
    android:hint="글자를 입력하세요."  
    android:layout_width="fill_parent"  
    android:lines="2"  
    android:gravity="top"/>
```

lines를 지정하지 않으면, 자동
으로 라인 증가
inputType을 지정하면 자동
증가 안됨

Widget1 - main.xml

```
<RelativeLayout xmlns:android="http..."
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="글자 받아오기"/>
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textview"
        android:layout_toRightOf="@id/button"
        android:textColor="#8000FF00"
        android:textSize="20sp"
        android:typeface="serif"
        android:textStyle="bold/italic"
        android:text="@string/default_text"/>
```

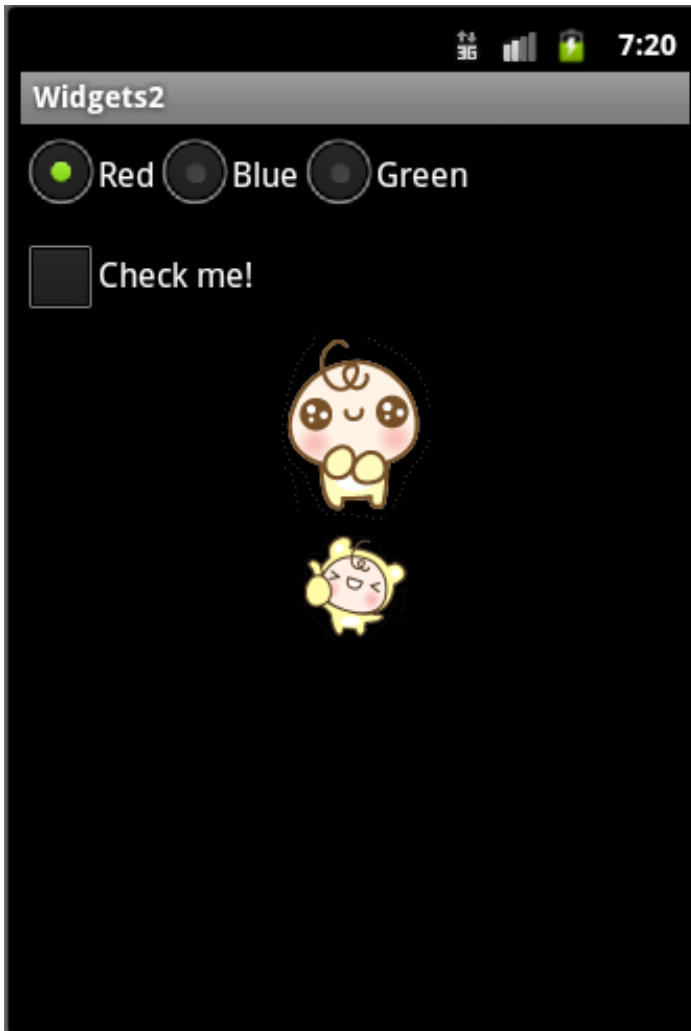
```
</RelativeLayout>
```

```
</LinearLayout>
```

*** 주의사항 ***
typeface (모두 소문자)



Widget2 (RadioButton, CheckBox, ImageView)

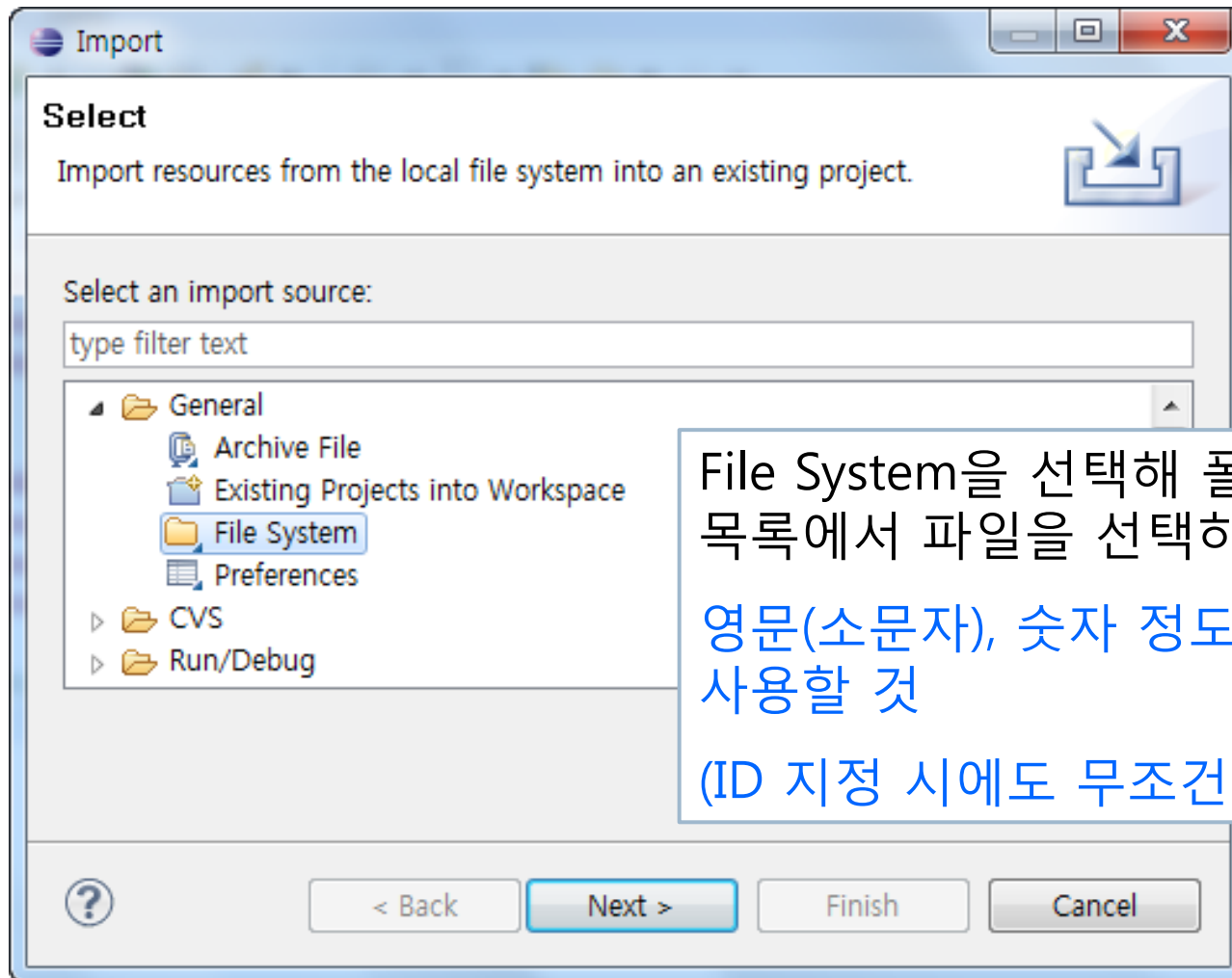


RadioButton : RadioGroup에서 1개만 선택

CheckBox : 개별 동작으로 선택/비선택

ImageView: 단순히 이미지를 출력하는 기능으로 클릭하여 동작하게 하려는 용도로 사용하고자 할 때는 ImageButton 을 사용해야 함

Image Import



File System을 선택해 폴더를 선택 후,
목록에서 파일을 선택하여 import함
영문(소문자), 숫자 정도(첫 글자 안됨)만
사용할 것
(ID 지정 시에도 무조건 소문자로 지정)

Widget2 - main.xml

```
<RadioGroup android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:id="@+id/radioGroup"
    android:orientation="horizontal"
    android:checkedButton="@+id/radio2">
```

orientation: 배열 방향
checkedButton: 선택 버튼

```
<RadioButton android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/radio1"
    android:text="Red"
    android:checked="true"/>
```

checked: 선택 버튼
checked가 checkedButton
보다 우선함

```
<RadioButton android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Blue"
    android:id="@+id/radio2"/>
</RadioGroup>
```



Widget2 - main.xml

```
<CheckBox android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/checkbox"
    android:text="Check me!"/>
```

```
<ImageView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/smile"/>
```

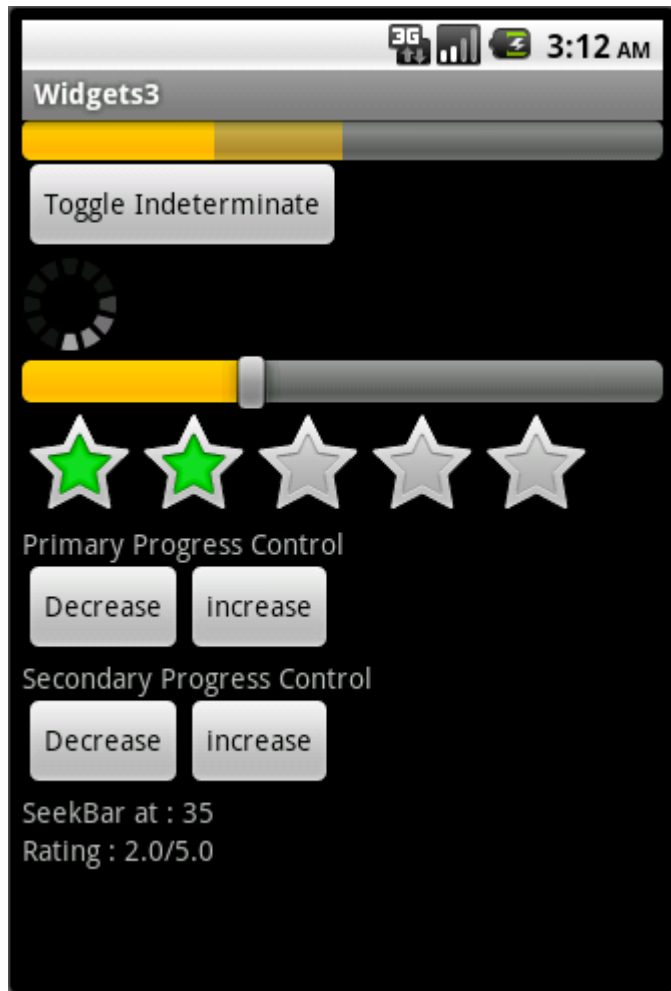
```
<ImageView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/jjang"
    android:maxHeight="50px"
    android:maxWidth="50px"
    android:adjustViewBounds="true" />
```

src : 이미지 지정
이미지 이름은 소문자, 숫자로 이름이 되어 있으면 안됨
maxHeight: 최대 높이
maxWidth: 최대 너비
축소된 이미지로 표시가능
adjustViewBounds를 true로
해야 축소됨

이미지를 출력하기만 함
동작을 위해서는
ImageButton 사용해야 함



Widget3 (Progress Bar, SeekBar, RatingBar)



Widget3 - main.xml

```
<ProgressBar  
    android:layout_height="wrap_content"  
    style="?android:attr/progressBarStyleHorizontal"  
    android:layout_width="fill_parent"  
    android:max="100"  
    android:progress="30"  
    android:secondaryProgress="50"  
    android:id="@+id/progressHorizontal"/>
```

max : 최대
progress : 진행표시
secondaryProgress: 흐리게
표시되는 진행표시

```
<SeekBar android:layout_height="wrap_content"  
    android:layout_width="fill_parent"  
    android:id="@+id/seekBar"/>
```

```
<RatingBar android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/ratingBar"/>
```



Date Picker, Time Picker



Java의 **Calendar Class**를 이용하며 시간에 대한 정보를 얻어와서 설정을 할 수 있음

Widget4 - main.xml

```
<DatePicker android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/datePicker"/>
```

```
<TextView android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/dateText"  
    android:text="[DateText]"/>
```

```
<TimePicker android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/timePicker"/>
```

```
<TextView android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="[TimeText]"  
    android:id="@+id/timeText"/>
```

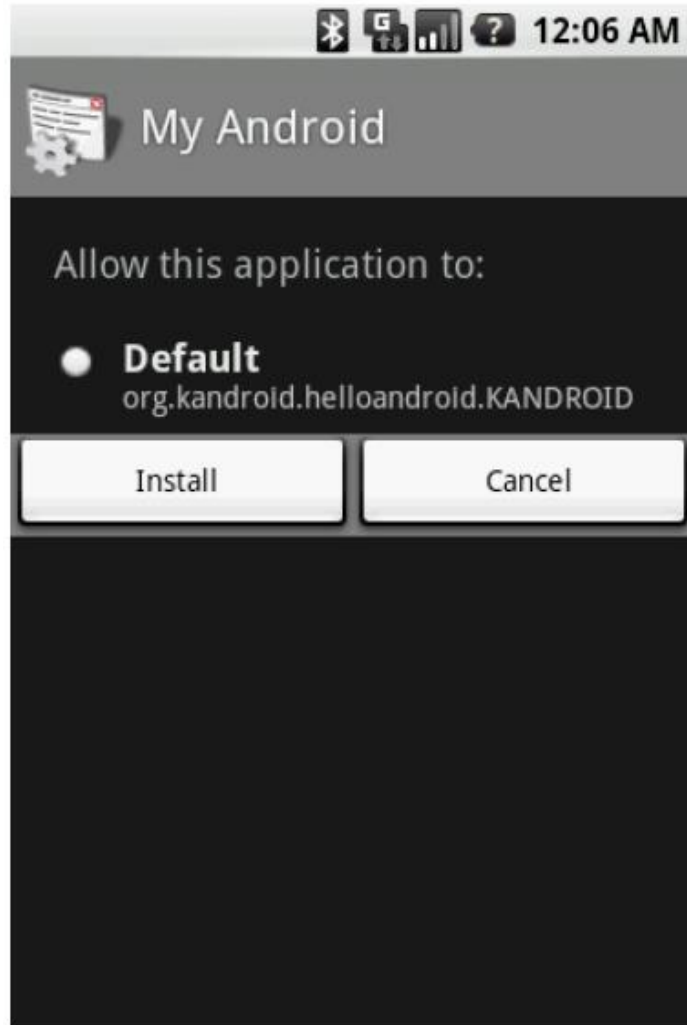




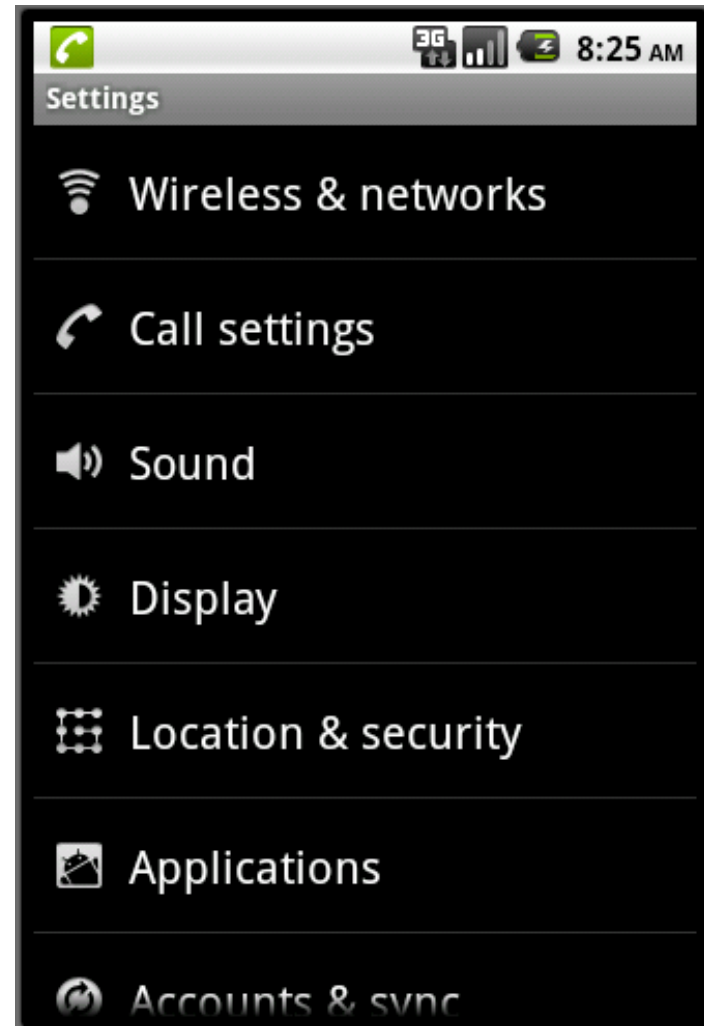
View Group의 사용 (Layout)

View Group

LinearLayout



AdapterView - ListView



View Component - Layout

Layout의 기능 : 다른 View Component를 배치하고 관리하는 기능

다음 세 가지를 가장 많이 사용, LinearLayout은 default로 사용함

LinearLayout	가장 많이 사용하는 Layout으로 단 방향(가로/세로)으로만 배치
RelativeLayout	Component 간의 관계에 따라 배치, 보다 세밀한 배치가 가능함
TableLayout	행/열 구조로 배치

다음 두 가지는 많이 사용되지 않는 방식

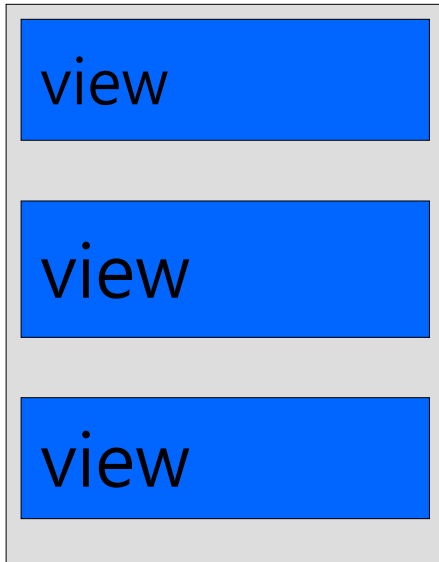
AbsoluteLayout	좌표를 지정하여 배치, 해상도 변화에 유연성 없음
FrameLayout	기준좌표가 좌측 최상단 모서리로 고정됨. (여러 개 Component를 그리는 경우 Component가 중첩됨)



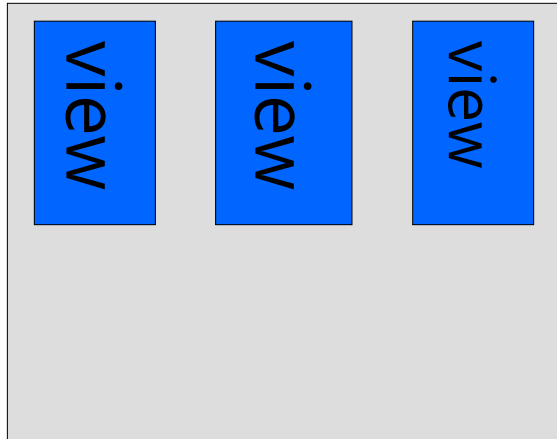
LinearLayout

모든 하위 구성 요소를 수직 또는 수평 형태 중 하나로 정렬하여 그리는 방식

`android.orientation="vertical/horizontal"`



`orientation="vertical"`



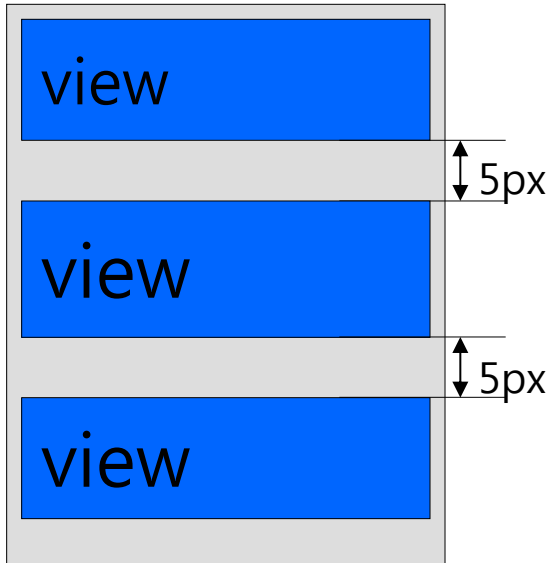
`orientation="horizontal"`



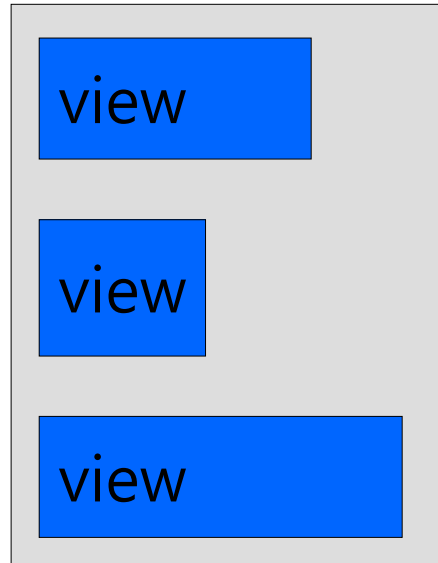
LinearLayout

모든 하위 구성 요소를 수직 또는 수평 형태 중 하나로 정렬하여 그리는 방식

`layout_margin`, `layout_gravity`, `layout_weight`, `layout_baselineAligned`



`layout_margin` : 5px



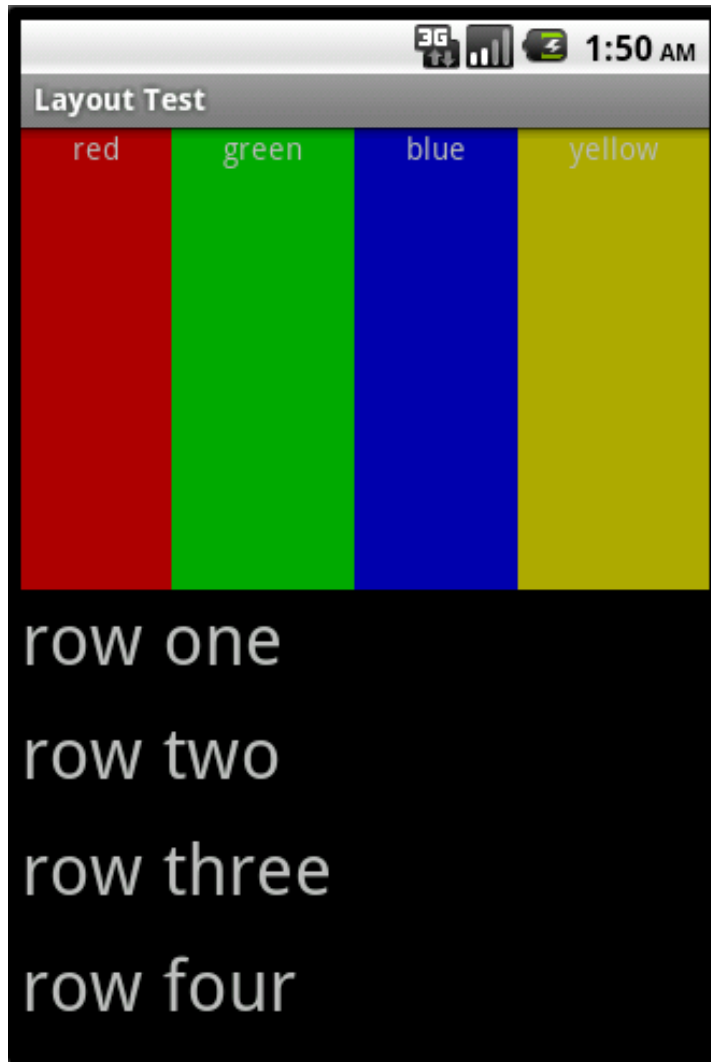
`gravity` : 내용물 정렬

`layout_gravity` : 뷰 자체 정렬



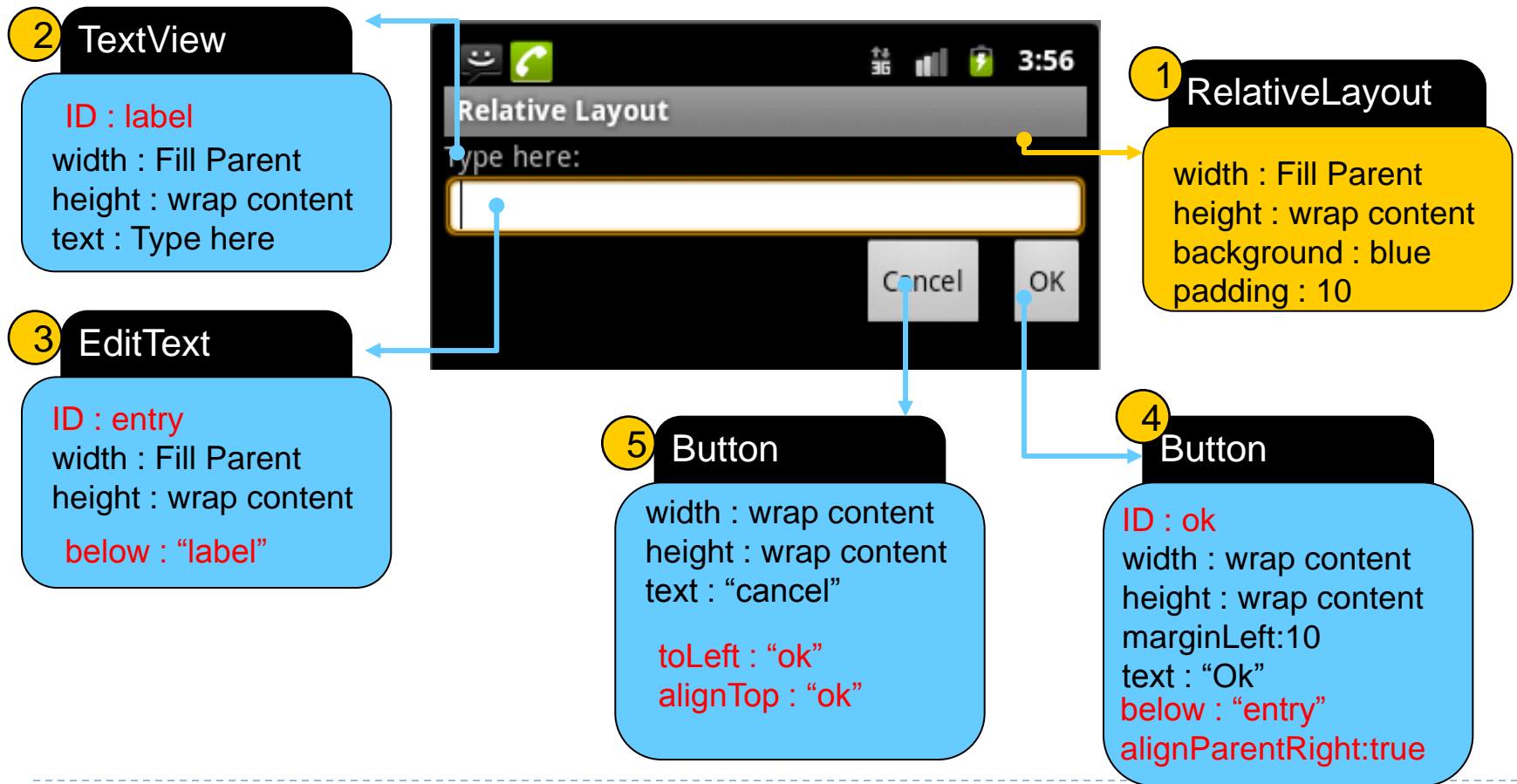
`layout_weight` :
남은 영역 분할

LinearLayout



RelativeLayout

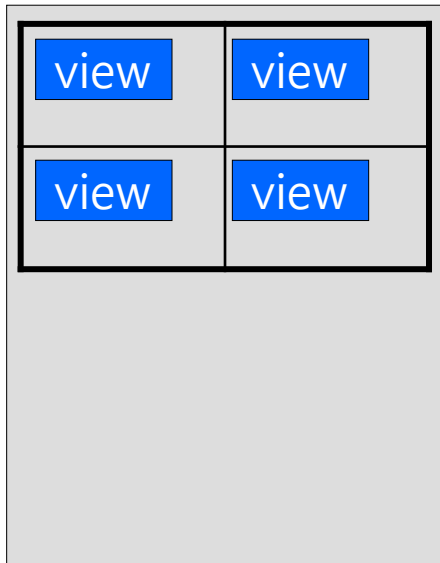
하위 구성 요소들이 다른 구성 요소의 상대적인 위치에 따라 그려짐
두 개의 객체가 있을 때 하나를 다른 객체의 아래에 그리거나 위에 그리는 등의 설정이 가능함



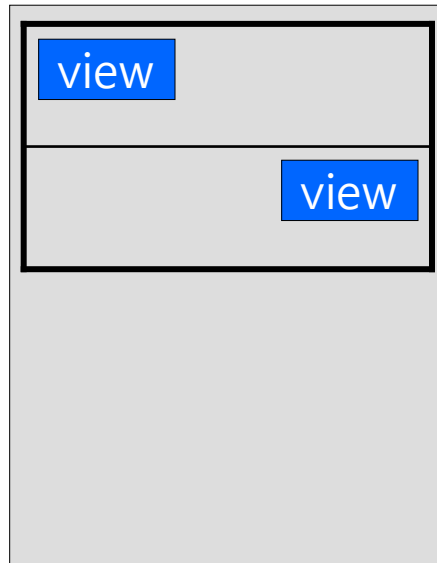
TableLayout

행과 열이라는 개념이 있어서 테이블의 행과 열을 기준으로 하위 구성 요소들을 배치함. 각 셀은 View 오브젝트로 구성되며 각 View Component의 화면상 절대적 위치는 테이블 구조에 따라 상대적으로 결정됨

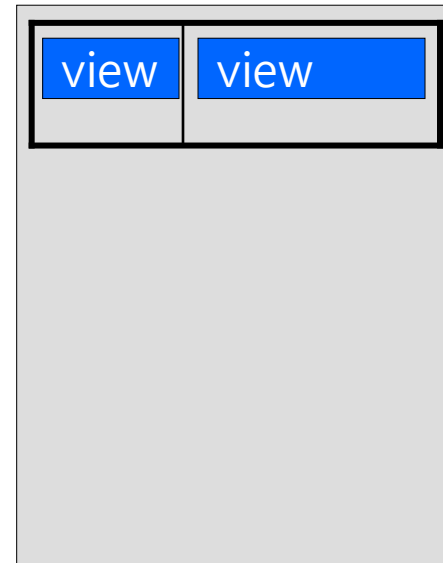
viewgroup : [TableLayout](#)



gravity 설정 : 정렬

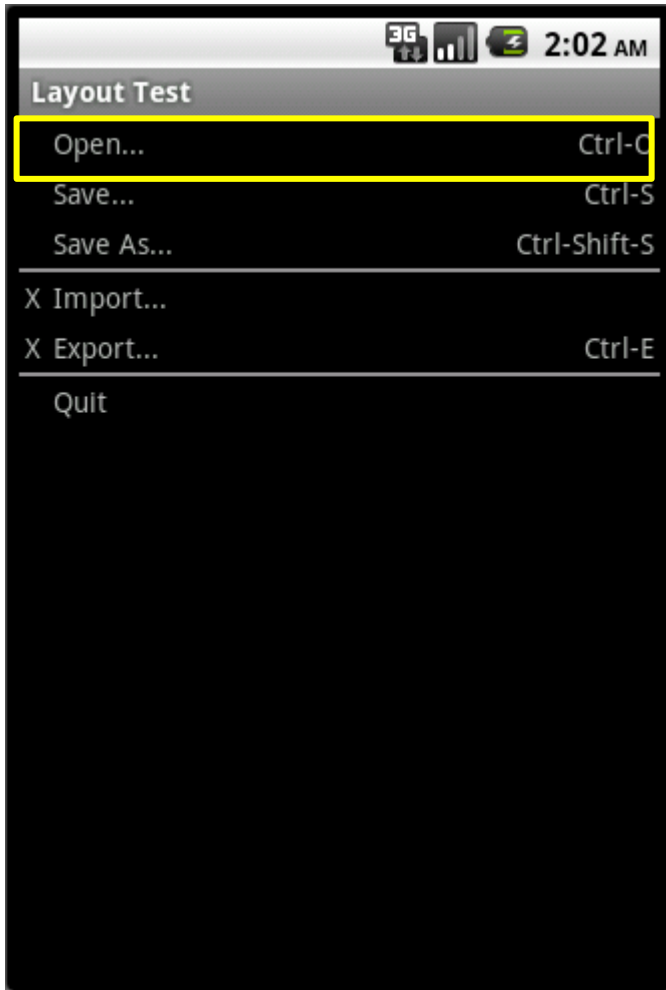


weight 설정 : 가중치 값

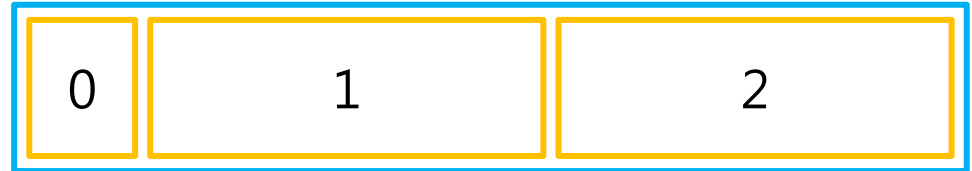


padding 처리 가능

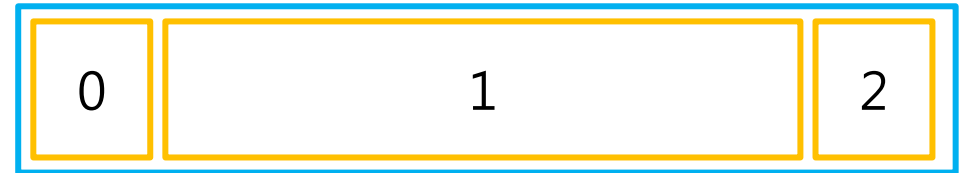
TableLayout



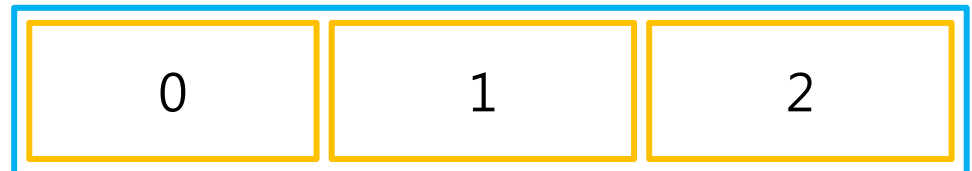
`android:stretchColumns="1,2"`



`android:stretchColumns="1"`



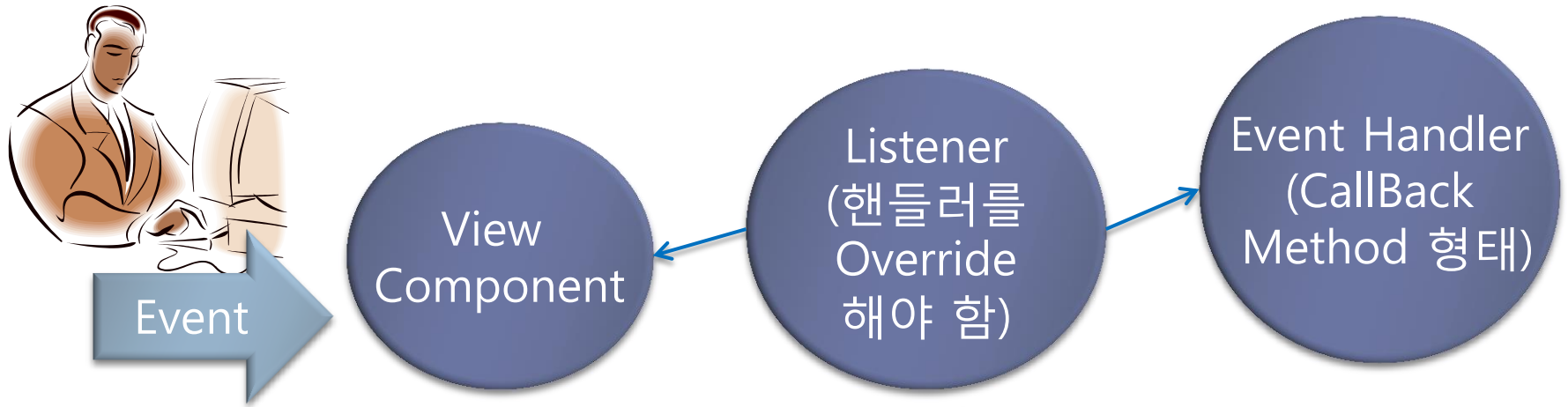
`android:stretchColumns="*"`



Activity Code 작성

View Component의 Event 처리/리스너/핸들러

Event 처리



1. Event 확인
2. Event Handler 호출
(다양한 방법으로 구현)

Event 처리 분류

1. View가 제공하는 Event 처리 – Listener(Interface)
2. Activity가 제공하는 Event 처리 – Method 방식

같은 것이 View와 Activity에 있으면, View가 제공하는 Event처리가 우선



Event 처리 기본

`view객체.setOnXXListener(class객체);`

- view객체: 리스너가 필요한 Event발생 View Component
예) Button, RadioButton, EditText 등
- setOnXXListener: 리스너 등록 메소드
- class객체: 해당 리스너의 핸들러가 포함되어 있는 객체



Event 처리 실습

Relative Layout 예제에서 사용된 예제에서

- ① Finish 버튼을 Cancel 버튼 왼쪽에 추가
- ② OK 버튼을 누르면 화면에 입력한 글자가 Toast로 출력됨
- ③ Cancel 버튼을 누르면 입력했던 글자가 모두 지워짐
- ④ Finish 버튼을 누르면 Application 종료

switch ~case로 3개의 버튼을 처리

Listener 구현 방법

- 1. 콜백 메서드 재정의
- 2. 리스너 인터페이스 구현
- 3. 액티비티가 리스너 구현
- 4. 뷰가 리스너 구현
- 5. 익명 이너 클래스 구현
- 6. 익명 이너 클래스의 임시 객체 사용



Event 처리

[1] 콜백 메서드 재정의

- boolean onTouchEvent (MotionEvent event)
- boolean onKeyDown (int keyCode, KeyEvent event)
- boolean onKeyUp (int keyCode, KeyEvent event)
- 메서드를 재정의 하기 위해 반드시 슈퍼 클래스(View, Button, TextView ...)를 상속 받아야 함
- 모든 이벤트에 대한 콜백이 다 정의되어 있는 것이 아니어서 **제한적으로만 적용**할 수 있음



Event 처리

[1] 콜백 메서드 재정의 =onXX()

```
public 클래스명 extends Activity {  
    public void onCreate() {  
        super.onCreate(savedInstanceState);  
        View vw = new 클래스명(this); //Activity의 콜백 메서드에 선언  
        setContentView(vw);  
    }  
}
```

```
protected class 클래스명 extends View {  
    public 클래스명(Context context) { // 생성자  
        super(context);  
    }  
    public boolean onXX(..., XXEvent event){ // Callback 메서드  
        // 처리 내용  
    }  
}  
}
```



Event 처리

[2] 리스너 인터페이스 구현 = OnXXListener(리스너), onXX(핸들러)

- 리스너: 이벤트를 처리하는 인터페이스(이벤트 리스너)
- View 클래스의 이너 인터페이스로 선언되어 있음
- 각 리스너는 추상 메서드(이벤트 핸들러)를 가지고 있음
 - View.OnTouchListener : `boolean onTouch(View v, MotionEvent event)`
 - View.OnKeyListener : `boolean onKey(View v, int keyCode, KeyEvent event)`
 - View.OnClickListener: `void onClick(View v)`
 - View.OnLongClickListener: `boolean onLongClick(View v)`
 - View.OnFocusChangeListener : `void onFocusChange(View v, boolean hasFocus)`
- 작성 절차
 1. 리스너를 구현하는 클래스 선언 및 추상 메서드 구현
 2. 리스너 객체 선언 및 생성
 3. 준비된 리스너 객체를 뷰의 이벤트와 연결
- 인터페이스 구현을 위해 별도의 클래스를 하나 더 선언해야 함



Event 처리

[2] 리스너 인터페이스 구현

```
public 클래스명 extends Activity {  
    public void onCreate() {  
        setContentView(R.layout.main);  
        findViewById(R.id.ok).setOnXXListener(리스너객체명);  
        // 다른 View Component도 동일한 방식으로 추가  
    }  
}
```

```
class 리스너클래스명 implements View.OnXXListener {  
    public void onXX(View v){  
        // 소스 구현  
        switch(v.getId()){  
            }  
        }  
    }  
}  
  
리스너클래스명 리스너객체명 = new 리스너클래스명();
```



Event 처리

[3] 액티비티가 리스너 구현

- 이벤트 받는 것이 여러 개일 경우 사용
- 액티비티에 implements View.OnXXListener 를 붙여 사용 (리스너 구현)

```
public 클래스명 extends Activity implements View.OnXXListener {  
    public void onCreate() {  
        setContentView(R.layout.main);  
        findViewById(R.id.ok).setOnTouchListener(this);  
    }  
  
    public void onXX(View v){  
        // 여기에 구현  
        switch(v.getId()){  
            }  
        }  
    }  
}
```



Event 처리

[4] 뷰가 리스너 구현

```
public 클래스명 extends Activity {  
    public void onCreate() {  
        setContentView(R.layout.main);  
        MyView mv = new MyView(this);  
        findViewById(R.id.ok).setOnXXListener(mv);  
    }  
    protected class MyView extends View implements  
        View.OnXXListener {  
        public void onXX(View v){  
            // 여기에 구현  
            switch(v.getId()){  
                }  
            }  
        }  
    }  
}
```



Event 처리

[5] 익명 이너 클래스 구현

```
CLASSNAME extends Activity {  
    public void onCreate() {  
        setContentView(R.layout.main);  
        MyView mv = new MyView(this);  
        findViewById(R.id.ok).setOnXXListener(리스너객체명);  
    }  
    Interface 리스너객체명 = new Interface( ) {  
        public void onXX(View v){  
            // 소스 구현  
            switch(v.getId()){  
            }  
        }  
    };  
}
```

- Interface의 예) Button.OnClickListener 인터페이스 사용
 - Interface obj = new Interface() { 메서드 구현 };
-



Event 처리

[6] 익명 이너 클래스의 임시 객체 사용

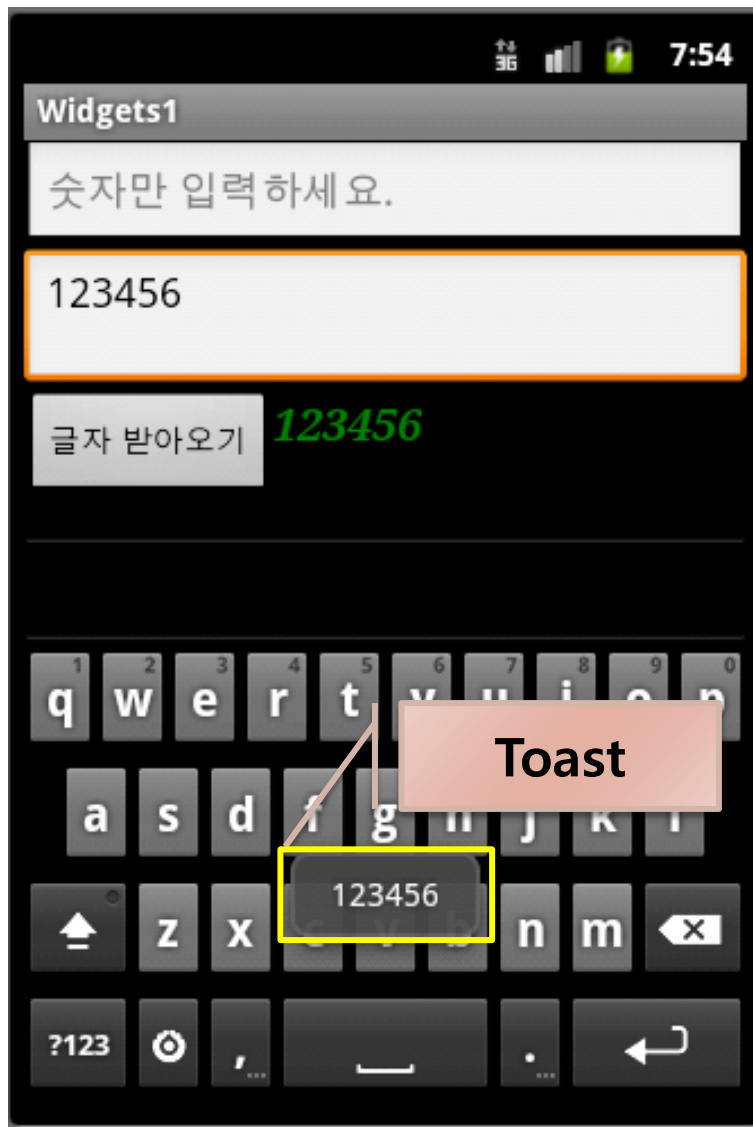
```
public 클래스명 extends Activity {  
    public void onCreate() {  
        setContentView(R.layout.main);  
        Button ok = (Button)findViewById(R.id.ok);  
        ok.setOnClickListener( new View.OnClickListener() {  
  
            public void onXX(View v){  
                // 소스 구현  
            }  
        } );  
    }  
}
```



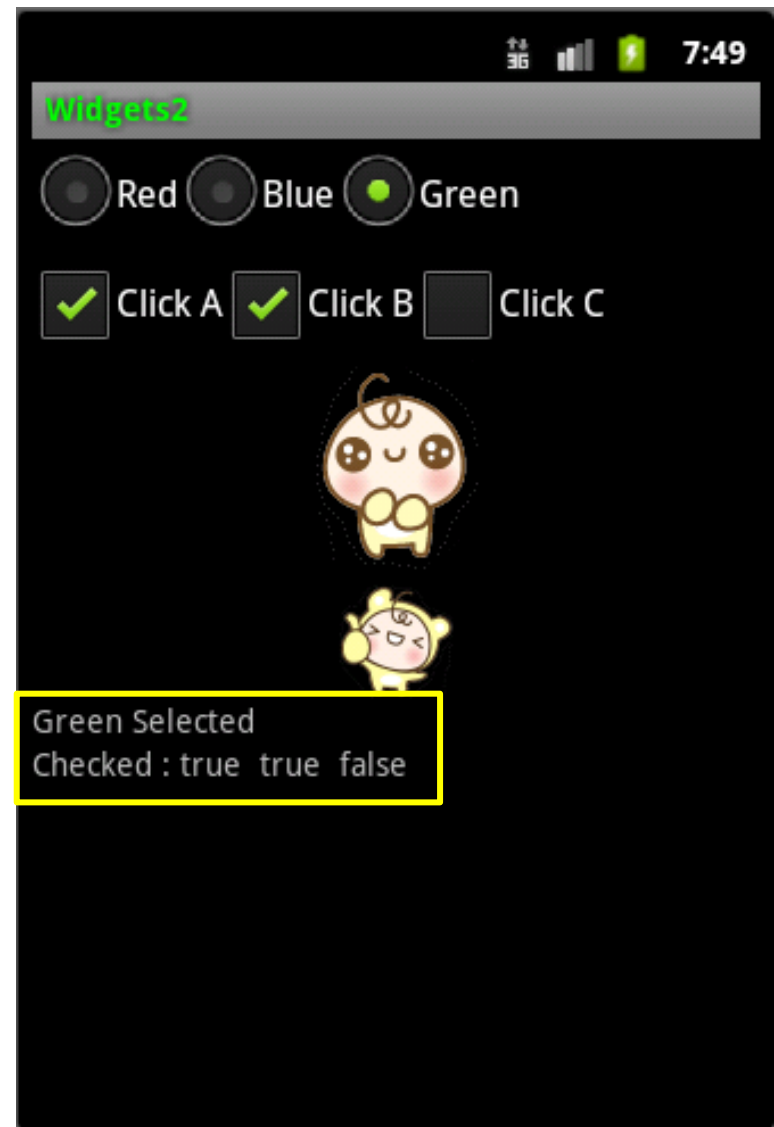
Widget1, 2예제에 작업

Component의 Event 처리/리스너/핸들러

Widgets1.java



Widgets2.java



Widgets1.java

```
public class Widgets1 extends Activity {  
    private EditText editText;  
    private TextView textView;  
    private Button getTextButton;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        editText = (EditText)findViewById(R.id.editText);  
        textView = (TextView)findViewById(R.id.textview);  
        getTextButton = (Button)findViewById(R.id.button);  
    }  
}
```



Widgets1.java

[6] 익명 이너 클래스의 임시 객체 사용

```
getTextButton.setOnClickListener(new OnClickListener(){  
    @Override  
    public void onClick(View v) {  
        textView.setText(editText.getText().toString());  
        // Quiz1: Toast 출력을 연습해 보세요.  
    }  
});
```

// Quiz2: 버튼을 클릭하지 않아도 editText에 입력하는 내용이 textView에 표시되도록 하세요. 내용이 없는 경우는 default_text가 표시되도록 하세요. (setOnKeyListener 사용)

R.타입.id

code에서 참조시
R.string.default_text

@타입/id

리소스(xml)에서 참조시
@string/default_text



Toast

- 작은 팝업 대화상자
- 사용자에게 임시적인 알림 사항을 전달할 때 유용함
- 디버깅용으로도 쓰임
- 플로팅 형태로 화면 하단에 잠시 나타나며 자동으로 사라짐

- 생성 및 사용

static Toast makeText(Context context, int resId, int duration)

R.string.default_text

static Toast makeText(Context context, CharSequence text, int duration)

.getText().toString()



Widget2.java

[3] 액티비티가 리스너 구현

```
public class Widgets2 extends Activity implements  
OnCheckedChangeListener, OnClickListener{  
    private RadioGroup radioGroup;  
    private static final int CHECKBNO = 3;  
    private CheckBox [] checkBox = new CheckBox[CHECKBNO];  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        radioGroup = (RadioGroup)findViewById(R.id.radioGroup);  
        checkBox[0] = (CheckBox)findViewById(R.id.checkbox1);  
        checkBox[1] = (CheckBox)findViewById(R.id.checkbox2);  
        checkBox[2] = (CheckBox)findViewById(R.id.checkbox3);  
        radioGroup.setOnCheckedChangeListener(this);  
        for (int i=0; i<checkBox.length; ++i)  
            checkBox[i].setOnClickListener(this);  
    }  
}
```



Widget2.java

[3] 액티비티가 리스너 구현

```
public void onCheckedChanged(RadioGroup group, int  
checkedId) {  
    // Quiz1: chechedId가 R.id.radio1인 경우 "Red Selected"라고  
    Toast 메시지를 보내고, setTitleColor()를 이용해 색을  
    Color.RED로 변경하고, R.id.radio2인 경우 "Blue Selected"라고  
    Toast 메시지를 보내고, setTitleColor()를 이용해 Color.BLUE로  
    변경하도록 switch ~ case문을 작성하시오.  
}  
public void onClick(View v) {  
    // Quiz2: 화면에 CheckBox의 상태를 표시하시오.  
}  
}
```

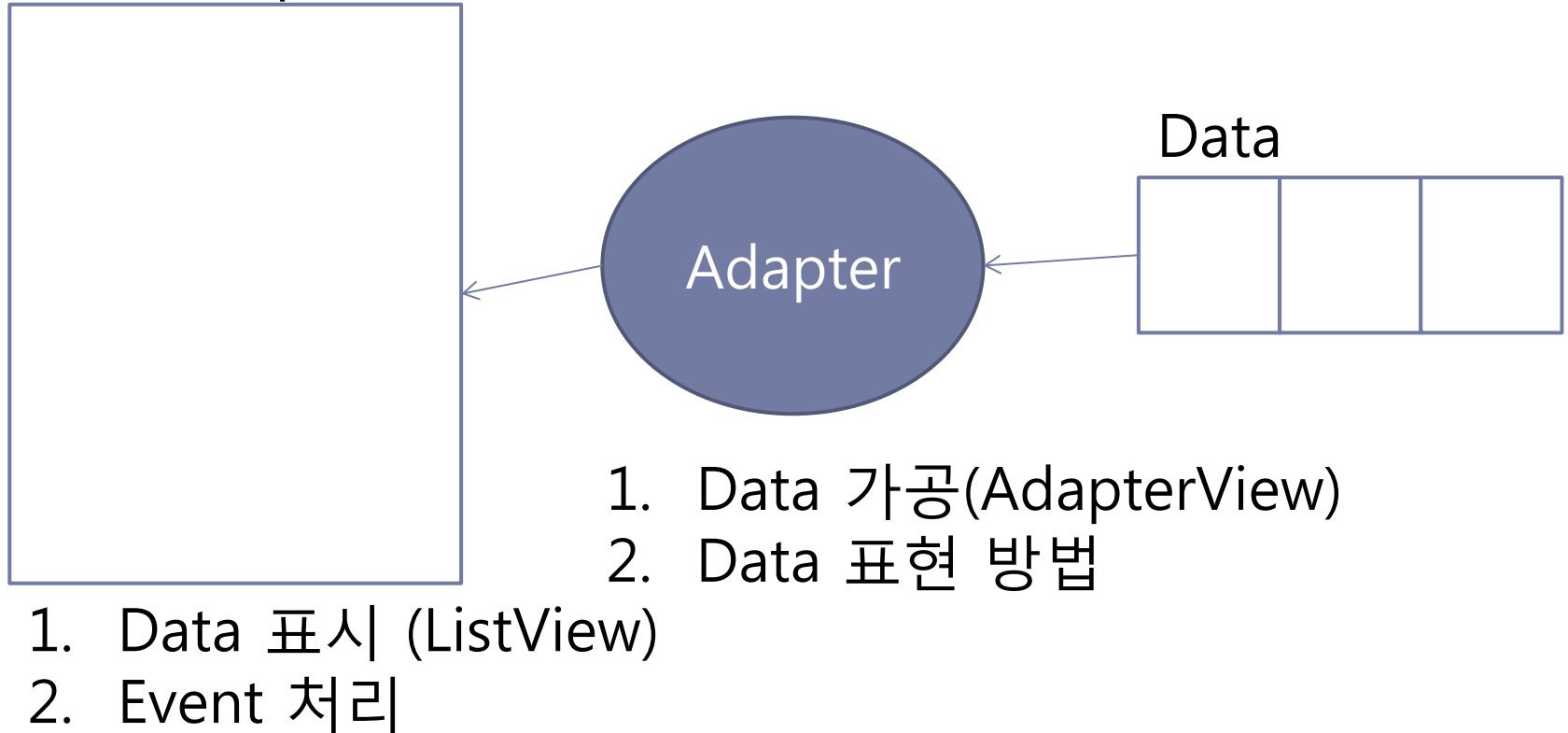




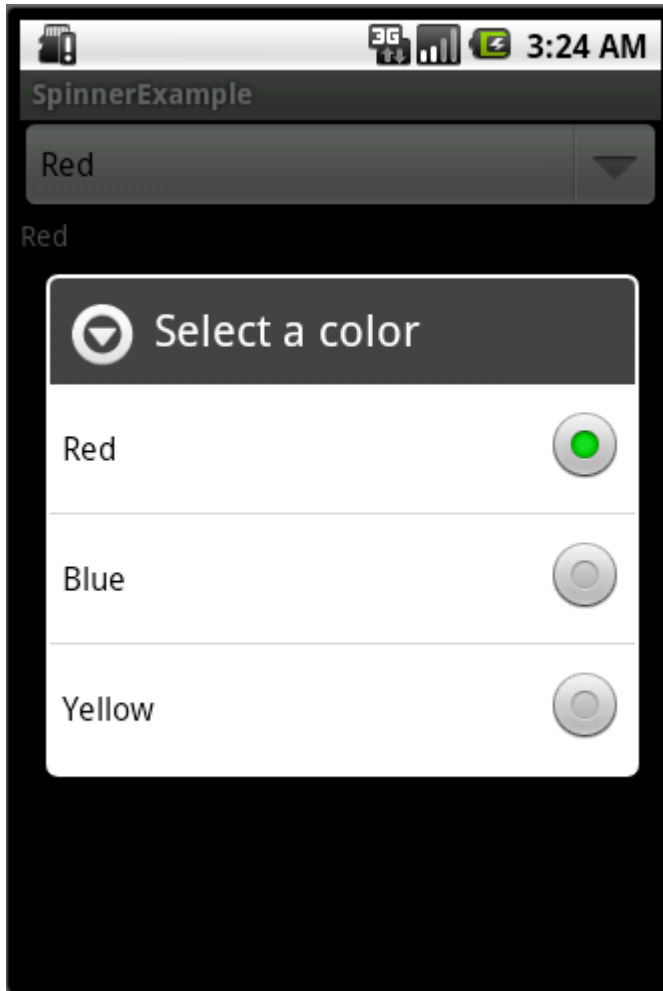
View Group의 사용 (AdapterView)

View Component - AdapterView

View Component



Spinner



- 클릭할 때만 팝업으로 펼쳐짐
- 여러 가지 선택 사항 중 하나를 선택 받을 때 사용
- 목록을 array로 지정
- AdapterView를 이용하여 모양 변경

Exercise1Activity.java – main.xml

```
<Spinner android:layout_height="wrap_content"
    android:id="@+id/spinner"
    android:layout_width="fill_parent"
    android:entries="@array/colorlist"
    android:prompt="@string/colortitle"
/>
```

Spinner : 콤보 박스
entries: 배열 형식이어야 함
prompt: 제목 – "text" 형식
일 수 없음, strings.xml에
등록된 것을 이용해야 함

```
<TextView android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textview"
    android:text="Please select one Item on List."
/>
```



Exercise1Activity.java – strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Change Title Color</string>
    <string-array name="colorlist">
        <item>Red</item>
        <item>Blue</item>
        <item>Green</item>
    </string-array>
    <string name="colortitle">Choose a color</string>
</resources>
```

spinner의 관리를 위해 배열 사용



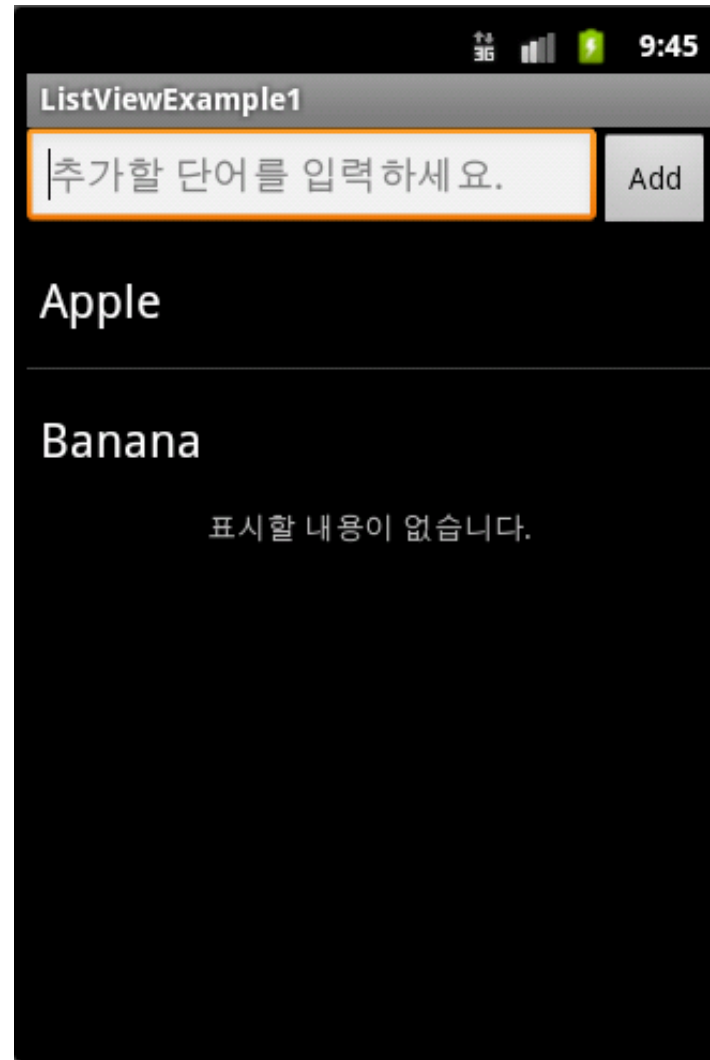
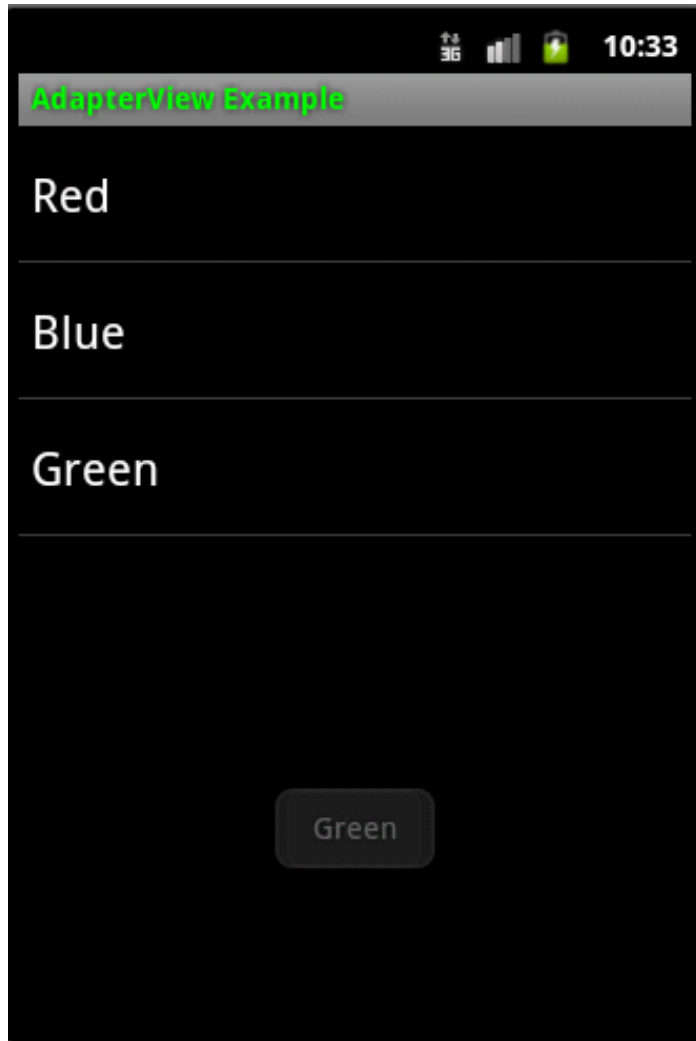
SpinnerExempl.java

[6] 익명 이너 클래스의 임시 객체 사용

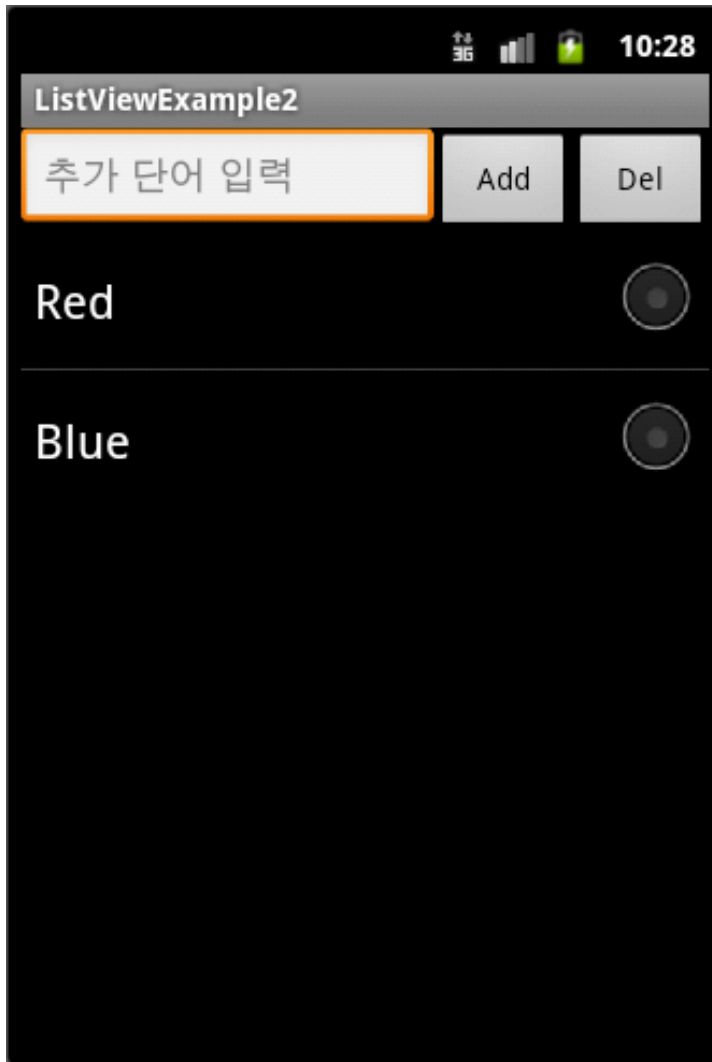
```
private Spinner spinner;  
spinner = (Spinner)findViewById(R.id.spinner);  
textView = (TextView)findViewById(R.id.textView);  
  
spinner.setOnItemSelectedListener(new OnItemSelectedListener(){  
    public void onItemSelected(AdapterView<?> parent, View view,  
        int position, long id) {  
        String str = (String)parent.getItemAtPosition(position);  
        textView.setText(str);  
    }  
    public void onNothingSelected(AdapterView<?> parent) {  
        // TODO Auto-generated method stub  
        textView.setText("Please select one item on the list.");  
    }  
});
```



AdapterViewExample/ ListViewExample1



ListViewExample2



- 목록을 실행 중에 변경 할 수 있음
- ListActivity를 상속 받으면 더욱 편리하게 구현할 수 있음

ListViewExample2 – main.xml

```
<LinearLayout android:layout_height="wrap_content"
    android:orientation="horizontal" android:layout_width="fill_parent">
    <EditText
        android:layout_height="wrap_content"
        android:inputType="text"
        android:layout_width="0dp" android:layout_weight="3"
        android:id="@+id/inputText"
        android:lines="1"
        android:hint="추가 단어 입력"/>
    <Button android:layout_height="wrap_content"
        android:layout_width="0dp" android:layout_weight="1"
        android:id="@+id/addButton"
        android:text="Add"/>
    <Button android:layout_height="wrap_content"
        android:layout_width="0dp" android:layout_weight="1"
        android:id="@+id/delButton"
        android:text="Del"/>
</LinerLayout>
```



ListViewExample2 – main.xml

<ListView

 android:id="@+id/list"

 android:layout_height="wrap_content"

 android:layout_width="fill_parent" />

→ ListActivity 사용시 id를 android:id로 변경이 필요함



ListviewExample2 – ListViewExample2.java

```
public class ListViewExample extends Activity {  
    private ArrayList<String> list;  
    private ArrayAdapter<String> adapter;  
    private EditText inputText;  
    private Button inputButton1;  
    private Button inputButton2;  
    private ListView listView;  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        inputText = (EditText)findViewById(R.id.inputText);  
        inputButton1 = (Button)findViewById(R.id.inputButton1);  
        inputButton2 = (Button)findViewById(R.id.inputButton1);  
    }  
}
```



ListViewExample2 – ListViewExample2.java

```
list = new ArrayList<String>();  
list.add("Red");  
list.add("Blue");  
  
adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_single_choice, list);  
listView = (ListView)findViewById(R.id.list);  
listView.setAdapter(adapter);  
listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
```

ListActivity가 아닌 Activity로 만드는 경우
"findViewById가 필요 없고, setListAdapter를 사용합니다."

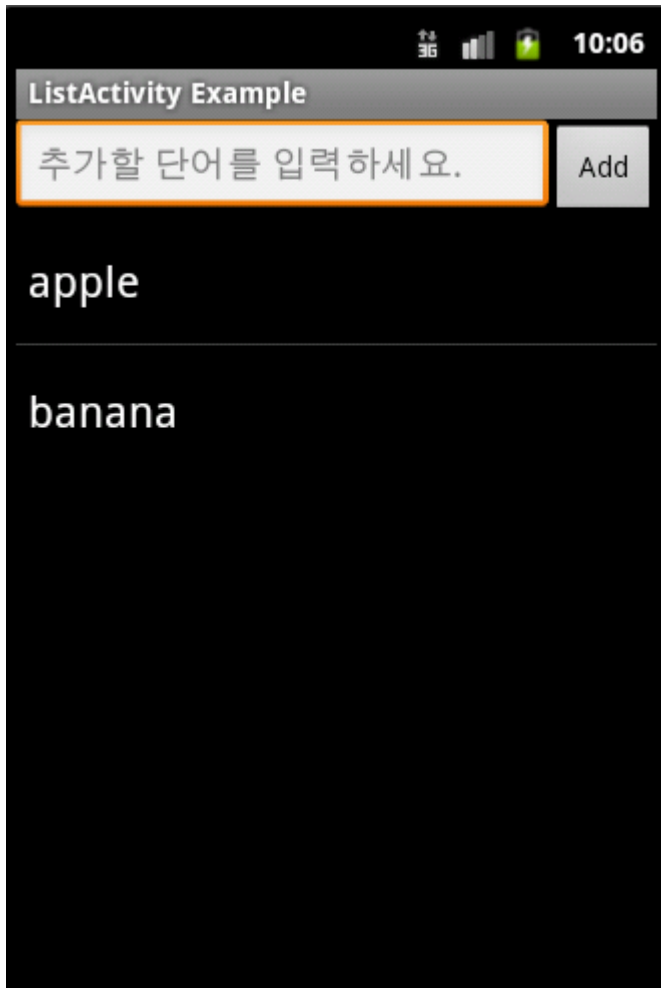


ListViewExample2 – ListViewExample2.java

```
inputButton1.setOnClickListener(new OnClickListener(){
    @Override
    public void onClick(View v) {
        if(inputText.getText().length() != 0){
            list.add(inputText.getText().toString());
            inputText.setText("");
            adapter.notifyDataSetChanged();
        } else {
            Toast.makeText(ListViewExample.this, "Input a word" ,
                Toast.LENGTH_SHORT).show();
        }
    }
});
```



ListActivity Example // Activity가 아닌 ListActivity를 상속 받음



- Activity가 아닌 ListActivity를 상속 받도록 함

실습과제

ListView Example 예제에 다음 기능을 추가하시오.

- ① Del 버튼을 추가하세요.
- ② 항목 선택 후, Del 버튼을 클릭하면 선택 항목이 삭제되도록 하세요.
- ③ 내용이 없는 상태에서 [Add]를 누르면 추가되지 않고, 단어를 입력하라는 메시지를 Toast로 표시하세요.



ListActivityExample – main.xml

```
<LinearLayout android:layout_height="wrap_content"
    android:orientation="horizontal" android:layout_width="fill_parent">
    <EditText
        android:layout_height="wrap_content"
        android:inputType="text"
        android:layout_width="0dp" android:layout_weight="3"
        android:id="@+id/inputText"
        android:lines="1"
        android:hint="추가 단어 입력"/>
    <Button android:layout_height="wrap_content"
        android:layout_width="0dp" android:layout_weight="1"
        android:id="@+id/addButton"
        android:text="Add"/>
    <Button android:layout_height="wrap_content"
        android:layout_width="0dp" android:layout_weight="1"
        android:id="@+id/delButton"
        android:text="Del"/>
</LinerLayout>
```



ListActivityExample- main.xml

```
<ListView android:layout_height="wrap_content"  
    android:id="@android:id/list"  
    android:layout_width="fill_parent" />
```

```
<TextView android:layout_width="wrap_content"  
    android:id="@android:id/empty"  
    android:text="표시할 내용이 없습니다."  
    android:layout_height="wrap_content"  
    android:layout_gravity="center_horizontal"/>
```

```
android:id="@+id/list"
```



ListActivityExample – ListViewExample.java

```
public class ListViewExample extends ListActivity {  
    private ArrayList<String> list;  
    private ArrayAdapter<String> adapter;  
    private EditText inputText;  
    private Button inputButton;  
    // private ListView listView; (필요 없음)  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        inputText = (EditText)findViewById(R.id.inputText);  
        inputButton1 = (Button)findViewById(R.id.inputButton);  
        // listView = (ListView)findViewById(R.id.list); (필요 없음)  
        list = new ArrayList<String>();  
    }  
}
```



ListActivityExample – ListViewExample.java

```
adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_1, list);  
setListAdapter(adapter);  
  
addButton.setOnClickListener(new OnClickListener(){  
    @Override  
    public void onClick(View v) {  
        list.add(inputText.getText().toString());  
        inputText.setText("");  
        adapter.notifyDataSetChanged();  
    }  
});  
}
```

ListActivity가 아닌 Activity로 만드는 경우
"findViewById가 필요 하고, View객체.setAdapter() 를 사용함





Menu, Dialog

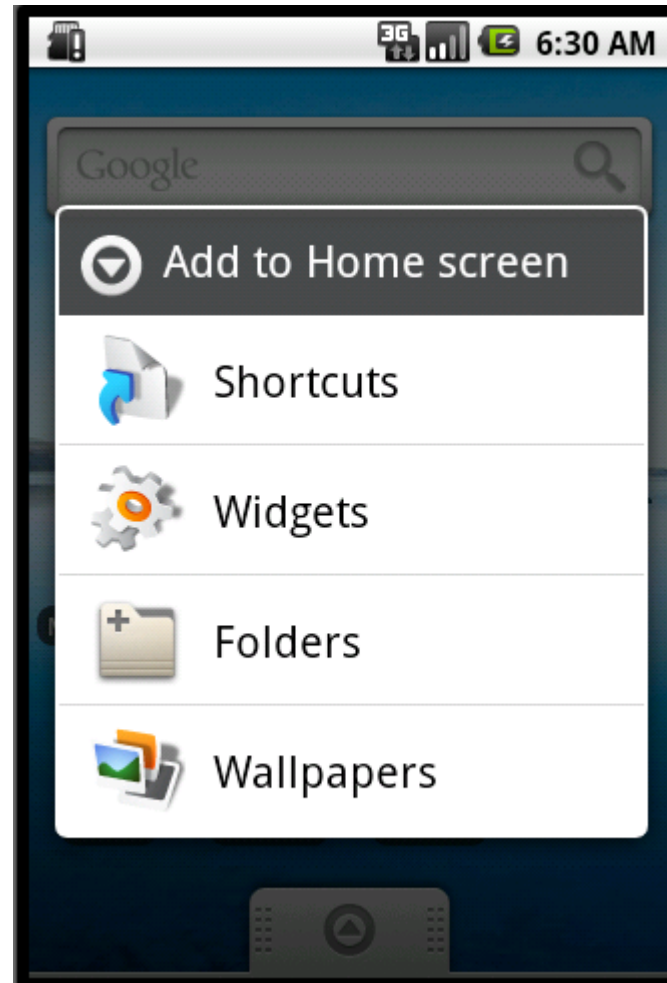
Menu

1. Option Menu : Menu Key 선택 시 호출되는 메뉴
 - 메뉴 항목은 5개 까지만 지원 됨
 - 미리 만들어져 있어야 함
2. Context Menu : Activity 화면을 길게 선택 시 호출 되는 메뉴 (Popup)
 - 항목의 수는 제한되어 있지 않음 (자동 스크롤)

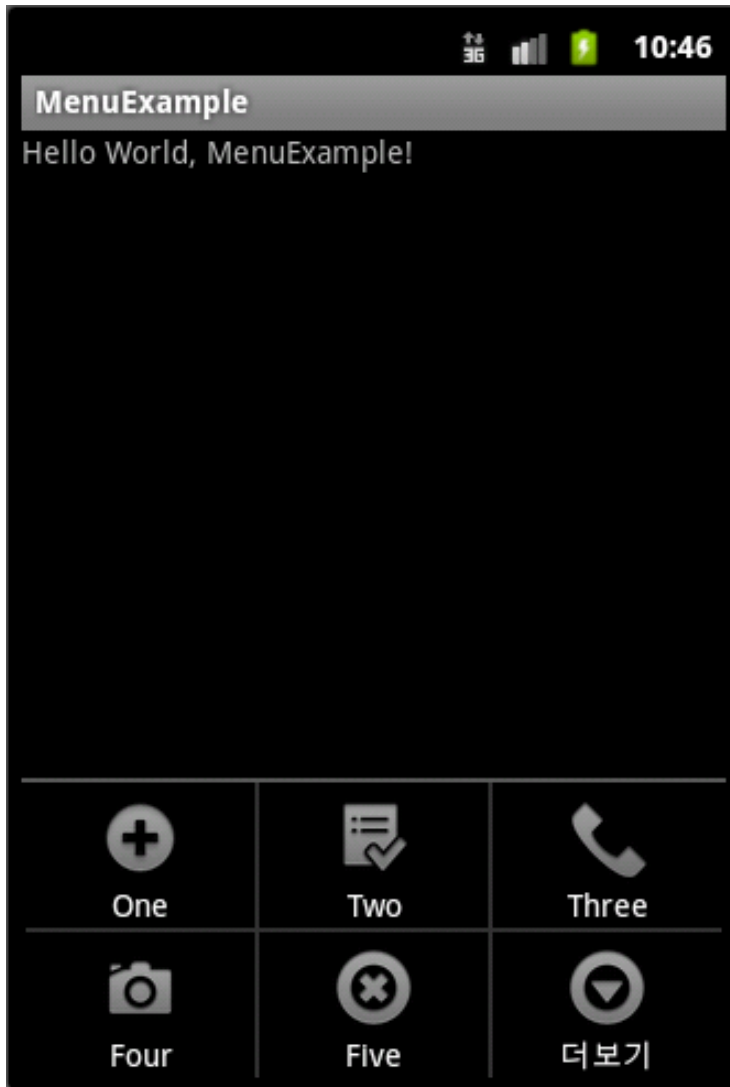
void	<code>onCreateContextMenu (ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo)</code> Called when a context menu for the <code>view</code> is about to be shown.
CharSequence	<code>onCreateDescription ()</code> Generate a new description for this activity.
boolean	<code>onCreateOptionsMenu (Menu menu)</code> Initialize the contents of the Activity's standard options menu.



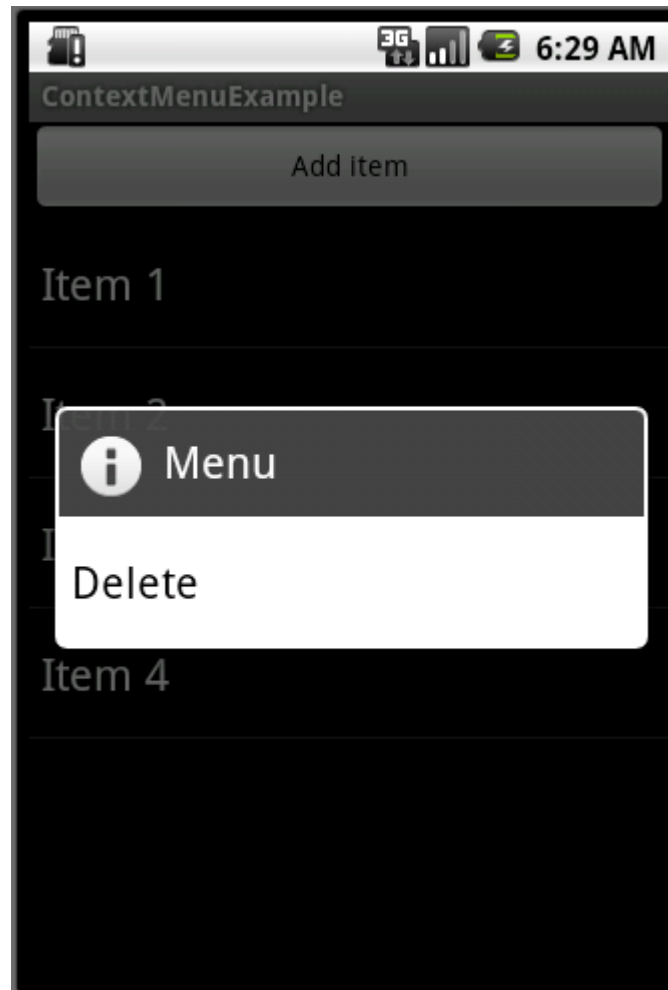
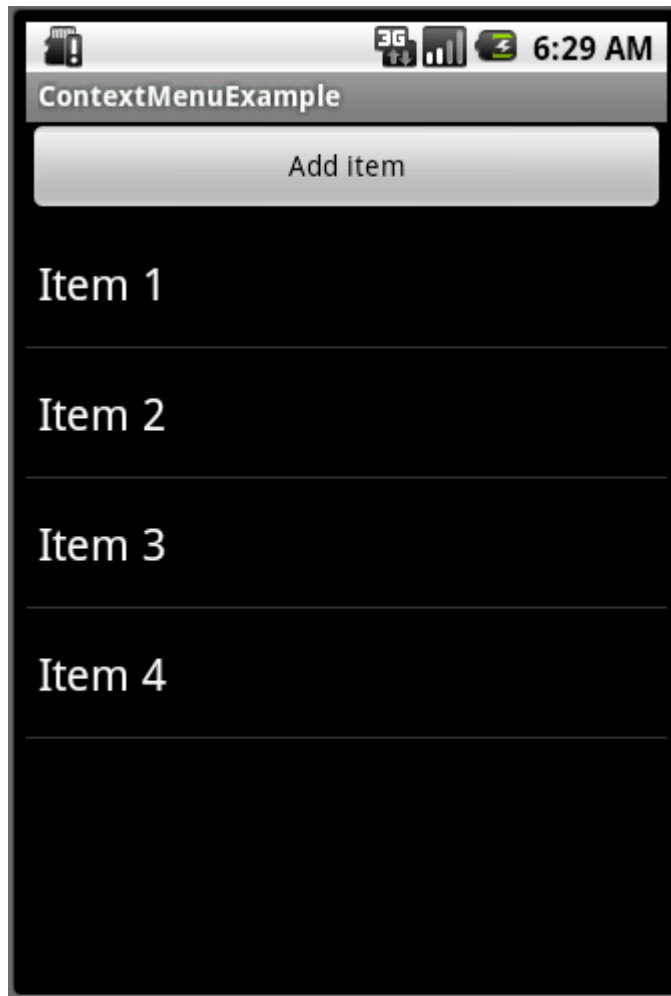
Option Menu & Context Menu



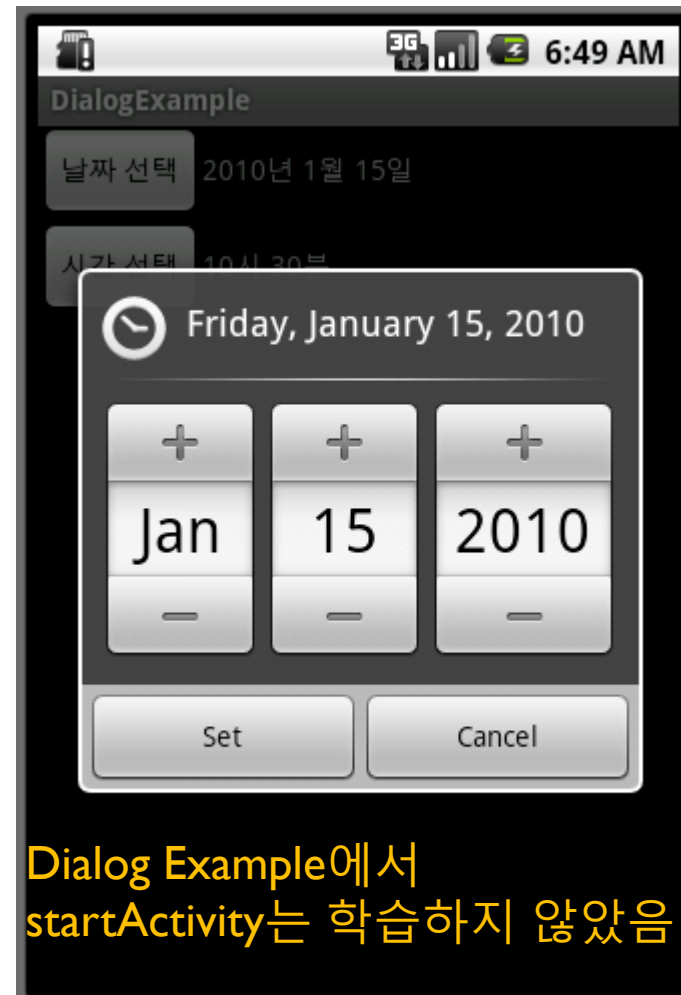
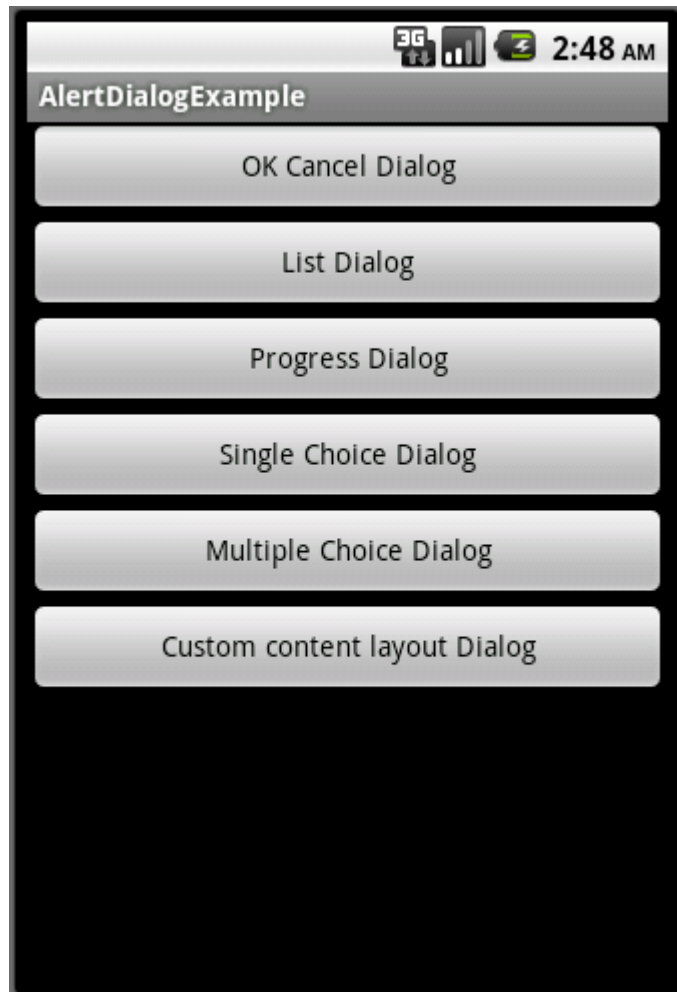
Option Menu (MenuExample)



Context Menu (ContextMenuExample)



DialogBox (AlertDialogExample, DialogExample)





Resource

리소스의 종류

종류	내용
색(Color)	폰트나 Background 등의 색을 지정하는 수치
문자열(String)	Activity에 표시하기 위한 문자열
크기(Dimensions)	폰트의 크기나 사이즈 등을 나타내는 단위가 있는 크기
그림 그리기 (Drawable)	Activity에 표시하기 위한 이미지
애니메이션 (Animation)	이미지나 문자 등을 이동시키기 위한 동작 정보
레이아웃(Layout)	Activity 내의 뷰(View)의 위치나 크기를 지정하는 정보
스타일과 테마 (Style, Theme)	레이아웃에서 사용하는 속성 조(pair)의 이름



리소스의 디렉터리 구성

디렉터리	리소스 타입
res/anim/	프레임이나 이미지의 전환 때의 애니메이션을 표현하는 XML 파일, 파일명이 ID가 됨
res/drawable/	JPEG나 PNG 등의 형식 파일, 파일명이 ID가 됨
res/layout/	Activity의 레이아웃을 표시하는 XML 파일, 파일명이 ID가 됨
res/values/	데이터를 표현하는 XML 파일. XML의 요소가 ID가 됨. 파일명은 임의이지만 보통은 다음과 같은 일이 많음. arrays.xml : 데이터의 배열, colors.xml : 색의 16진수 표현이나 색을 나타내는 Drawable 클래스, dimens.xml : 크기, strings.xml : 문자열, styles.xml : 스타일
res/xml/	임의의 xml 파일을 정의함
res/raw/	디바이스를 직접 복사하는 임의의 파일



Resource

XML file 형식: 값(values), 화면 구성(layout), image(drawable), Animation(anim) 등

절대 변경되면 안 되는 resource(raw, asset) : 디렉터리 형태로 존재

res

▶ drawable-hdpi

▶ drawable-ldpi

▶ drawable-mdpi

▶ layout

main.xml

▶ layout-land

main.xml

▶ values

colors.xml

strings.xml

▶ values-ko

strings.xml

리소스 파일의 이름에는 s 접미어가 붙고
모든 문자는 소문자로 작성함

문자열 : strings.xml

색상 : colors.xml

스타일, 테마: styles.xml

배열: arrays.xml

크기 정보: dimens.xml

R.타입.id

code에서 참조시
R.string.default_text

@타입/id

리소스(xml)에서 참조시
@string/default_text

대체 리소스

환경의 종류	가능한 접미어
언어	ISO 639-1이 정의하는 두 자리 소문자 국가 코드, us, kr, fr, ja, ru 등
지역	소문자 r 다음에 대문자 두 자리로 된 지열 코드, rUS, rKR, rFE
화면 방향	port, land, square
해상도	92dpi, 108dpi
터치 스크린	notouch, stylus(터치펜), finger
키보드 유무	keysexposed, keyshidden
입력 장치	nokeys, qwerty, 12key
내비게이션	nonnav, dpad, trackball, wheel
화면 크기	320*240, 480*320 등 가로, 세로에 상관 없이 항상 큰 값이 앞에 옴

슬라이드 노트 참조



대체 리소스

환경에 따라 변경되어야 하는 것을 resource가 저장되는 폴더에 따라서 관리

1. 여러 개의 접미어를 붙일 때는 **대시(-)로 구분하여 붙이며 반드시 도표의 순서에 맞게 작성해야 함**
2. 모든 폴더는 같은 부모에 속해야 하며 중첩되어서는 안 됨
3. 접미는 **대소문자를 구분함**. 한국어 레이아웃은 layout-kr 이어야 하며, layout-KR이라고 적어서는 안됨
4. 한 환경에 대해서는 하나의 접미만 가능, layout-kr-en은 안됨
5. **코드나 리소스에서 참조할 때는 접미를 붙이지 않음. 접미는 어디까지나 운영체제가 리소스를 선택할 때만 사용됨**

예) /res/drawables-en-port-640x480

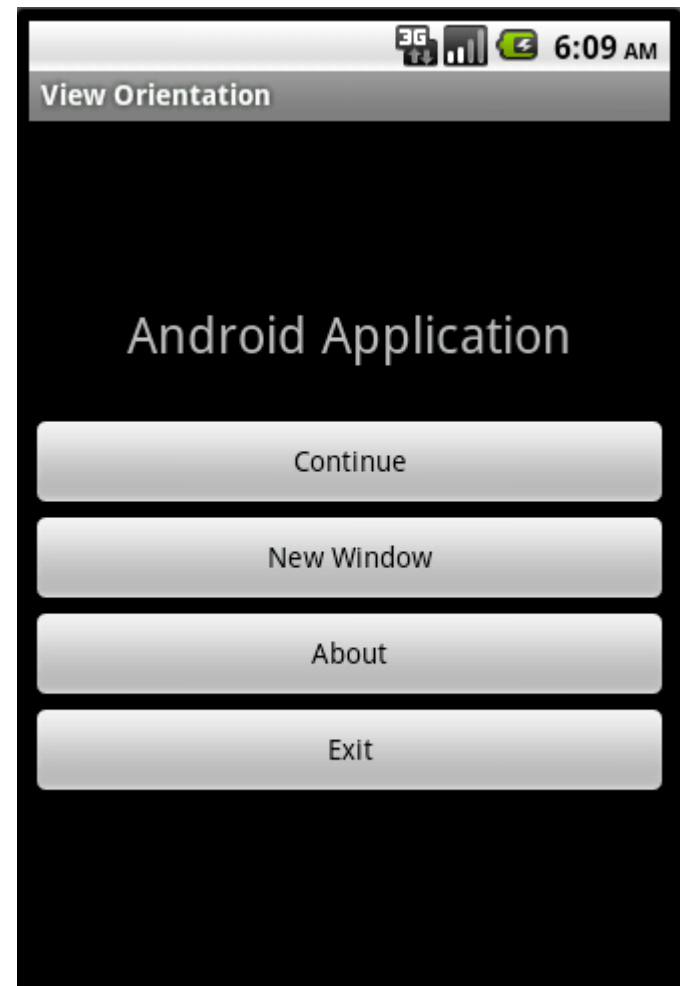
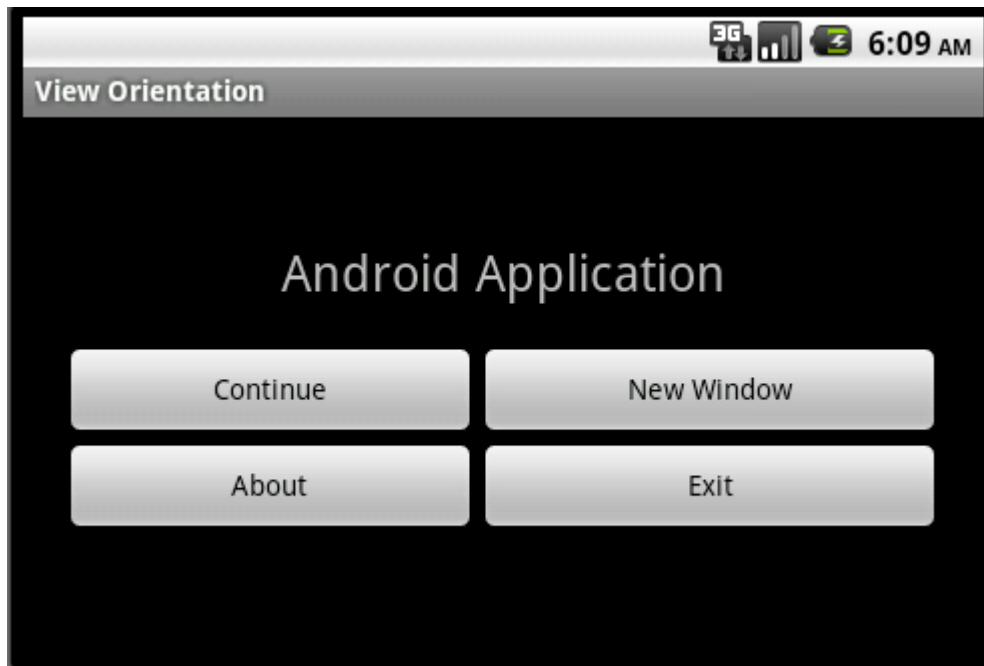
/res/values-en-rUS-port-finger

/res/drawables-en-rUS-land-640x480

/res/values/



View Orientation



- res
 - drawable-hdpi
 - drawable-ldpi
 - drawable-mdpi
 - layout
 - main.xml
 - layout-land
 - main.xml

[Ctrl] + [F11]을
눌러서 방향전환

폴더를 이용한 언어 변경

