



2장 : 안드로이드 UI 구성

옥 상 훈

Blog: <http://okgosu.tistory.com>

Web Site: <http://okgosu.net>

twitter @okgosu

2장:유저인터페이스구성

:안드로이드 UI구성요소와 액티비티



by OkGosu.Net

2.1 모바일 UX

- 모바일 환경에 대한 이해
- 모바일 화면 기획
- UX 향상을 위한 테크닉



모바일 3대 제약

- ❑ 리소스 제약
 - ❑ CPU 성능, 메모리, 전력
 - ❑ 네트워크
- ❑ 스크린 제약
 - ❑ 좁은 스크린
 - ❑ 다양한 크기의 폰
- ❑ 인터페이스 제약
 - ❑ 작은 버튼, 키보드, 터치



모바일 디바이스 3대 특징

- ❑ Multi-media
 - ❑ 음성, 영상 통화
 - ❑ 카메라 촬영, 비디오 녹화
 - ❑ 음악, 비디오 재생
- ❑ Always connected
 - ❑ 3G망
 - ❑ 무선 인터넷
- ❑ Smart Sensors
 - ❑ GPS
 - ❑ 가속도
 - ❑ 디지털 지구자기
 - ❑ 근접



UI 구성 방법

- ❑ 해상도 결정
- ❑ 화면 레이아웃 정의
- ❑ 컨트롤 배치
- ❑ 이벤트, 데이터 처리
- ❑ 액티비티 연결, 등록



OS

1.0



G1(HTC)

1.5



안드로-1(LG전자)

2.0



Droid(모토로라)



모토로이(모토로라)

2.1



Nexus One(HTC)



디자이너(HTC)



LG-LU2300 (LG전자)



SHW-M120S (삼성전자)



HW 키보드 유무



G1(HTC)



Droid(모토로라)



안드로-1(LG전자)



LG-LU2300 (LG전자)



Nexus One(HTC)



디자이너(HTC)



SHW-M120S (삼성전자)



모토로이(모토로라)



아이폰(애플)



스크린 사이즈

3



안드로-1
(LG전자)

3.2



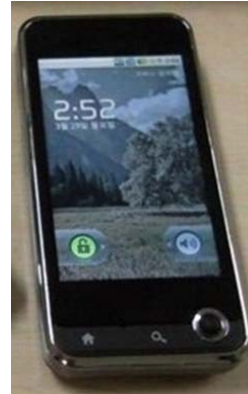
SHW-M120S
(삼성전자)

3.3



G1(HTC)

3.5



LG-LU2300
(LG전자)



디자이너
(HTC)

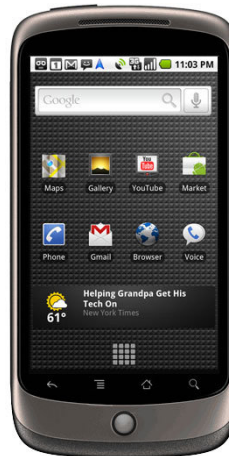


아이폰
(애플)

3.7



Droid(모토로라)



Nexus One(HTC)



모토로이(모토로라)



LCD 해상도

320 x 240(QVGA)



안드로-1(LG전자)

480 x 320(HVGA)



G1(HTC)

아이폰(애플)

854 x 480(WVGA854)



Droid(모토로라)

모토로이(모토로라)

800 x 480(WVGA800)



Nexus One(HTC)



디자이너(HTC)



SHW-M120S (삼성전자)



LG-LU2300 (LG전자)



카메라 해상도

2MP



아이폰 (애플)

3.2MP



G1(HTC)

8MP



모토로이(모토로라)

5MP



Droid
(모토로라)



Nexus One
(HTC)



안드로-1
(LG전자)



디자인어
(HTC)



LG-LU2300
(LG전자)



SHW-M120S
(삼성전자)



by OkGosu.Net

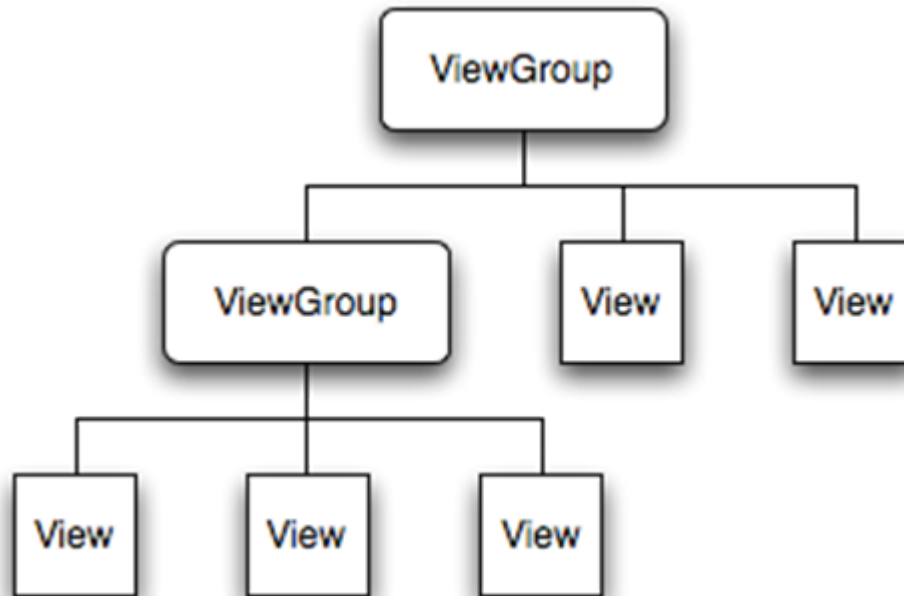
2.2 안드로이드 UI 구성 요소

- ❑ 안드로이드 UI 아키텍처
- ❑ 위젯의 종류
- ❑ 레이아웃의 종류
- ❑ XML 기반 레이아웃



안드로이드 UI 아키텍처

- View와 ViewGroup
- setContentView() 함수를 통해 화면에 추가



안드로이드 UI 아키텍처

- ❑ android.view.View
 - ❑ 사용자와 인터렉션하는 안드로이드 플랫폼의 UI 구성요소
 - ❑ widget을 만드는 베이스 클래스

- ❑ android.view.ViewGroup
 - ❑ 여러 개의 View나 ViewGroup 포함가능
 - ❑ layout 클래스의 모체

- ❑ [참고]
 - ❑ widget, layout 모두 android.widget.* 패키지에 소속됨



위젯의 종류

- 사용자가 조작하는 UI요소들
 - 예) 버튼, 체크박스, 라디오버튼 등
- GUI 툴킷의 'Views' 카테고리

Views

- GestureOverlayView
- SurfaceView
- View
- ViewStub
- WebView
- AnalogClock
- AutoCompleteTextView

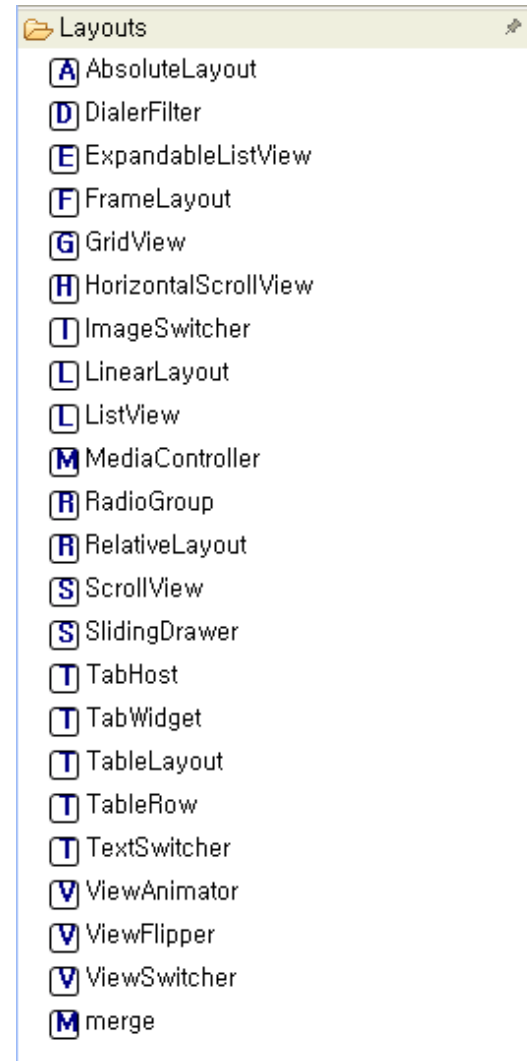
- Button
- CheckBox
- CheckedTextView
- Chronometer
- DatePicker
- DigitalClock
- EditText
- Gallery
- ImageButton
- ImageView
- MultiAutoCompleteTextView
- ProgressBar

- QuickContactBadge
- RadioButton
- RatingBar
- SeekBar
- Spinner
- TextView
- TimePicker
- ToggleButton
- TwoLineListItem
- VideoView
- ZoomButton
- ZoomControls
- include



레이아웃의 종류

- ❑ 위젯이나 레이아웃 포함 가능
- ❑ 위젯을 담아 배치할 수 있음
 - ❑ 예) 수평LinearLayout, GridLayout
- ❑ GUI 툴킷의 'Layouts' 카테고리



XML 레이아웃

❑ 역할

- ❑ Flex처럼 UI 구성을 정의하는 XML파일
- ❑ MVC 아키텍처에서 View 담당

```
<?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="wrap_content"
```

```
        android:layout_height="wrap_content" android:id="@+id/txtView"
```

```
        android:text="@string/app_name"/>
```

```
</LinearLayout>
```



XML 레이아웃

- 화면 속성

- *android:id="@+id/txtView"*
 - *android:layout_width="fill_parent"*
 - *android:layout_height="fill_parent"*



XML 레이아웃

□ 화면 연결

- 안드로이드 SDK의 aapt툴에서 레이아웃으로부터 R.java 생성
- setContentView에서 레이아웃 지정
- findViewById를 통해 레이아웃 컴포넌트 참조

```
Button btn;  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    btn = (Button)findViewById(R.id.btn);  
}
```



인플레이션

- ❑ XML 레이아웃에 정의된 내용을 분석하여 View 객체의 트리 구조를 만들어 내는 작업
 - ❑ LayoutInflater 클래스
 - ❑ inflate 메소드에서 XML을 로드하여 화면 리턴
 - ❑ XML 정의 메뉴 인플레이션에서 사용



2.3 기본 위젯의 활용

- 기본 위젯의 종류와 특성
- 이벤트 처리 방법
- 위젯의 활용



기본 위젯의 종류

- ☐ TextView
- ☐ Button
- ☐ ImageView
- ☐ ImageButton
- ☐ EditText
- ☐ CheckBox
- ☐ RadioButton, RadioButtonGroup



기본 위젯의 특성

□ View의 하위 클래스

- visibility : 화면 표시 여부
- background : 배경색
- 포커스 관련 : nextFocusDown, nextFocusLeft, nextFocusRight, nextFocusUp
- 주요 메소드
 - setEnabled() / isEnabled()
 - setFocus() / requestFocus() / isFocused()
 - getParent() : 상위 위젯이나 컨테이너 리턴
 - findViewById() : 컨테이너 내부에서 지정한 ID에 해당하는 위젯리턴
 - getRootView() : 최상위 컨테이너(setContentView()에서 넘겨준 위젯이나 컨테이너) 리턴



이벤트 리스너 처리

□ anonymous 인터페이스 구현

```
// anonymous implementation of OnClickListener
private OnClickListener mCorkyListener = new OnClickListener() {
    public void onClick(View v) {
        ....
    }
};

protected void onCreate(Bundle savedInstanceState) {
    Button button = (Button)findViewById(R.id.corky);
    button.setOnClickListener(mCorkyListener);
}
```



이벤트 리스너 처리

- 클래스 레벨에서 인터페이스 구현
 - 별도의 클래스 로드와 오브젝트 할당을 피할 수 있음

```
public class ExampleActivity extends Activity
implements OnClickListener {
    protected void onCreate(Bundle savedInstanceState) {
        Button button = (Button)findViewById(R.id.corky);
        button.setOnClickListener(this);
    }
    // OnClickListener callback 구현
    public void onClick(View v) {
        ....
    }
    ...
}
```



View의 주요 이벤트 리스너

- ❑ `onClick()`
 - ❑ `View.OnClickListener`로부터 콜백
 - ❑ 터치모드일 때 사용자가 아이템을 터치하거나
 - ❑ 네비게이션키나 트랙볼로 아이템 위에서 엔터키를 누르거나 트랙볼을 눌렀을 때 호출

- ❑ `onLongClick()`
 - ❑ `View.OnFocusChangeListener`로부터 콜백
 - ❑ 터치모드일 때 사용자가 아이템을 터치해서 잡거나
 - ❑ 네비게이션키나 트랙볼로 해당 아이템을 포커스하여 엔터키를 누르고 있거나 또는 1초동안 트랙볼을 누르고 있을 때 호출

- ❑ `onFocusChange()`
 - ❑ `View.OnFocusChangeListener`로부터 콜백
 - ❑ 네비게이션키 또는 트랙볼을 사용하여 아이템 위로 움직이거나 벗어날 때 호출



View의 주요 이벤트

- ❑ onKey()
 - ❑ View.OnKeyListener로부터 콜백
 - ❑ 아이템을 포커스하여, 디바이스에 있는 키를 누르거나 놓았을 때 호출

- ❑ onTouch()
 - ❑ View.OnTouchListener로부터 콜백
 - ❑ 아이템의 경계내에서 스크린을 누르고, 놓고, 또는 어떤 움직임 행위를 포함하는 터치 액션을 수행할 때 호출

- ❑ onCreateContextMenu()
 - ❑ View.OnCreateContextMenuListener로부터 콜백
 - ❑ 일정시간 클릭하여 컨텍스트 메뉴가 만들어져 있을 때 호출



이벤트 핸들러 예

❑ android.view.KeyEvent

- ❑ onKeyDown(int, KeyEvent) - 신규 키 이벤트가 발생할 때 호출
- ❑ onKeyUp(int, KeyEvent) - 키 업(up) 이벤트가 발생할 때 호출

❑ android.view.MotionEvent

- ❑ onTrackballEvent(MotionEvent) - 트랙볼trackball 모션 이벤트가 발생할 때 호출
- ❑ onTouchEvent(MotionEvent) - 터치 스크린 모션 이벤트가 발생할 때 호출
- ❑ onFocusChanged(boolean, int, Rect) - 뷰가 포커스를 얻거나 잃게될 때 호출



기본 위젯

□ TextView

□ 텍스트 레이블 표시

- `android:text="표시할 텍스트값"`
- `android:typeface="폰트명"`
- `android:textStyle="normal/bold/italic"`
- `android:textColor="#FF0000"`
- `android:textSize="20sp"`
- `android:gravity = "right/center_vertical"`
- `android:singleLine = "true"`

Plain
Serif
Bold
Italic

□ 사이즈

- px : 픽셀,
- dip (dp)
 - 장치독립적 픽셀, 픽셀의 크기와 밀도에 따른 개체의 크기 변화를 막음
 - LCD density가 160일 경우, 1 DIP = 1 Pixel
 - density 값이 240으로 변경되면 1 DIP = 1.5 Pixel
- sp : dp와 유사하나 사용자의 글꼴 크기 설정에 의해 측정됨
- pts : 포인트, 1pts = 1/72in
- in : 인치
- mm : 밀리미터



기본 위젯

□ Button

□ TextView 하위 클래스

- *android:layout_width="wrap_content"*
- *android:layout_height="wrap_content"*

□ 이벤트 처리

```
public class MyActivity extends Activity {  
    protected void onCreate(Bundle icle) {  
        super.onCreate(icle);  
  
        setContentView(R.layout.content_layout_id);  
  
        final Button button = (Button) findViewById(R.id.button_id);  
        button.setOnClickListener(new View.OnClickListener() {  
            public void onClick(View v) {  
                // Perform action on click  
            }  
        });  
    }  
}
```

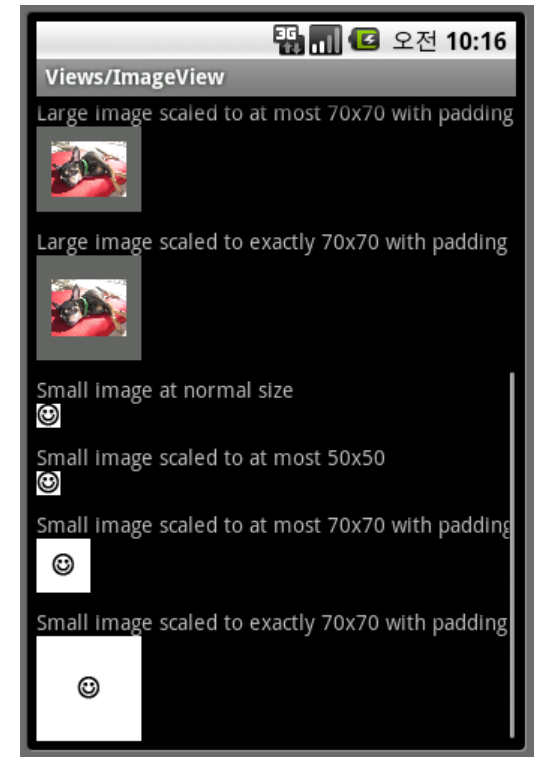


기본 위젯

❑ ImageView

- ❑ TextView 하위 클래스
- ❑ android:src 속성에 이미지 지정
- ❑ res/drawable 디렉토리 이미지파일명까지 지정
- ❑ 외부 이미지는 setImageURI()

```
<ImageView android:id="@+id/ImageView01"  
    android:layout_width="wrap_content"  
    android:src="@drawable/kind"  
    android:layout_height="wrap_content"/>
```

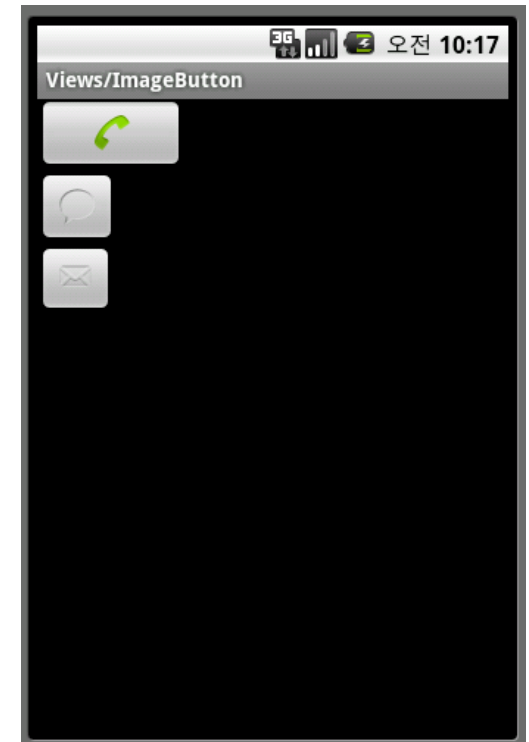


기본 위젯

❑ ImageButton

- ❑ Button 하위 클래스
- ❑ 이미지 버튼 스킨 리소스를 지정

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
        android:drawable="@drawable/button_pressed" />
    <item android:state_focused="true"
        android:drawable="@drawable/button_focused" />
    <item android:drawable="@drawable/button_normal" />
</selector>
```



기본 위젯

□ EditText

□ TextView 하위 클래스

□ 텍스트 입력

- *android:autoLink* = "링크주소를 찾아 자동으로 클릭가능하게함"
- *android:autoText* = "자동 스펠링 교정 여부"
- *android:capitalize* = "자동 대문자 변경 여부"
- *android:digits* = "특정 숫자만 입력 받도록 제한"
- *android:singline* = "한줄 또는 여러줄 입력"
- *android:hint* = "배경 글씨"
- *android:lines* = "라인수"
- 기타 속성
 - *numeric, password, phoneNumber*

```
<EditText android:id="@+id/EditText01" android:text="@string/gosu_str"
android:layout_width="fill_parent" android:singleLine="false"
android:layout_height="120sp"/>
```



기본 위젯

☐ CheckBox

- ☐ TextView – Button – CompoundButton 하위 클래스
- ☐ *android:checked* : 체크 상태 설정
- ☐ isChecked() : 체크 상태 확인
- ☐ setChecked() : 체크 지정
- ☐ toggle() : 체크 상태 변경
- ☐ OnCheckedChangeListener : 상태변경 감지



```
<CheckBox android:id="@+id/CheckBox01"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="CheckBox"  
    android:checked="true"/>
```



기본 위젯

☐ RadioButton

☐ TextView – Button – CompoundButton 하위 클래스

☐ RadioButtonGroup으로 버튼 관리

☐ orientation: 나열 방향

☐ check() : 특정 라디오 버튼 체크

☐ clearCheck() : 특정 라디오 버튼 체크 해제

☐ getCheckedRadioButtonId() : 현재 선택된 라디오 버튼 ID리턴



```
<RadioGroup android:id="@+id/RadioGroup01" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:orientation="horizontal">
```

```
<RadioButton android:id="@+id/RadioButton01" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Yes" android:checked="true"/>
```

```
<RadioButton android:id="@+id/RadioButton02" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="No"/>
```

```
</RadioGroup>
```



2.4 안드로이드 레이아웃

- 레이아웃의 종류
- 레이아웃의 사용법



레이아웃의 종류

□ 위젯 배치

- LinearLayout : 수직, 수평으로 배치
- RelativeLayout : 특정 위젯을 기준으로 상하좌우에 배치
- TableLayout : html 테이블 형태로 배치
- AbsoluteLayout : 좌표 기준 배치
- FrameLayout : 좌측 상단에 중첩 배치

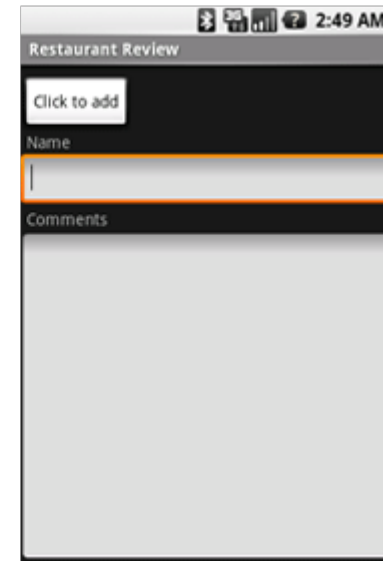
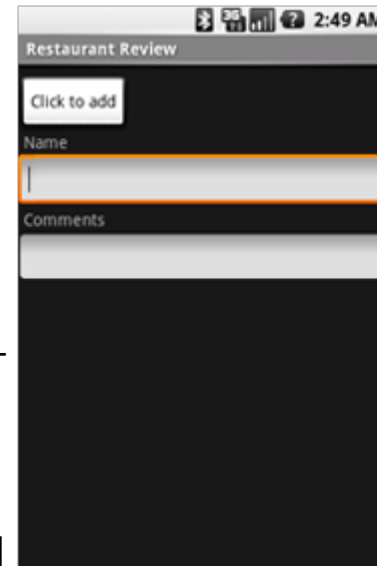
□ 특수 기능

- ScrollView : 스크롤 기능
- TabHost : 탭 리스트
- ViewFlipper : 뉴스티커 형태



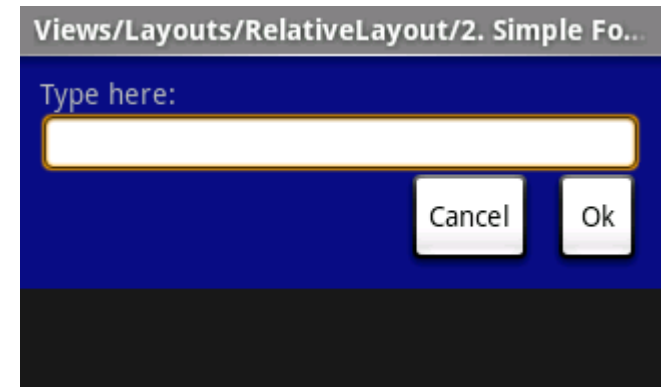
LinearLayout

- 주요 속성
 - 방향
 - `android:orientation= "horizontal" 또는 vertical`
 - `setOrientation()`
 - 가로세로채우기
 - `android:layout_width, layout_height`
 - `px` 지정, `wrap_content`, `fill_parent`
 - 가중치
 - `android:layout_weight`
 - 위젯을 보여주고 남은 공간을 할당하는 상대비율
 - 그래비티
 - `android:layout_gravity`
 - 레이아웃이 아니라 내부 위젯들에 대해 설정함으로써 정렬방향지정
 - 패딩
 - `android:padding`
 - 내부위젯과 테두리사이 간격



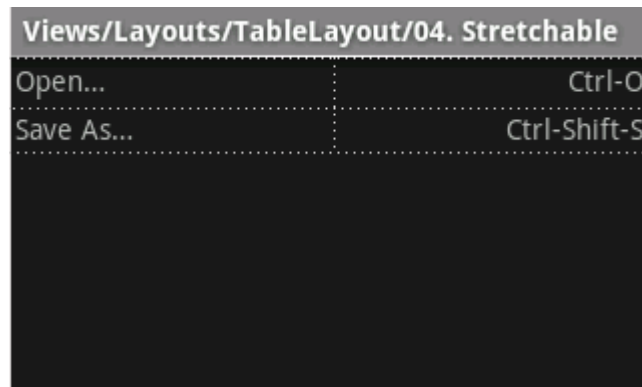
RelativeLayout

- ❑ 위젯이나 컨테이너 레이아웃을 기준으로 배치하기 위함
 - ❑ 예) A는 B의 항상 오른쪽에 위치
 - ❑ 컨테이너 레이아웃 기준
 - ❑ `layout_alignParentTop`,
`layout_alignParentBottom`,
`layout_alignParentLeft`,
`layout_alignParentRight`,
`layout_alignCenterHorizontal`,
`layout_alignCenterVertical`,
`layout_alignCenterInParent` (정중앙)
- ❑ 특정 위젯 기준
 - ❑ 기준 위젯에는 `android:id` 속성값 지정해야함
 - ❑ 기준 위젯은 먼저 나와야함
 - ❑ 기준 위젯이 `fill_parent`하면 공간을 다 차지해버림
 - ❑ `layout_above`, `layout_below`,
`layout_toLeftOf`, `layout_toRightOf`
 - ❑ `layout_alignTop`, `layout_alignBottom`,
`layout_alignLeft`, `layout_alignRight`,
`layout_alignBaseline`



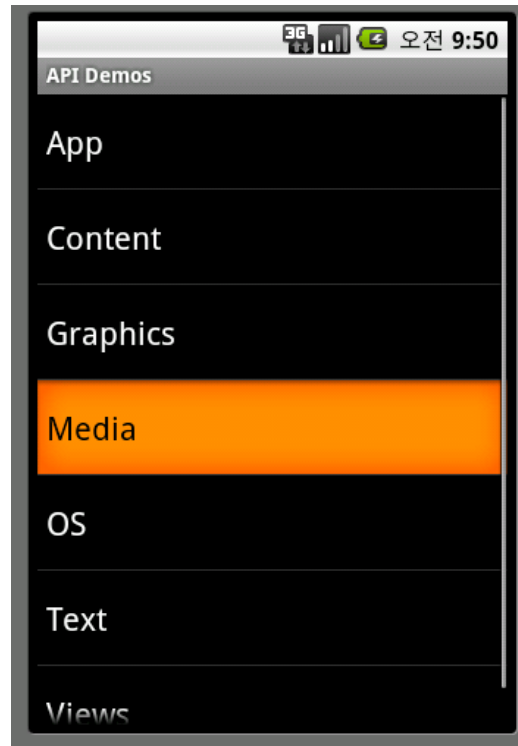
TableLayout

- ❑ html 테이블 형태
- ❑ 각 행은 <TableRow>
- ❑ 행속의 컬럼 번호 설정: android:layout_column
- ❑ 구분선 지정가능
 - ❑ <View android:layout_height="2px"
android:background="#FF0000"/>
- ❑ 컬럼폭 늘임 설정 : android:stretchColumns
- ❑ 컬럼폭 줄임 설정 : android:shrinkColumns
- ❑ 컬럼 숨기기 : android:collapseColumns



ScrollView

- ❑ 상하로 화면 스크롤
- ❑ 내용이 매우 길어서 한 화면에 표시 하지 못할 경우 ScrollView에 감싸준다



2.5 데이터 집합 표시 뷰의 활용

- ❑ 데이터 집합 표시 뷰의 종류
- ❑ ArrayAdapter
- ❑ ArrayAdapter와 뷰의 활용



데이터 집합 표시 뷰의 종류

☐ 위젯 계열

- ☐ Spinner : 콤보박스 형태
- ☐ Gallery : 갤러리 형태



☐ 레이아웃 계열

- ☐ ListView : 리스트 형태
- ☐ GridView : 그리드 형태



Adapter

- ❑ 서로 관련 없는 API 에 공통의 인터페이스를 제공
- ❑ 종류
 - ❑ CursorAdapter
 - ❑ DB쿼리결과 또는 콘텐츠프로바이더로부터의 내용을 화면에 표시하도록 함
 - ❑ SimpleAdapter
 - ❑ XML 리소스에 들어 있는 내용 변환
 - ❑ ActivityAdapter, ActivityIconAdapter
 - ❑ 특정 인텐트를 사용해 실행될 액티비티의 이름이나 아이콘 사용
 - ❑ ArrayAdapter
 - ❑ array나 java.util.List에 저장된 data를 위한 adapter



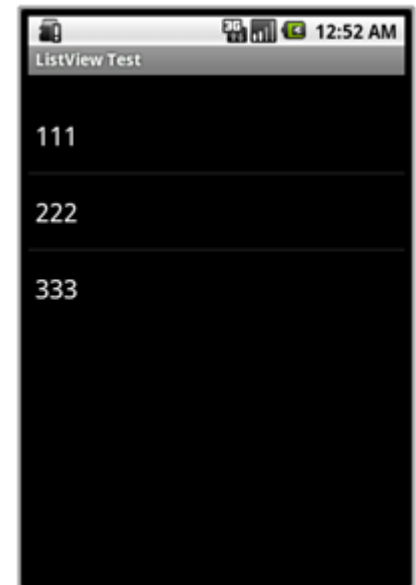
ArrayAdapter

❑ 역할

- ❑ 자바배열, java.util.List 인스턴스를 사용해 위젯의 데이터를 표시하도록 함
- ❑ getView() 메소드를 오버라이드하면 문자열 대신에 다른 위젯으로 표시 가능

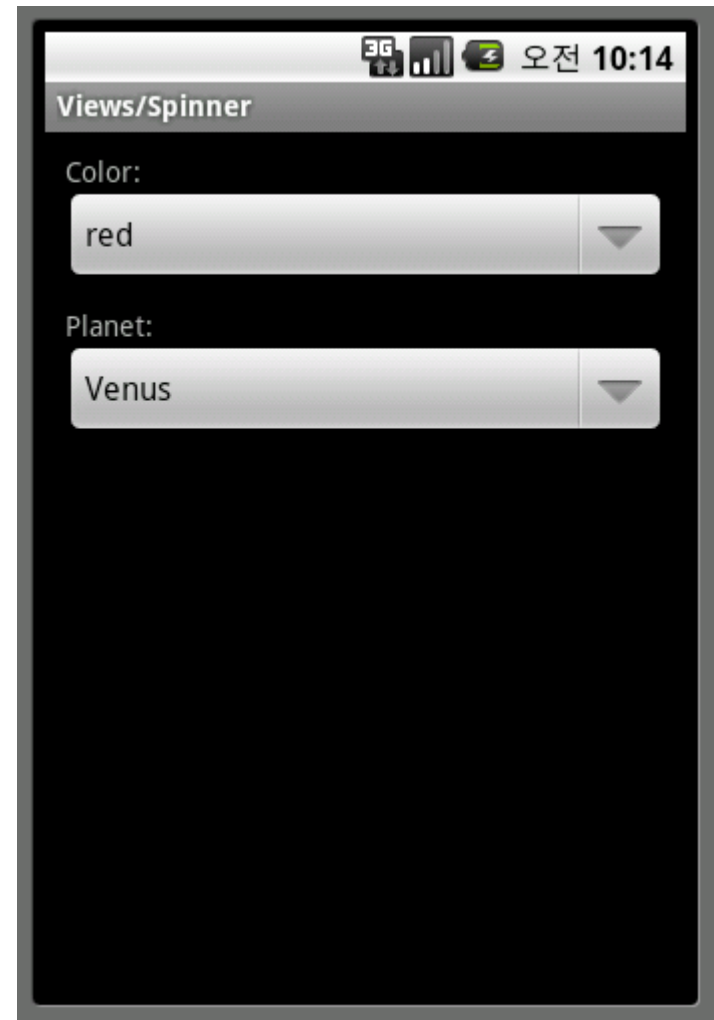
❑ 사용방법

```
String items[] = { "111", "222", "333" };  
ArrayAdapter <String> arrAdt = new ArrayAdapter <String> (this,  
                                                         android.R.layout.simple_list_item_1, items);  
ListView 인스턴스.setAdapter(arrAdt);
```



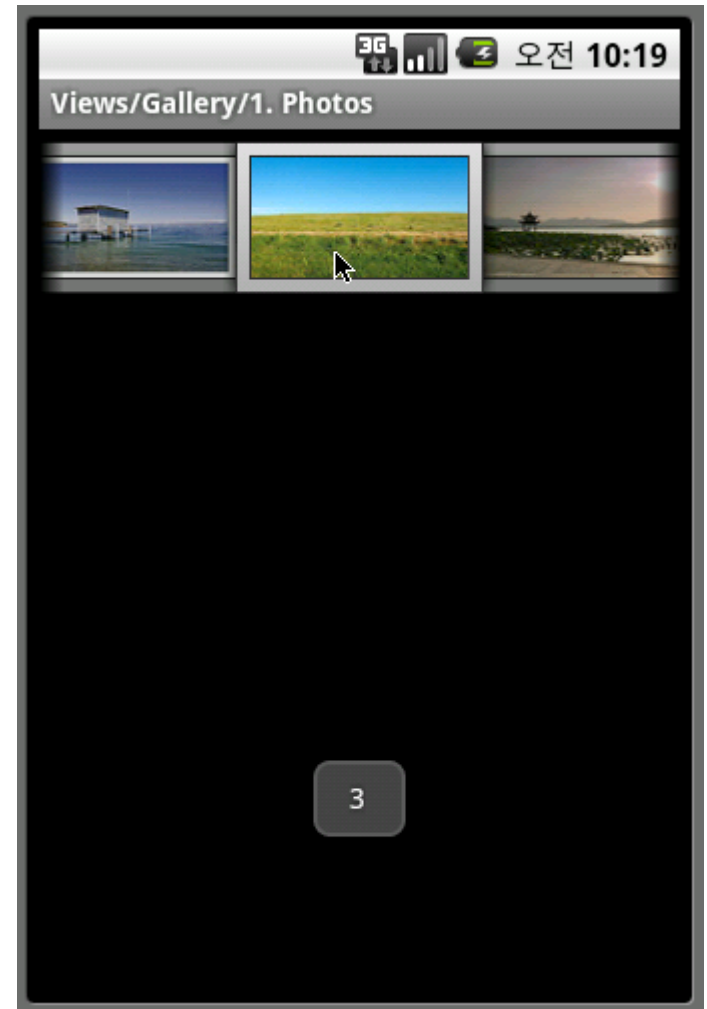
Spinner

- ❑ 콤보박스 형태
- ❑ 선택여부
 - ❑ `setOnItemSelectedListener()`
- ❑ 화면에 표시할 위젯 ID 지정
 - ❑ `setDropDownViewResource()`
- ❑ 화살표 모양 표시
 - ❑ `drawSelectorOnTop`



Gallery

- ☐ 이미지 미리보기
- ☐ 가로형태의 리스트
- ☐ 선택된 항목이 하이라이트
- ☐ 속성
 - ☐ spacing: 여백
 - ☐ spinnerSelector
 - ☐ 선택된 내용 표시 방법
 - ☐ drawSelectorOnTop
 - ☐ 선택 상태를 표시하는 기능이 본문항목을 그리기 전(false)또는 후(true)에 동작할지를 지정
 - ☐ true일 경우 투명도 지정



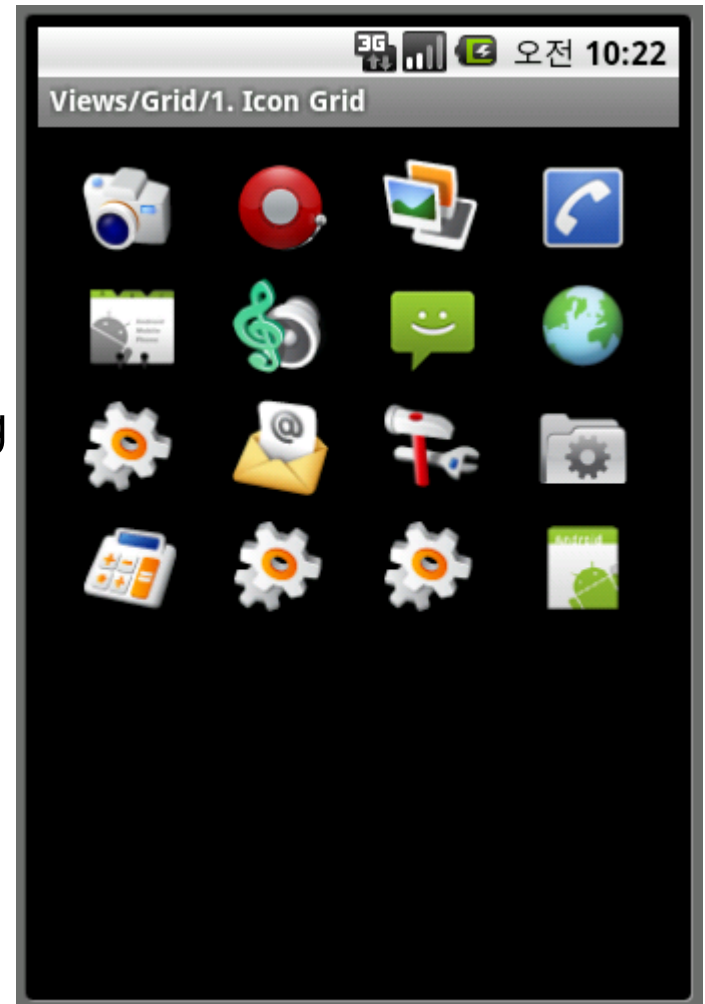
ListView

- ❑ 구현 클래스
 - ❑ ListView를 Activity에 추가
 - ❑ ListActivity 사용
- ❑ 구현방법
 - ❑ setListAdapter
 - ❑ ArrayAdapter연결
 - ❑ setOnItemSelectedListener
 - ❑ 선택된 항목 감지



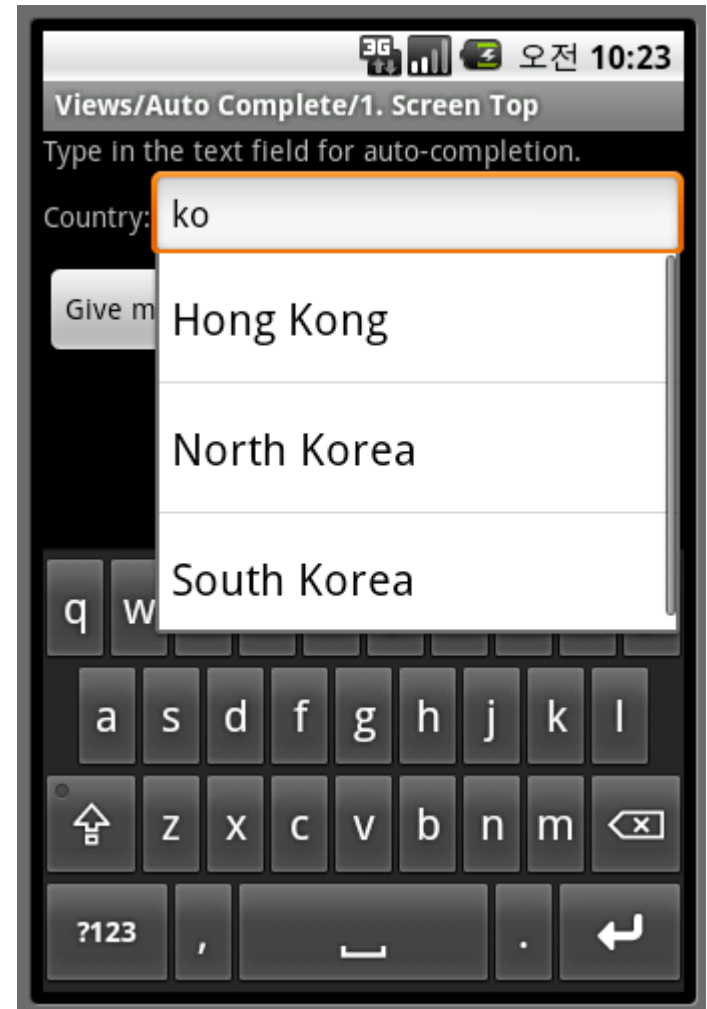
GridView

- ❑ 2차원 그리그 모양
- ❑ 속성
 - ❑ `columnWidth` : 컬럼폭
 - ❑ `numColumns` : 컬럼 개수
 - ❑ `auto_fit` 자동 지정
 - ❑ `verticalSpacing`, `horizontalSpacing`
 - ❑ 그리드 내부 항목간 여백
 - ❑ `stretchMode`
 - ❑ `auto_fit`일 경우
 - ❑ 컬럼내부 여유공간 처리 방법
 - ❑ `columnWidth`
 - ❑ `spacingWidth`



AutoCompleteTextView

- ❑ EditText + Spinner
- ❑ 입력 단어에 대한 추천 리스트
 - ❑ onTextChanged



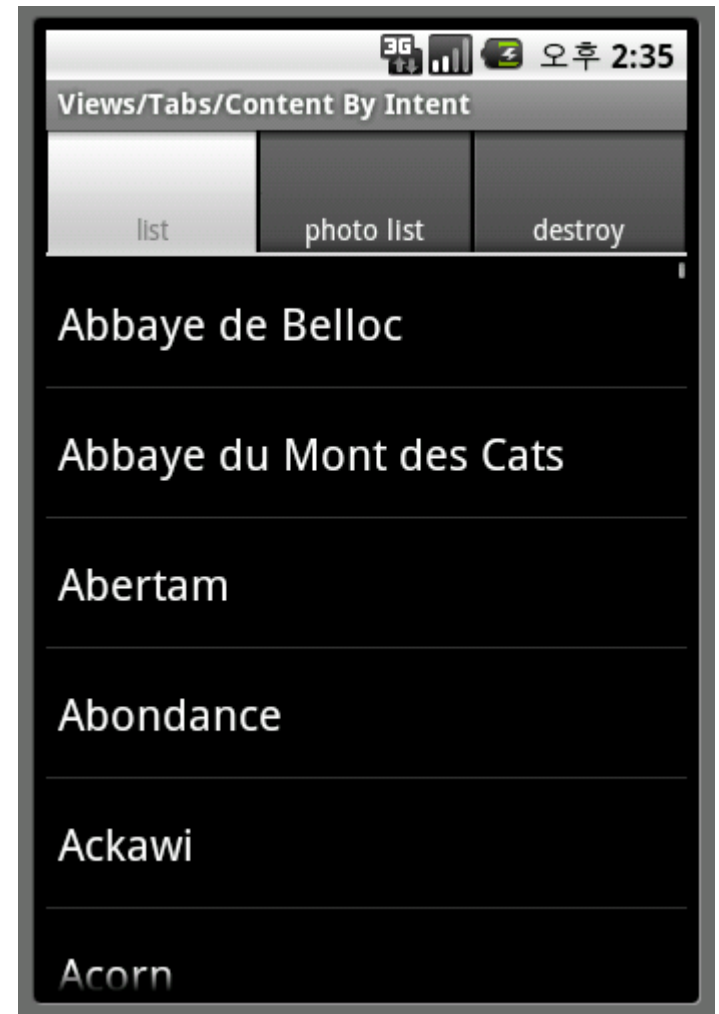
2.6 화면 네비게이션

- ☐ 탭 화면 구성
- ☐ 페이징 화면
- ☐ 메뉴
 - ☐ 옵션 메뉴
 - ☐ 컨텍스트 메뉴
 - ☐ XML 메뉴 인플레이션



탭 화면 구성 요소

- ❑ TabHost
 - ❑ 탭버튼, 탭컨텐츠에 대한 컨테이너
 - ❑ TabWidget (탭버튼)
 - ❑ 문자열 또는 아이콘 설정 가능
 - ❑ id값은 tabs로 지정
 - ❑ FrameLayout 상단에 여백지정
 - ❑ `paddingTop="60px"`
 - ❑ FrameLayout(탭컨텐츠)
 - ❑ 탭 컨텐츠 관리



탭추가

□ 정적 탭 추가

- setContent() : 레이아웃 아이디 설정
- setIndicator() : 탭버튼 표시 내용 설정

□ 동적 탭 추가

- setContent 호출시 TabHost.TabContentFactory 인스턴스 전달
- TabHost.TabContentFactory에서는 createTabContent() 호출



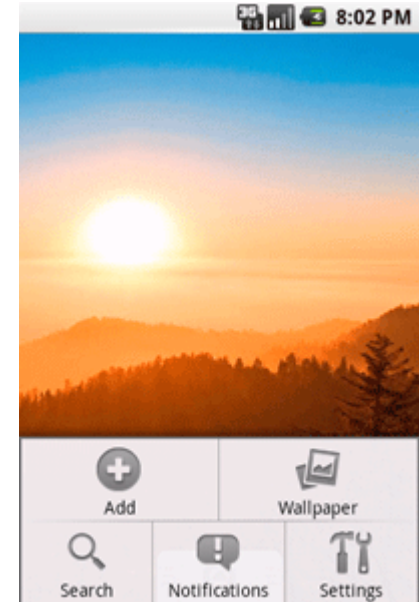
페이징 화면

- ❑ ViewPager
 - ❑ FrameLayout 상속
 - ❑ 화면을 책장 페이지 넘기도록 함
 - ❑ showNext() 함수 호출
 - ❑ 자동 넘기기
 - ❑ setFlipInterval()
 - ❑ startFlipping()



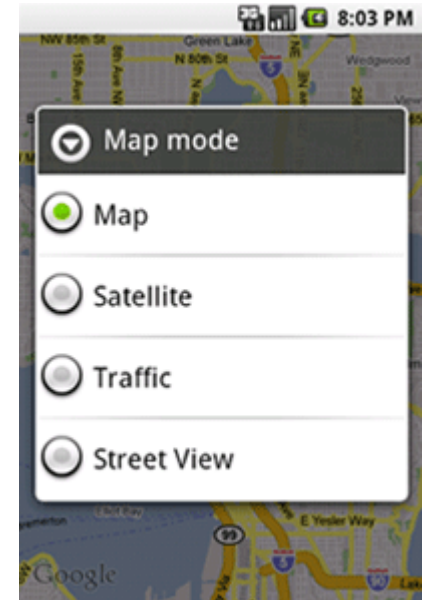
옵션 메뉴

- ❑ 메뉴버튼을 눌렀을 때 나타남
- ❑ 옵션 메뉴 생성
 - ❑ onCreateOptionsMenu() 함수를 오버라이딩
 - ❑ Menu인스턴스의 add() 메소드 호출
 - ❑ 그룹ID (메뉴 그룹 지정)
 - ❑ 선택 항목 ID
 - ❑ onOptionsItemSelected()에서 선택항목지정
 - ❑ 순서 ID
 - ❑ 메뉴명
- ❑ 단축키 설정
 - ❑ 알파벳 : setAlphabeticShortcut()
 - ❑ 숫자 : setNumericShortcut()



컨텍스트 메뉴

- ❑ 메뉴가 연결된 위젯을 터치하면 나타나는 메뉴
 - ❑ `registerForContextMenu()`에서 연결 위젯 설정
- ❑ 메뉴 생성 방법
 - ❑ `onCreateContextMenu()` 함수를 오버라이딩
- ❑ 선택된 항목 넘겨 받기
 - ❑ `onContextItemSelected()`



XML 메뉴 적용

- ❑ XML을 파싱해 뷰를 생성하는 작업으로 메뉴인플레이션이라 함
- ❑ 작업 단계
 - ❑ XML 메뉴 정의
 - ❑ <menu> - <item> 또는 <group>
 - ❑ 엘리먼트 속성
 - ❑ id, title, icon, orderInCategory, enabled, visible
 - ❑ 단축키
 - ❑ alphabeticShortcut, numericShortcut
 - ❑ 메뉴 인플레이션
 - ❑ onCreateOptionsMenu에서
 - ❑ new MenuInflater()



2.7 UX향상을 위한 기타 테크닉

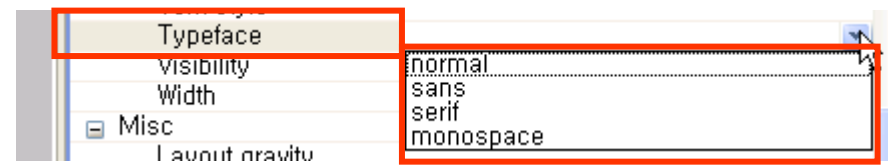
- ☐ 폰트
- ☐ 스타일과 테마
- ☐ 진행상태 표시
- ☐ 리소스 처리
- ☐ 화면회전/고정



폰트

□ 기본 글꼴

- sans, serif, monospace
- Ascender에서 제공한 Droid글꼴
- `typeface="sans, serif, monospace` 중 하나 지정



□ 외부 글꼴

- 프로젝트 assets 디렉토리에 폰트파일 복사
- `Typeface face = Typeface.createFromAsset(getAssets(), "xxx.ttf");`
- `view.setTypeface(face);`

□ 주의 사항

- 폰트의 용량
- 안드로이드 호환성 검증
- 한글 글꼴 지원 여부 확인



스타일과 테마

□ 스타일

- 레이아웃 XML 파일에 적용할 수 있는 포매팅 속성의 집합
- CSS처럼 특정 텍스트 크기와 컬러를 지정할 특정 뷰에 적용

□ 테마

- 애플리케이션 내의 모든 액티비티 또는 하나의 액티비티 단위로 적용할 수 있는 포매팅 속성의 집합



스타일 적용

□ res/values 디렉토리에 styles.xml 생성

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="SpecialText" parent="@style/Text">
        <item name="android:textSize">18sp</item>
        <item name="android:textColor">#008</item>
    </style>
</resources>
```

□ View의 style 속성에 적용

```
<EditText id="@+id/text1" style="@style/SpecialText"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Hello, World!" />
```



테마 적용

□ 테마파일 생성

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<style name="CustomTheme">
<item name="android:windowNoTitle">true</item>
<item name="windowFrame">@drawable/screen_frame</item>
<item name="windowBackground">@drawable/screen_background_white
</item>
<item name="panelForegroundColor">#FF000000</item>
<item name="panelTextColor">?panelForegroundColor</item>
<item name="panelTextSize">14</item>
<item name="menuItemTextColor">?panelTextColor</item>
<item name="menuItemTextSize">?panelTextSize</item>
</style>
</resources>
```

□ 안드로이드 매니페스트 내의 <application>과 <activity> 엘리먼트에서 지정

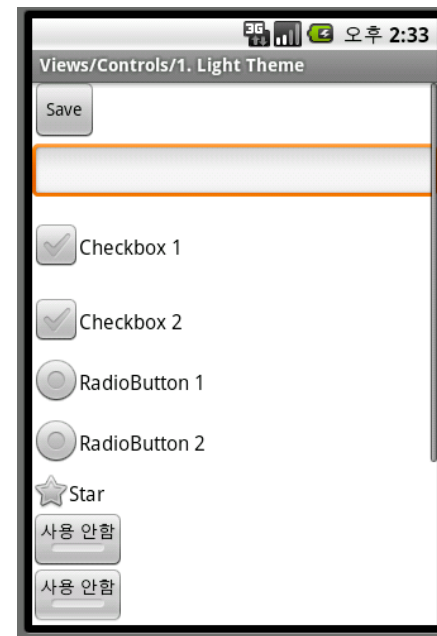
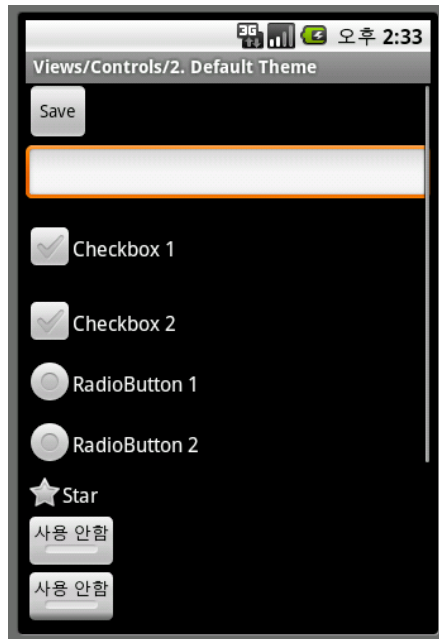
```
<application android:theme="@style/CustomTheme">
<activity android:theme="@android:style/Theme.Dialog">
```



런타임 테마 적용

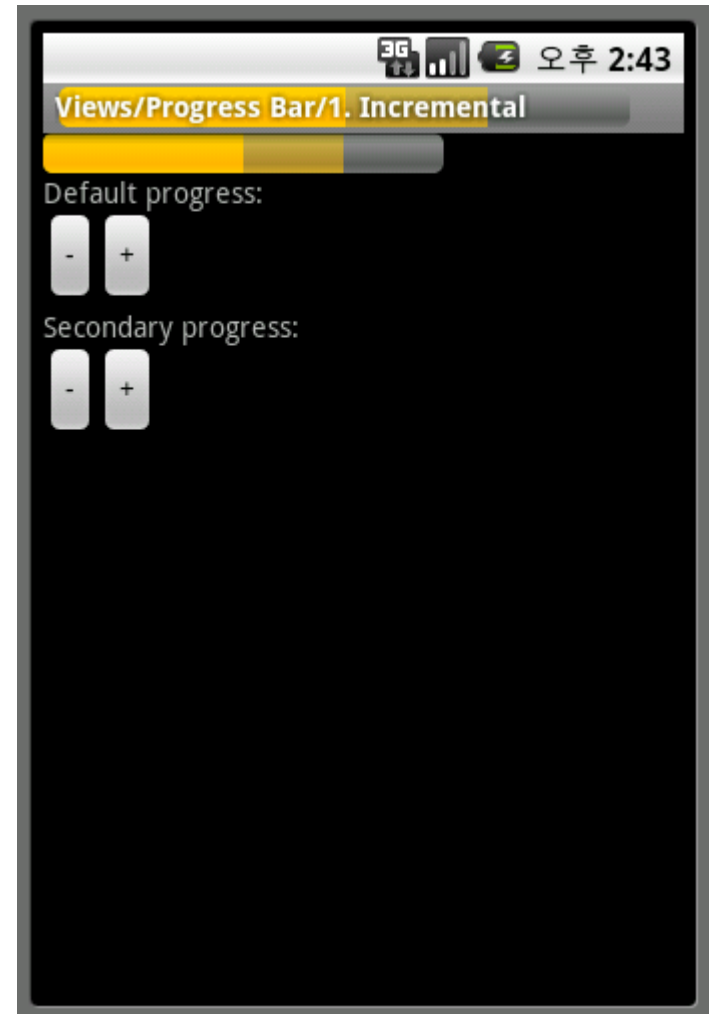
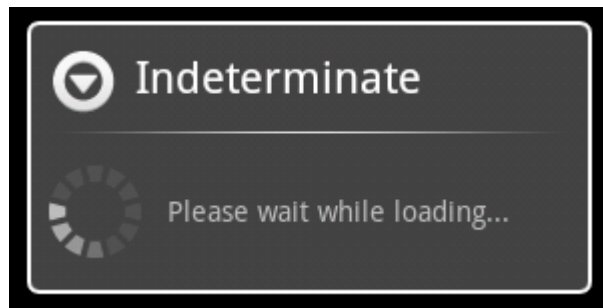
□ setTheme() 함수 호출

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState); ...  
    setTheme(android.R.style.Theme_Light);  
    setContentView(R.layout.linear_layout_3);  
}
```



진행상태 표시

- ProgressBar
 - 0 부터 시작되는 정수값으로 현재의 진행정도 설정
 - `setProgress()` 에서 값 설정
 - 진행정도는 `incrementProgressBy()` 사용
 - 명확하지 않는 작업은 `setIndeterminate()` 설정



리소스 처리

- ❑ 코드가 아닌 것들은 외부 리소스화

- ❑ res의 하위 폴더

- ❑ res/anim : 사용자 인터페이스에 들어갈 애니메이션 처리

- ❑ 뷰를 회전하고, 이동하고, 늘어뜨리고 페이드 효과를 주는데 사용하는 트윈드 애니메이션

- ❑ 연속된 드로어블 이미지들을 표시할수있는 프레임 바이 프레임 애니메이션

- ❑ res/drawable : 사용자 인터페이스에 들어갈 비트맵이미지

- ❑ res/values : 문자열, 색상, 치수 등

- ❑ 디렉토리명을 id값으로 설정해 다국어 처리 (res/values-ko, res/values-en 등)

- ❑ res/xml : XML데이터



리소스 처리

- ❑ 리소스 활용
 - ❑ static 클래스 R을 사용해 접근
 - ❑ R은 프로젝트를 컴파일하면 자동생성
 - ❑ 리소스 타입 각각을 위한 static 하위 클래스들을 존재

```
R.resource_type.resource_name  
android.R.resource_type.resource_name  
setContentView(R.layout.main_screen);
```

```
attribute="@[packagename:]resourcetype/resourceidentifier"
```

```
CharSequence httpError = getString(android.R.string.httpErrorBadUrl);
```



리소스 처리

- ❑ 다양한 환경 지원을 위한 리소스 처리
 - ❑ 언어: 두개의 소문자로 ISO 639-1언어코드를 사용 (ex: en)
 - ❑ 지역: 소문자"r" 뒤에 대문자로 된 ISO 3166-1-alpha-2 언어코드
 - ❑ 화면방향 - port(세로),land(가로), square(정사각형)
 - ❑ 화면 픽셀 밀도 - 인치당 도트수(dpi)로 표현 픽셀 밀도
 - ❑ 터치스크린 타입 - nottouch, stylus,finger
 - ❑ 키보드 사용 - keysexposed, keyshidden
 - ❑ 키보드 입력 타입 - nokeys, qwerty, 12key
 - ❑ UI 탐색 타입- notouch, dpad, trackball, wheel
 - ❑ 화면해상도 - 픽셀로 표현된 화면 해상도로서, 가장 큰치수가 먼저
- ❑ 예) res/layout-kr-rKO-port-nottouch-qwerty-640x480



화면회전/고정

❑ 화면회전

❑ AndroidManifest.xml에 android:configChanges 속성 추가

❑ keyboardHidden, orientation

- ❑ 키보드가 슬라이드 되면서 회전되는 경우와 가속도계를 통해 방향을 확인해 회전되는 경우

❑ Activity 내부에 onConfigurationChanged() 구현

❑ 화면고정

❑ AndroidManifest.xml에 screenOrientation="portrait" 또는 "landscape"로 지정



수고하셨습니다 ^^/



by OkGosu.Net