데이터 입력 화면 만들기

2017-08-29

이승진

**학습목표**

액티비티를 하나 더 구현한다.

메뉴를 구현한다.

메뉴를 눌러서 다른 액티비티 화면으로 전환하는 기능을 구현한다.

데이터 입력 화면을 구현한다.

입력 오류 메시지를 화면에 표시하는 기능을 구현한다.

**목차**

[1. E02Views 프로젝트 2](#_Toc491873665)

[1) 구현할 액티비티 2](#_Toc491873666)

[2) 프로젝트 생성 2](#_Toc491873667)

[2. 메뉴 만들기 5](#_Toc491873668)

[1) 문자열 리소스 만들기 5](#_Toc491873669)

[2) 메뉴 리소스 파일 생성 6](#_Toc491873670)

[3) 메뉴의 id 8](#_Toc491873671)

[4) 한글 깨짐 문제 해결 9](#_Toc491873672)

[5) 메뉴 메소드 구현 11](#_Toc491873673)

[6) 실행 13](#_Toc491873674)

[3. 회원 가입 화면 만들기 14](#_Toc491873675)

[1) 새 Activity 생성 14](#_Toc491873676)

[2) 생성된 파일 15](#_Toc491873677)

[4. 회원 가입 화면 레이아웃 16](#_Toc491873678)

[1) 로그인 아이디 TextView 추가 16](#_Toc491873679)

[2) activity\_signup.xml - 로그인 아이디 입력 EditText 추가 18](#_Toc491873680)

[3) 비밀번호 TextView 추가 23](#_Toc491873681)

[4) 비밀번호를 입력할 EditText 추가 24](#_Toc491873682)

[5) 비밀번호 확인 TextView 추가 25](#_Toc491873683)

[6) 비밀번호 확인 EditText 추가 26](#_Toc491873684)

[7) 이메일 TextView 추가 27](#_Toc491873685)

[8) 이메일 입력 EditText 추가 28](#_Toc491873686)

[9) 버튼 추가 29](#_Toc491873687)

[5. 회원 가입 Java 코드 30](#_Toc491873688)

[1) SignupActivity.java 30](#_Toc491873689)

[2) Editable 인터페이스 32](#_Toc491873690)

[3) 화면 전환 기능 구현 33](#_Toc491873691)

[4) 결과 소스코드 34](#_Toc491873692)

[6. 메모장 화면 만들기 35](#_Toc491873693)

[1) 새 Activity 생성 35](#_Toc491873694)

[2) 메모 화면 레이아웃 36](#_Toc491873695)

[7. 메모장 Java 코드 40](#_Toc491873696)

[1) MemoActivity.java 40](#_Toc491873697)

[2) 화면 전환 기능 구현 41](#_Toc491873698)

[3) 화면 제목 변경하기 42](#_Toc491873699)

[4) 결과 소스코드 43](#_Toc491873700)

# E02Views 프로젝트

## 구현할 액티비티

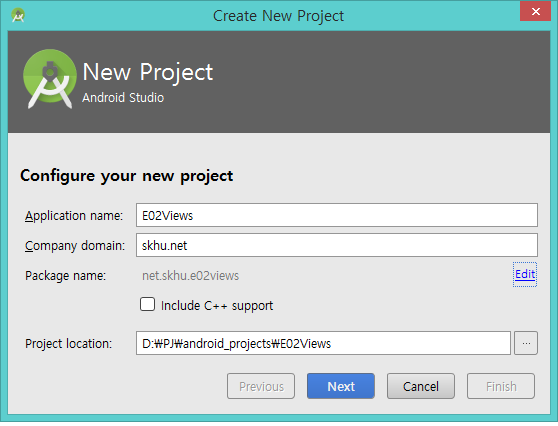
MainActivity : 메인 화면과 메뉴

SignupActivity : 회원가입 화면

## 프로젝트 생성

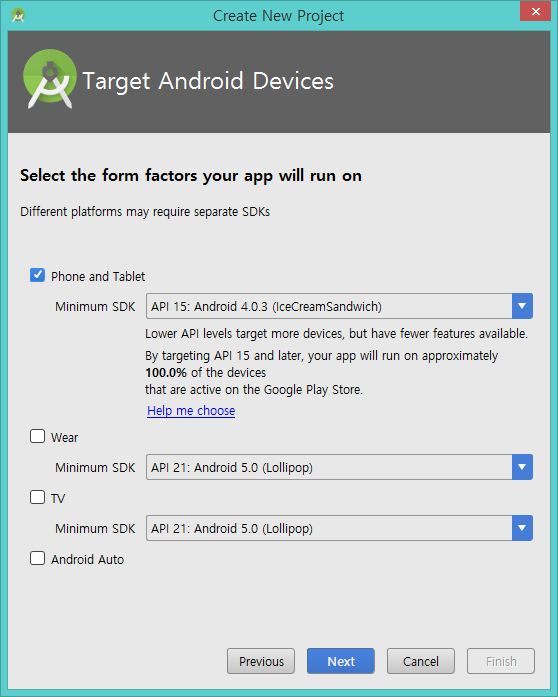
메뉴: File - New - New Project

### New Project 대화상자

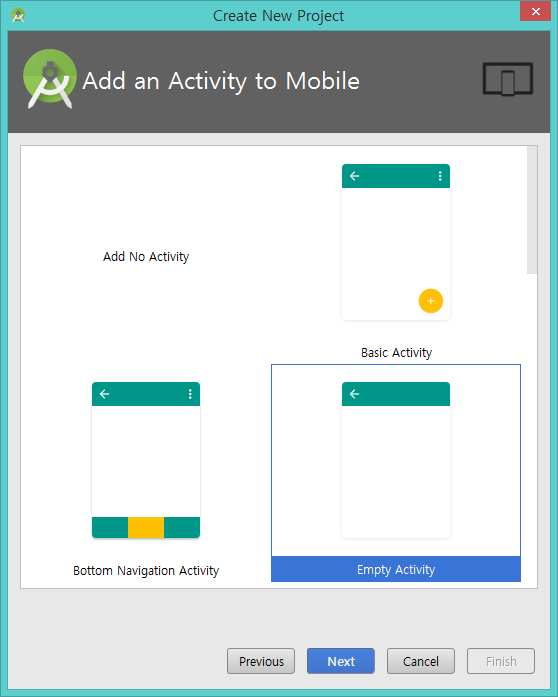


|  |  |
| --- | --- |
| Application name: | E02Views |
| Company Domain: | skhu.net |
| Project Location: | d:/pj/android\_projects/E02Views |

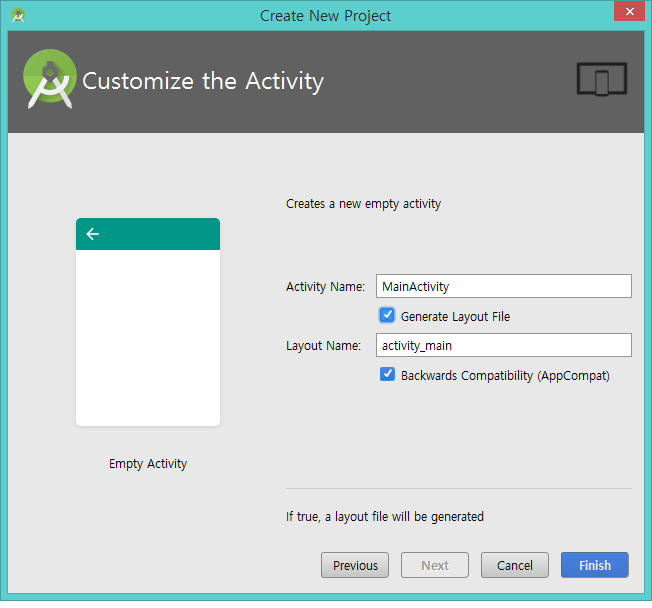
위와 같이 입력하고 Next를 계속 눌러서 진행하자.



Next 버튼 클릭

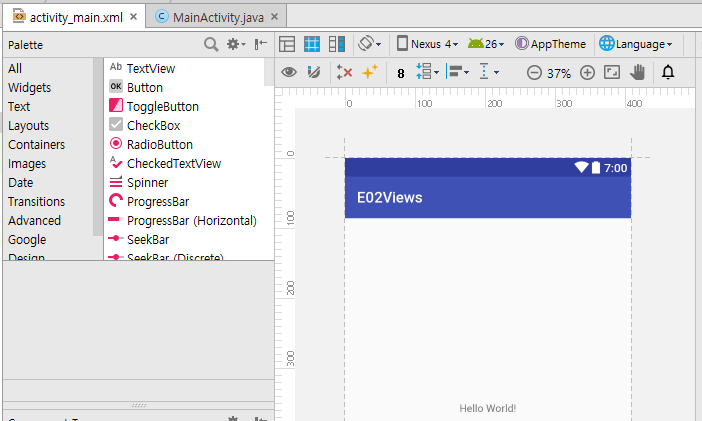


Next 버튼 클릭



Finish 버튼 클릭

한참을 기다리면 프로젝트가 생성되고 다음과 같이 편집창이 열리게 된다.



# 메뉴 만들기

## 문자열 리소스 만들기

한국어, 영어, 일본어, 중국어 등 다국어를 지원하는 앱을 만들기 위해서,

화면에 표시해야 하는 문자열은 모두 문자열 리소스(string resource)로 만들어야 한다.

문자열 리소스는 모두 문자열 리소스 XML 파일에 저장된다.

나중에 개발이 마무리되어갈 때, 문자열 리소스 XML 파일을 여러 개 (영어 버전, 일어 버전, 중국어 버전) 만들어 주기만 하면, 다국어 지원 기능의 구현이 완료된다.

스마트 폰의 기본 언어가 바뀌면, 여러 문자열 리소스 XML 파일들 중에서, 선택된 언어의 문자열 리소스 XML 파일이 앱의 기본 값으로 사용되기 때문에, 앱의 화면에 표시되는 문자열도 저절로 그 언어도 바뀌게 된다.

### 

화면 왼쪽 프로젝트 창에서 res/values/strings.xml 파일이 문자열 리소스 XML 파일이다.

이 파일을 더블 클릭하여 열자.

**strings.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <resources>  <string name="app\_name">E02Views</string>  <string name="action\_signUp">회원가입</string>  <string name="action\_memo">메모장</string>  </resources> |

이 파일의 내용을 위와 같이 수정하자.

<string> 태그의 name 애트리뷰트 값이 문자열 리소스 항목의 ID 이다.

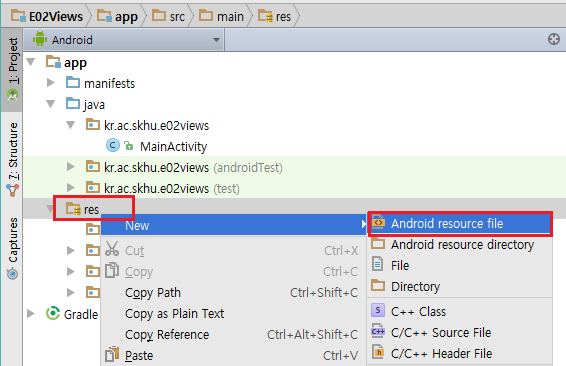
<string> 태그 사이의 내용이 문자열 리소스 항목의 내용이다.

문자열 리소스

|  |  |
| --- | --- |
| ID | 화면에 표시될 내용 |
| action\_signUp | 회원가입 |
| action\_memo | 메모장 |

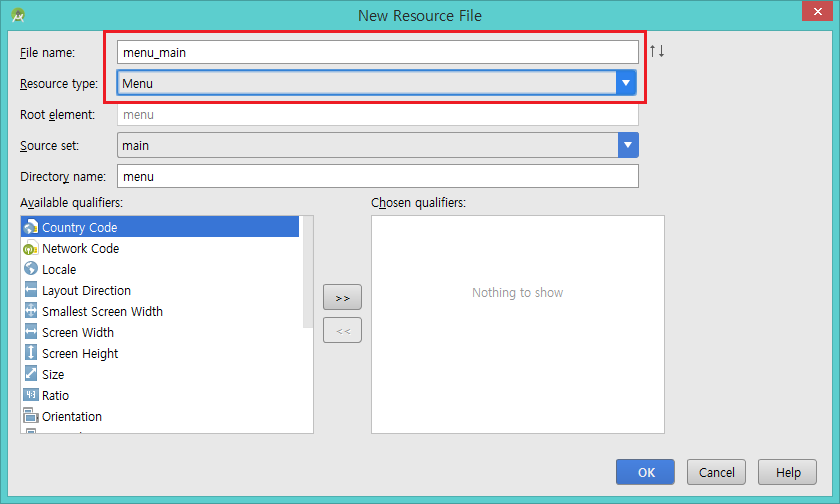
string.xml 파일을 저장하고 닫자

## 메뉴 리소스 파일 생성



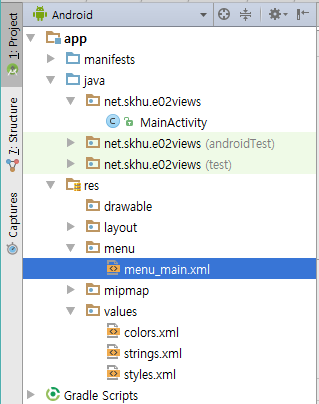
화면 왼쪽 Project 창에서 res 폴더를 마우스 오른쪽 버튼으로 클릭하고

메뉴에서 New - Android resource file 클릭



액티비티의 이름이 MainActivity 이므로, 메뉴 파일의 이름도 menu\_main 이라고 입력한다.

Ok 버튼 클릭.



res/menu/menu\_main.xml 파일이 생성된 메뉴 리소스 XML 파일이다.

이 파일을 더블 클릭하여 열자.

아래쪽의 text 탭을 눌러서 텍스트 편집창을 열자.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | *<?*xml version="1.0" encoding="utf-8"*?>*  <menu xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto">  <item android:id="@+id/action\_signUp"  android:title="@string/action\_signUp" app:showAsAction="always" />  <item android:id="@+id/action\_memo"  android:title="@string/action\_memo" app:showAsAction="always" />  </menu> |

위 XML 파일에서 <item> 태그가 메뉴 항목 하나이다.

<item> 태그에는 android:id, android:title, android:app 애트리뷰트가 있다.

android:id 애트리뷰트 값은 메뉴 항목의 id 이다.

리소스 XML 파일에서 어떤 항목의 id는 다음과 같은 형태이다.

"@+id/actoin\_signUp", "@+id/action\_memo"

"@id/**항목의\_ID**"

android:title 애트리뷰트 값은 메뉴 항목에 표시될 문자열이다.

이 애트리뷰트의 값은 문자열 리소스의 이름(name) 이다.

리소스 XML 파일에서 문자열 리소스의 이름은 다음과 같은 형태이다.

"@string/action\_signUp", "@string/action\_memo"

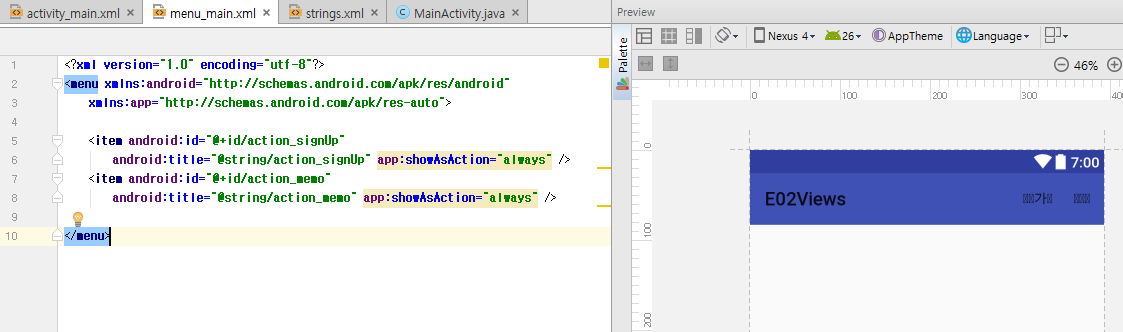
"@string/**문자열\_리소스\_이름**"

android:showAsAction 애트리뷰트 값은 메뉴 항목을 어떻게 표시할 것인지 선택하는 값이다.

이 애트리뷰트 값은 다음과 같다.

|  |  |
| --- | --- |
| android:showAsAction | 설명 |
| always | 메뉴를 언제나 표시한다. |
| ifRooms | 메뉴를 표시할 공간이 충분하면 표시하고, 그렇지 않으면 ... 으로 표시한다. |
| never | 메뉴를 ... 으로 표시한다. |

위 XML 파일에서 android:showAsAction 애트리뷰트 값을 변경해 보자. 그 결과를 화면 오른쪽 Preview 창에서 즉시 확인할 수 있다.



menu\_main.xml 파일을 저장하고 닫자.

## 메뉴의 id

menu\_main.xml 파일에 포함된 메뉴의 아이디는 다음과 같다.

"@+id/action\_signUp"

"@+id/action\_memo"

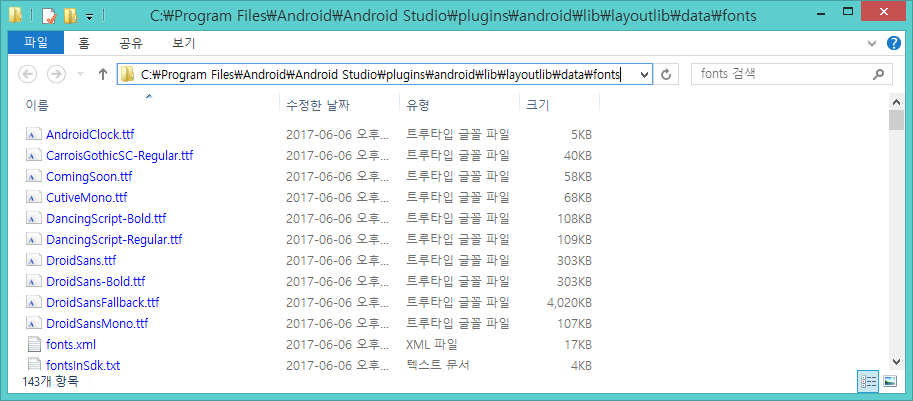
이 메뉴 id를 Java 소스 코드에서 사용할 때에는 다음과 같은 형태이어야 한다.

R.id.action\_signUp

R.id.action\_memo

XML과 Java의 문법이 서로 다르기 때문에, id 값의 형태도 서로 다르다.

## 한글 깨짐 문제 해결



|  |
| --- |
| C:\Program Files\Android\Android Studio\plugins\android\lib\layoutlib\data\fonts |

탐색 창에서 위 폴더에 들어간다.

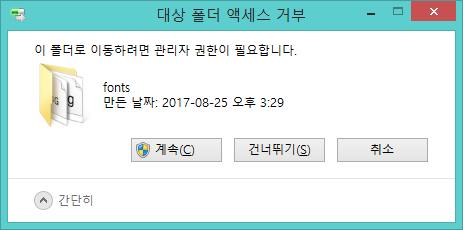
위 경로를 탐색창의 주소칸에 붙여 넣으면 이 폴더로 들어가게 된다.

Android Studio 2.3.3 인 경우에

강의자료 웹폴더의 소스코드 폴더의 fonts.xml 파일을 위 탐색창에 드롭한다.

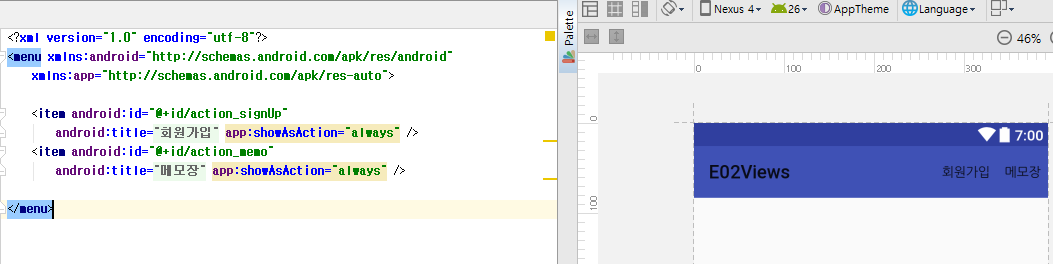


대상 폴더의 파일 덮어쓰기 클릭



계속 클릭

Android Studio 재시작



한글이 정상적으로 보임.

## 메뉴 메소드 구현

메뉴 메소드는 액티비티 클래스에 구현해야 한다.

모든 액티비티 클래스의 부모 클래스는 Activity 클래스이다.

이 클래스에 메뉴 메소드가 이미 구현되어 있다.

자식 액티비티 클래스에서 이 메소드를 재정의(override) 해야 한다.

### onCreateOptionsMenu 메소드

메뉴가 생성되어야 할 때 저절로 호출되는 메소드이다.

액티비티 클래스에서 이 메소드를 재정의(override)해서, 메뉴 생성 코드를 구현해야 한다.

### onOptionsItemSelected 메소드

메뉴 항목이 클릭되면 저절로 호출되는 메소드이다.

액티비티 클래스에서 이 메소드를 재정의(override)해서, 메뉴가 클릭되었을 때 해야할 일을 구현해야 한다.

MainActivity.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | **package** net.skhu.e02views;  **import** android.support.v7.app.AppCompatActivity;  **import** android.os.Bundle;  **import** android.view.Menu;  **import** android.view.MenuItem;  **import** android.widget.Toast;  **public class** MainActivity **extends** AppCompatActivity {  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  }  @Override  **public boolean** onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.***menu\_main***, menu);  **return true**;  }  @Override **public boolean** onOptionsItemSelected(MenuItem item) {  **int** id = item.getItemId();  **if** (id == R.id.***action\_signUp***) {  Toast.*makeText*(**this**, **"회원가입 메뉴 클릭"**, Toast.***LENGTH\_SHORT***).show();  **return true**;  } **else if** (id == R.id.***action\_memo***) {  Toast.*makeText*(**this**, **"메모장 메뉴 클릭"**, Toast.***LENGTH\_SHORT***).show();  **return true**;  }  **return super**.onOptionsItemSelected(item);  }  } |

|  |
| --- |
| @Override  **public boolean** onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.***menu\_main***, menu);  **return true**;  } |

액티비티의 메뉴를 생성하는 메소드이다.

메뉴가 달려 있는 액티비티에는 이 메소드가 반드시 있어야 한다.

이 메소드가 없으면 메뉴가 보이지 않을 것이다.

이 메소드의 코드를 자세히 설명하는 것은 뒤로 미루자.

XML 리소스 파일의 내용대로 Java 객체를 자동 생성해주는 기능을

안드로이드에서 inflation 이라고 부른다.

이런 일을 해주는 객체를 inflator 라고 부르고, 메소드의 이름은 inflate 이다.

inflate 메소드의 파라미터는 XML 리소스 파일의 아이디 이다.

줄19의 R.menu.menu\_main 부분을 주목하자.

이것은 res/menu/menu\_main.xml 메뉴 리소스 파일의 id 이다.

줄14의 R.layout.activity\_main 부분을 주목하자.

이것은 res/layout/activity\_main.xml 레이아웃 리소스 파일의 id 이다.

|  |
| --- |
| @Override **public boolean** onOptionsItemSelected(MenuItem item) {  **int** id = item.getItemId();  **if** (id == R.id.***action\_signUp***) {  Toast.*makeText*(**this**, **"회원가입 메뉴 클릭"**, Toast.***LENGTH\_SHORT***).show();  **return true**;  } **else if** (id == R.id.***action\_memo***) {  Toast.*makeText*(**this**, **"메모장 메뉴 클릭"**, Toast.***LENGTH\_SHORT***).show();  **return true**;  }  **return super**.onOptionsItemSelected(item);  } |

메뉴 항목이 클릭되면 자동으로 즉시 호출되는 메소드이다.

클릭된 메뉴 객체가 이 메소드의 파라미터로 전달된다.

**int** id = item.getItemId();

클릭된 메뉴의 id 값을 꺼낸다.

이 값은 R.id.action\_signUp, R.id.actoin\_memo 둘 중의 하나일 것이다.

Toast.*makeText*(**this**, **"회원가입 메뉴 클릭"**, Toast.***LENGTH\_SHORT***).show();

화면에 간단한 메시지를 잠시 보여주는 코드이다.

### Toast 메시지

화면에 짧은 문자열 메시지를 잠깐 보여줄 때 Toast 클래스를 사용한다.

Toast.makeText(this, "메시지", Toast.LENGTH\_LONG).show();

makeText는 Toast 클래스의 public static 메소드이다.

이 메소드의 첫째 파라미터 this는 현재 액티비티 객체의 this 이다.

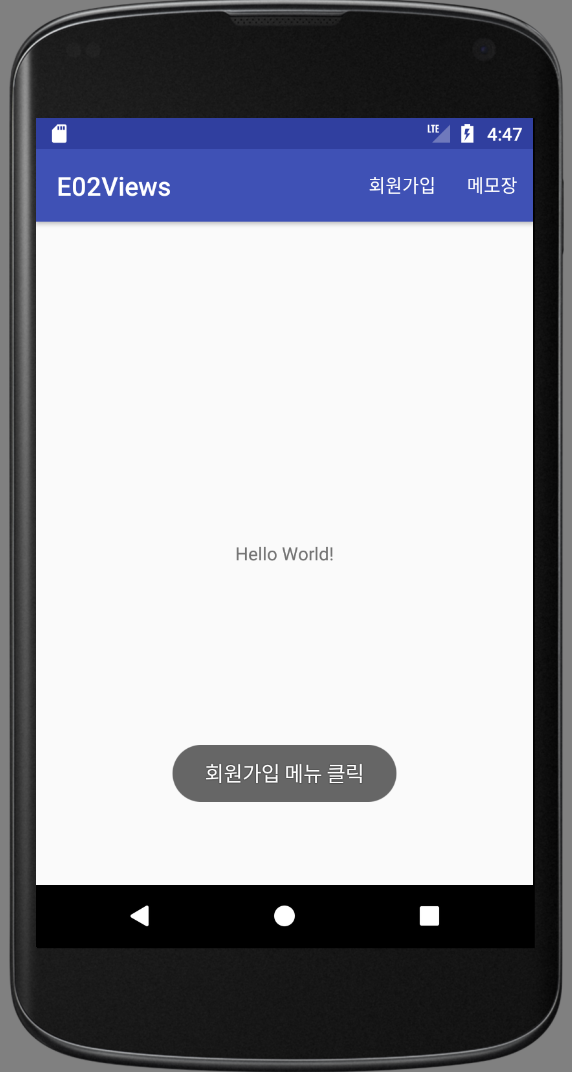
둘째 파라미터는 화면에 표시할 문자열이다.

셋째 파라미터는 LENGTH\_SHORT, LENGTH\_LONG 둘 중의 하나이다.

LENGTH\_SHORT : 메시지가 보여지는 시간이 짧다.

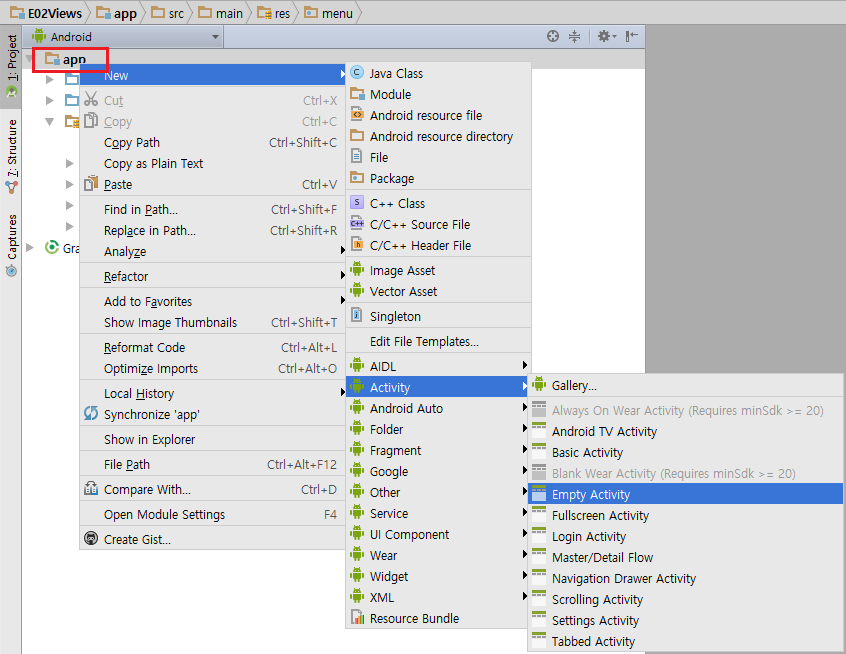
LENGTH\_LONG : 좀 더 오래 메시지를 보여준다.

## 실행



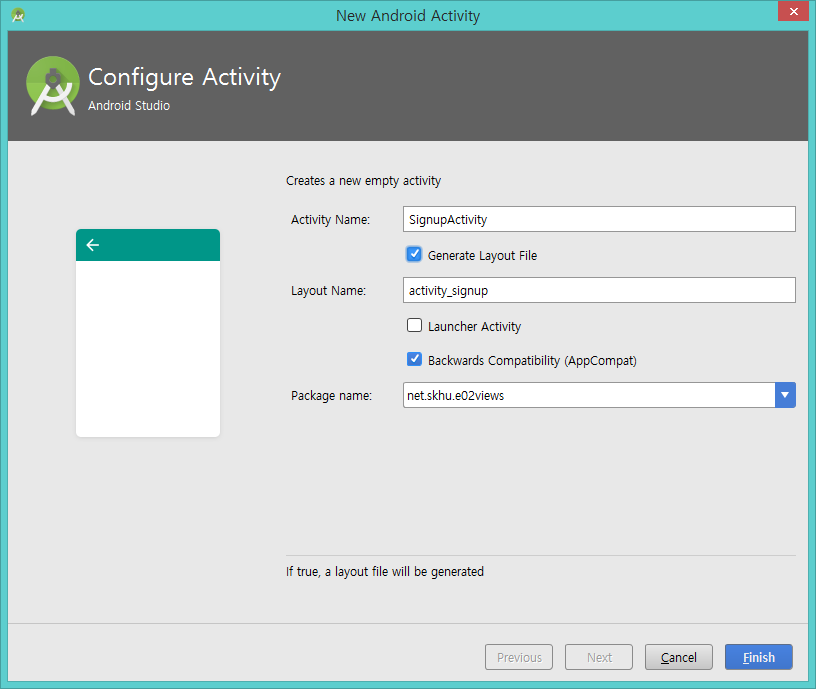
# 회원 가입 화면 만들기

## 새 Activity 생성



Project 창에서 app 폴더를 마우스 오른쪽 버튼으로 클릭하고

메뉴에서 New - Activity - Empty Activity 클릭



위와 같이 입력하고 Finish 버튼 클릭

## 생성된 파일

SingupActivity.java 파일과 activity\_signup.xml 파일이 생성된다.

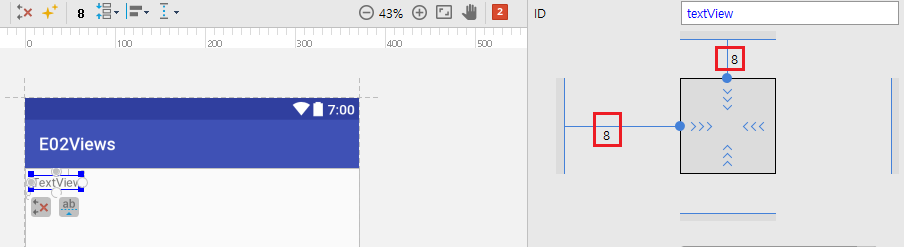
SingupActivity.java : SignupActivity 클래스 소스 파일

activity\_signuup.xml : SignupActivity 레이아웃 리소스 XML 파일

# 회원 가입 화면 레이아웃

activity\_signup.xml 파일 수정

## 로그인 아이디 TextView 추가



Palette 창에서 TextView 항목을 액티비티 화면에 드래그 드롭하고,

위쪽과 왼쪽 constraint 연결선만 화면 끝에 연결한다.

이렇게 연결하면, TextView 객체가 화면 왼쪽 위에 위치하게 된다.

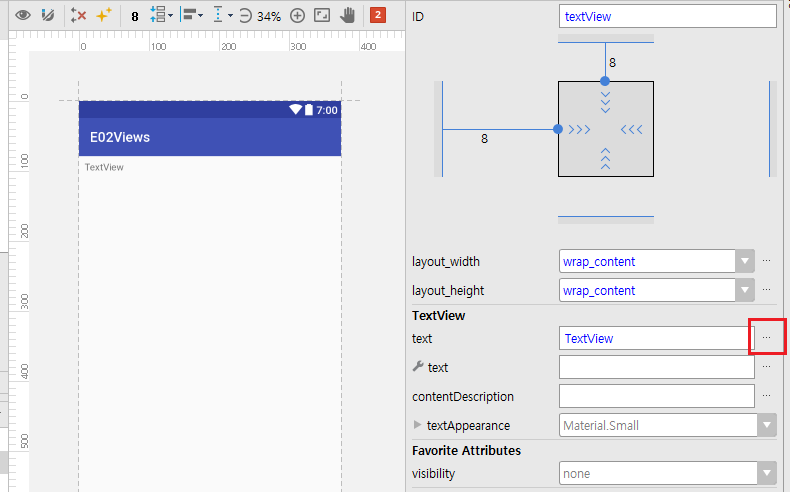
그렇지만 화면 끝에 딱 붙지는 않고, 기본적으로 8 크기의 여백(margin)을 둔다.

여백을 조정하려면, 위 화면에서 8 로 표시된 부분을 클릭하면 된다.

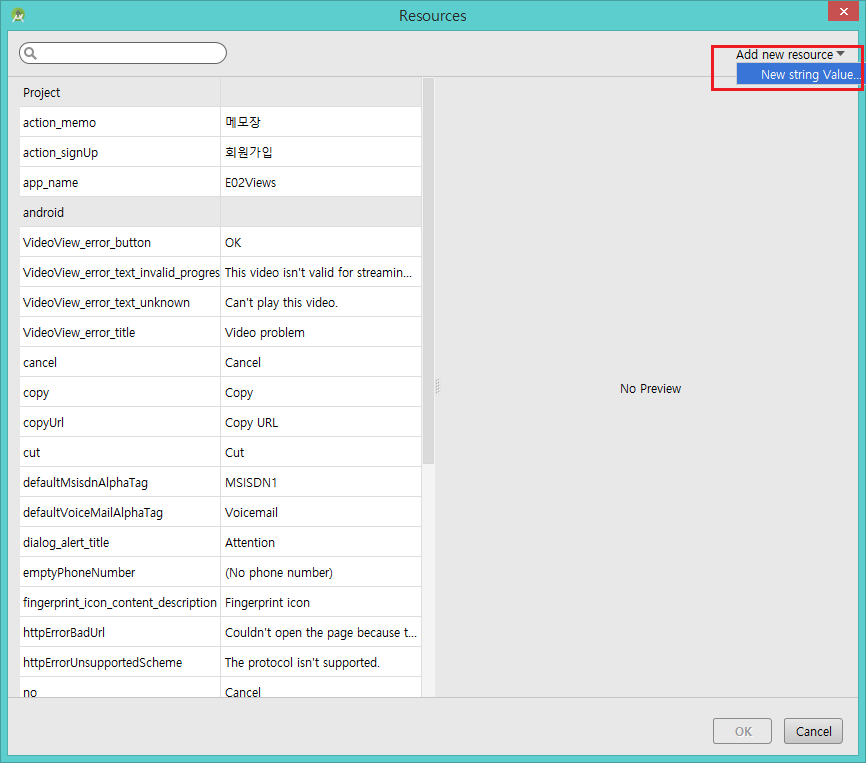
화면에 문자열을 표시하기 위한 TextView 객체가 액티비티에 추가된다.

TextView 객체에 표시되는 문자열을 text 속성에 입력해야 한다.

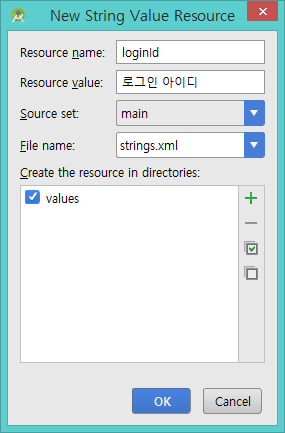
화면에 표시될 문자열을 문자열 리소스로 만들어 주는 것이 바람직하다.

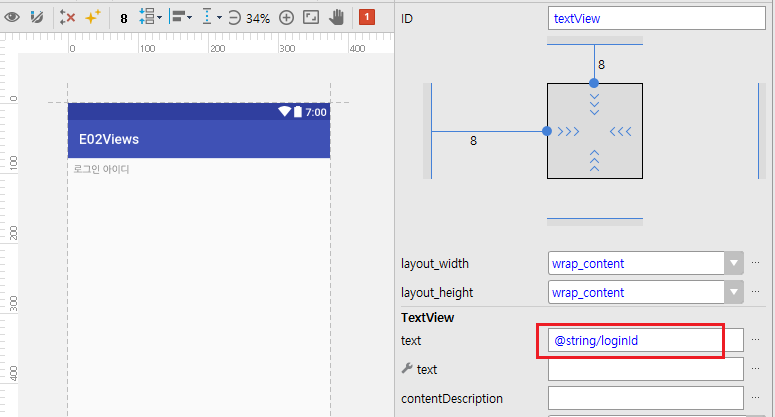


text 속성 오른쪽 ...을 클릭하여 문자열 리소스를 생성하자.



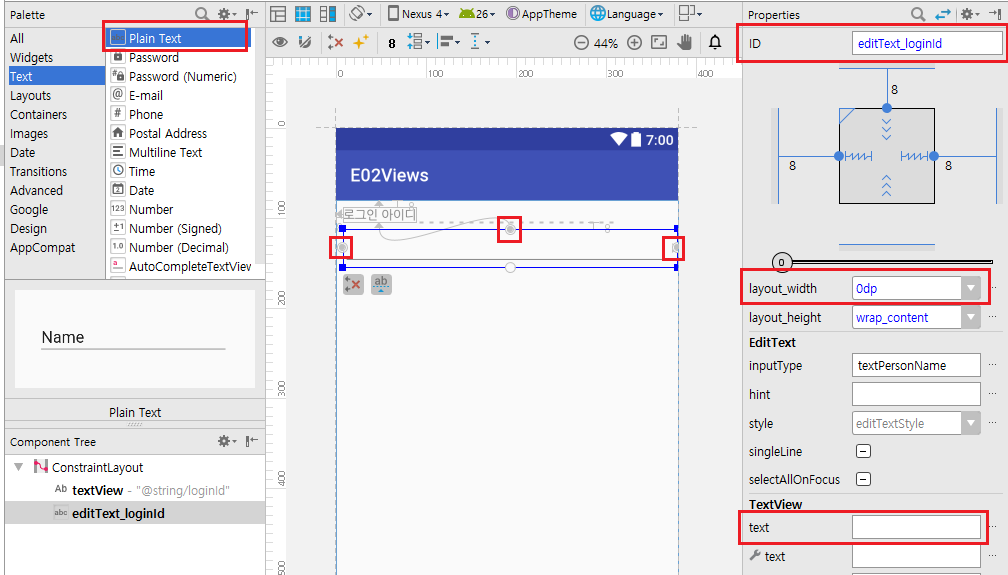
Add New Resource - New string value 버튼을 클릭





방금 생성한 문자열 리소스의 ID는 @string/loginId 이다.

## activity\_signup.xml - 로그인 아이디 입력 EditText 추가



Palette 창에서 Plain Text 항목을 액티비티 화면에 드래그 드롭하자.

Plain Text 항목은 EditText 클래스 객체이다.

위쪽 constraint 연결선을 TextView 아래에 연결하고

왼쪽 constraint 연결선을 화면 왼쪽 끝에 연결하자.

오른쪽 constraint 연결선을 화면 오른쪽 끝에 연결하자.

ID 속성에 editText\_loginId 을 입력하자.

화면 왼쪽의 Component Tree 창을 보면, TextView 객체 하나와 EditText 객체 하나가, ConstraintLayout 객체 내부에 추가되었음을 확인할 수 있다.

text 속성 값을 지우자.

### layout\_width 속성

layout\_width 속성은 가로 폭(width)을 지정하는 속성이다.

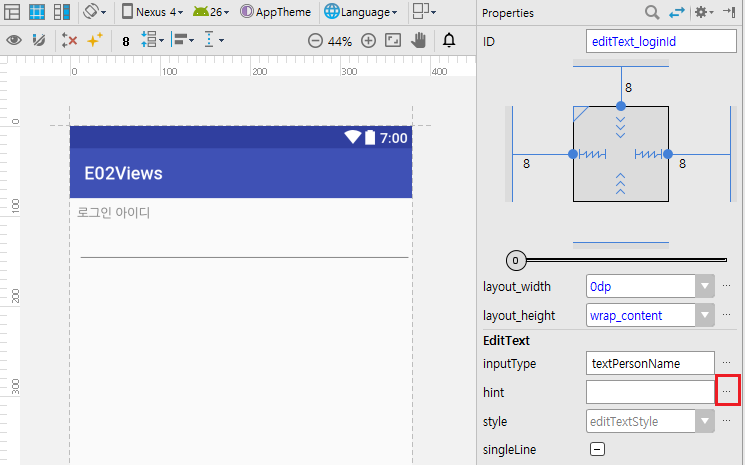
layout\_width 속성 값을 0dp로 지정하면, 가로 방향 폭이 화면 크기가 된다.

layout\_width 속성 값을 0dp로 지정하는 것은, match\_constraint 값을 지정하는 것과 효과가 같다.

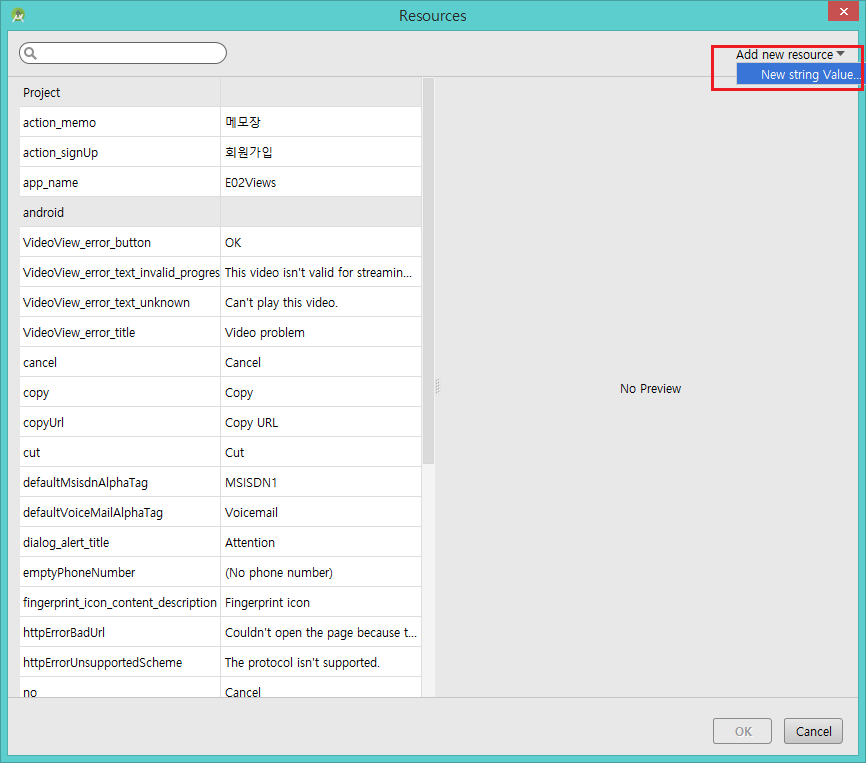
### hint 속성

무엇을 입력할지 안내하는 문자열이 EditText의 배경에 표시될 수 있도록 해주는 속성이다.

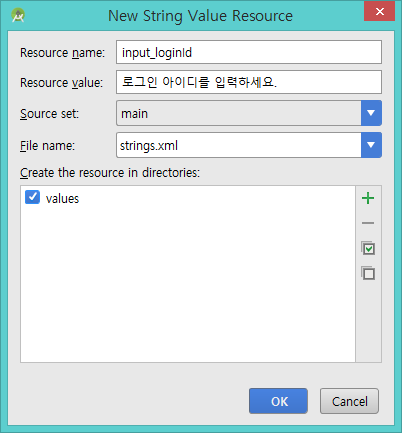
안내 문자열을 문자열 리소스로 만들어 주는 것이 바람직하다.

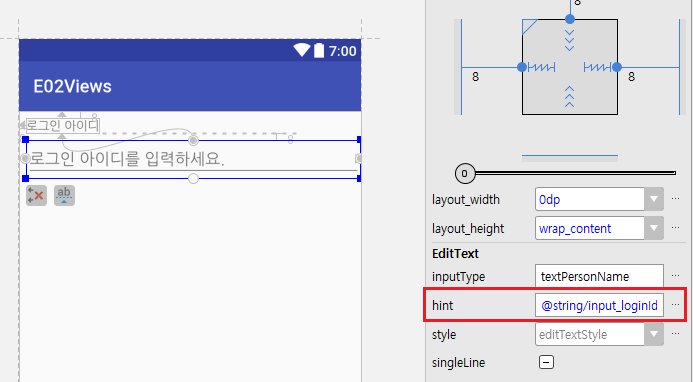


hint 속성 옆의 ... 버튼을 클릭하자.



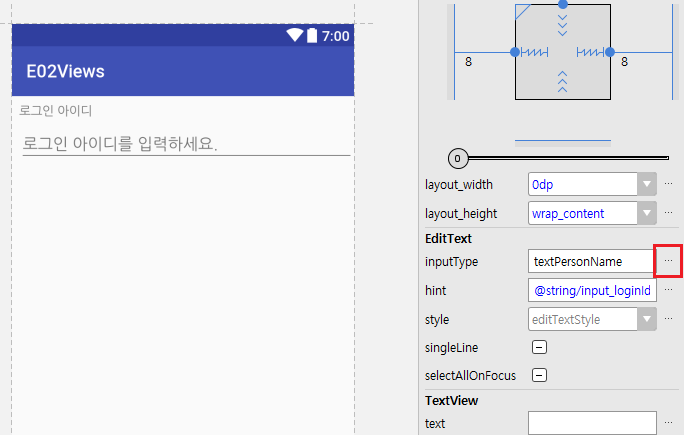
Add New Resource - New string value 버튼을 클릭





hint 속성을 지정하면, 화면의 EditText에 안내 문구가 표시된다.

### inputType 속성



inputType 속성은, 화면에 표시할 적당한 입력 자판(keyboard)을 지정하는 용도로 사용된다.

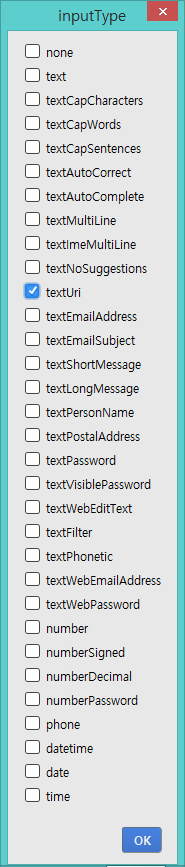
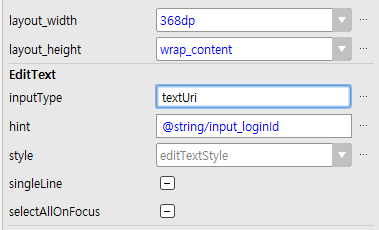
숫자를 입력할 때는 숫자 입력 자판이, 영문을 입력할 때는 알파벳 자판이 표시되는 것이 바람직하다.

사용자가 편리하게 입력할 있도록 inputType 속성 값을 지정해 주자.

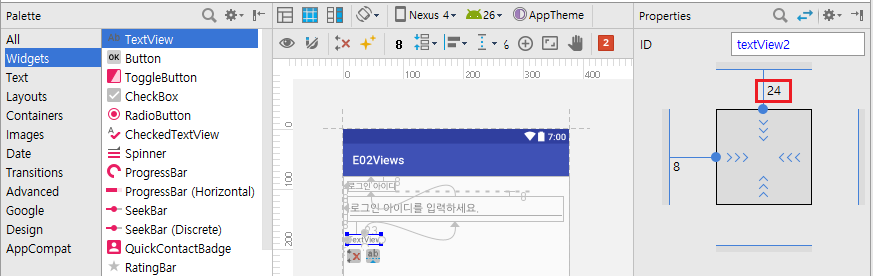
inputType 속성이 text 이면, 한글 자판이 표시된다.

영문 자판이 표시되게 하려면, inputType 속성을 textUri 로 설정하자.

로그인 아이디에 영문 알파벳이 입력되야 하므로 textUri 를 선택하자.

## 비밀번호 TextView 추가



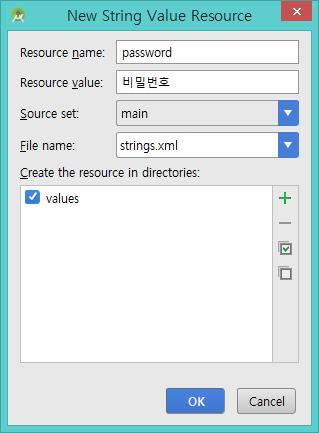
Palette 창에서 TextView 항목을 액티비티 화면에 드래그 드롭하자.

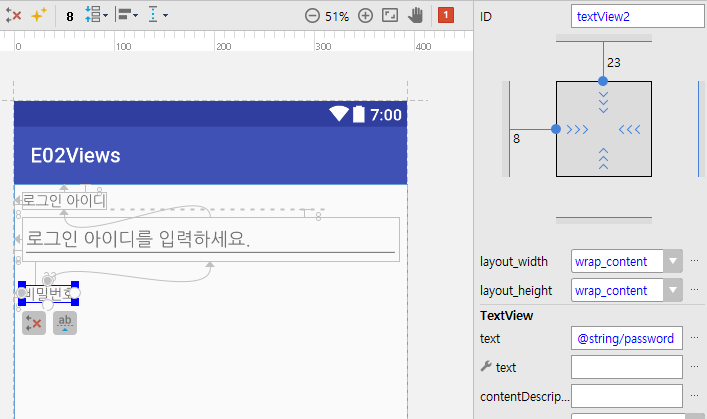
위쪽 constraint 연결선을 EditText 아래에 붙이자.

왼쪽 constraint 연결선을 화면 왼쪽 끝에 붙이자.

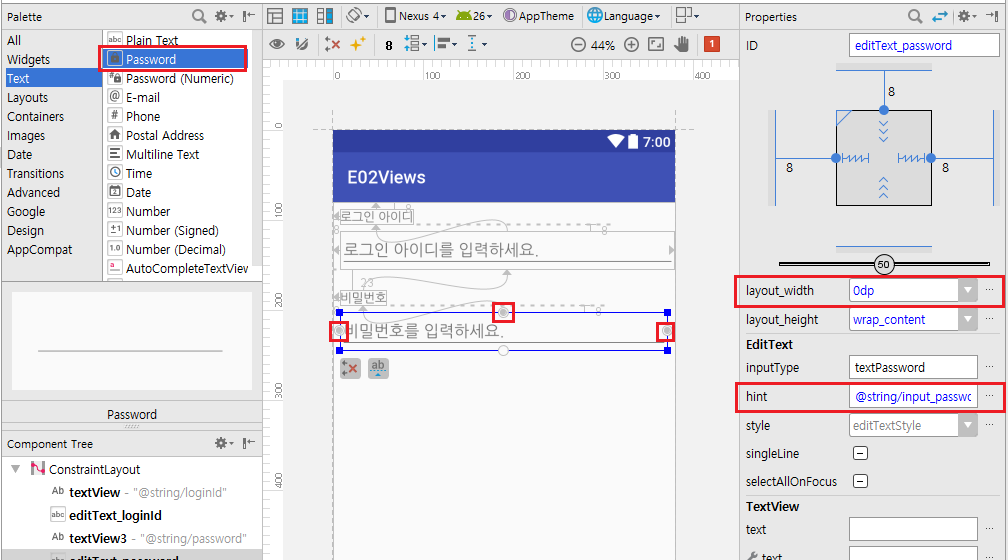
위쪽 여백(margin) 값을 24로 수정하자.

text 속성 옆의 ... 버튼을 클릭하여 문자열 리소스를 만들자.





## 비밀번호를 입력할 EditText 추가



Palette 창에서 Password 항목을 액티비티 화면에 드래그 드롭하자.

Password 항목은 EditText 클래스 객체이다.

위쪽 constraint 연결선을 TextView 아래에 연결하고

왼쪽 constraint 연결선을 화면 왼쪽 끝에 연결하자.

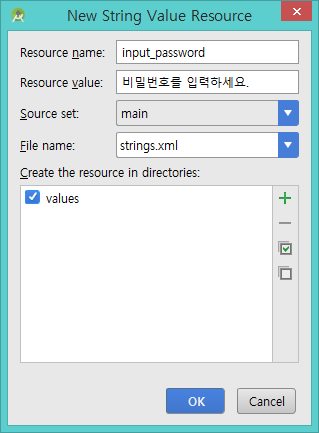
오른쪽 constraint 연결선을 화면 오른쪽 끝에 연결하자.

ID 속성에 editText\_password 을 입력하자.

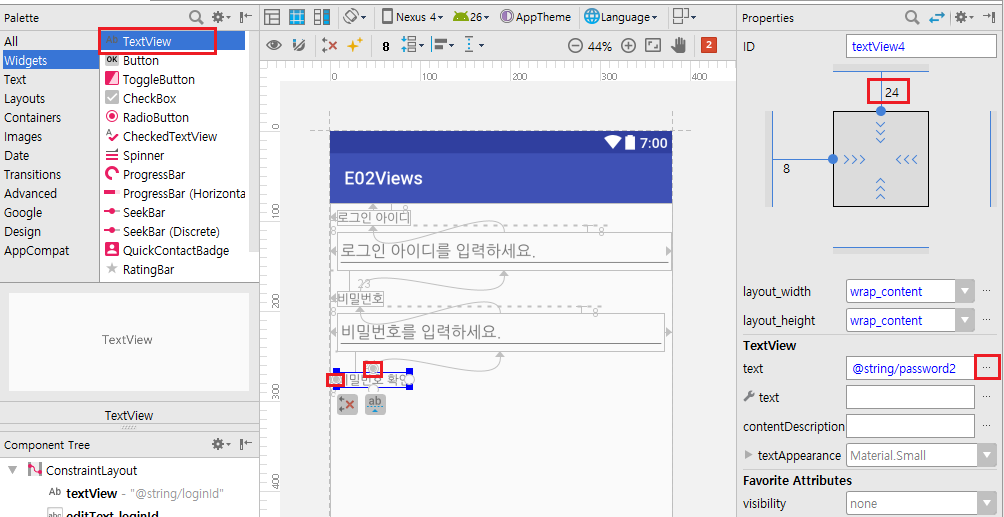
layout\_width 속성은 가로 폭(width)을 지정하는 속성이다.

layout\_width 속성 값을 0dp로 지정하면, 가로 방향 폭이 화면 크기가 된다.

hint 속성 값으로 아래와 같은 문자열 리소스를 설정하자.



## 비밀번호 확인 TextView 추가



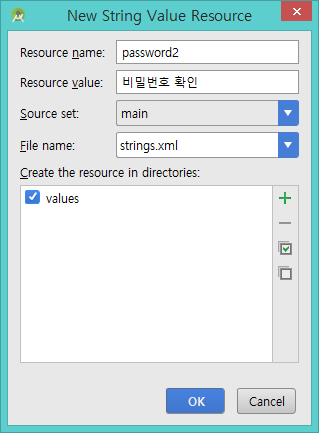
Palette 창에서 TextView 항목을 액티비티 화면에 드래그 드롭하자.

위쪽 constraint 연결선을 EditText 아래에 붙이자.

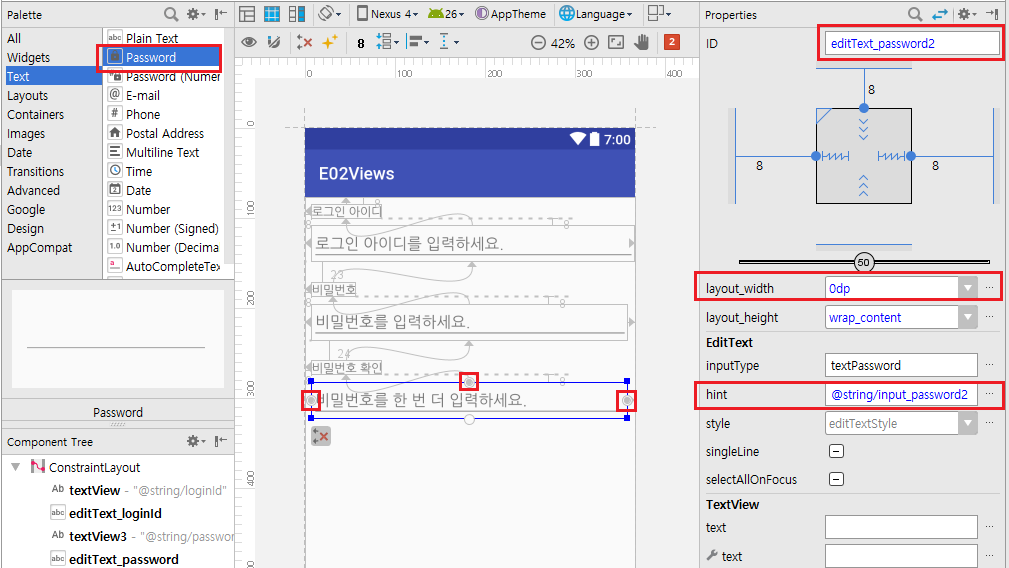
왼쪽 constraint 연결선을 화면 왼쪽 끝에 붙이자.

위쪽 여백(margin) 값을 24로 수정하자.

text 속성 옆의 ... 버튼을 클릭하여 문자열 리소스를 만들자.



## 비밀번호 확인 EditText 추가



Palette 창에서 Password 항목을 액티비티 화면에 드래그 드롭하자.

Password 항목은 EditText 클래스 객체이다.

위쪽 constraint 연결선을 TextView 아래에 연결하고

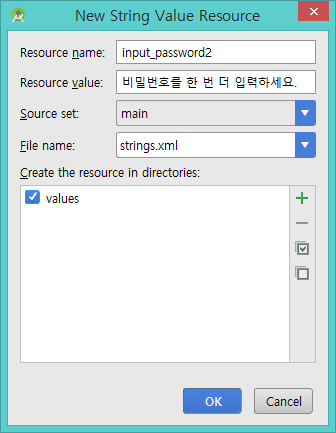
왼쪽 constraint 연결선을 화면 왼쪽 끝에 연결하자.

오른쪽 constraint 연결선을 화면 오른쪽 끝에 연결하자.

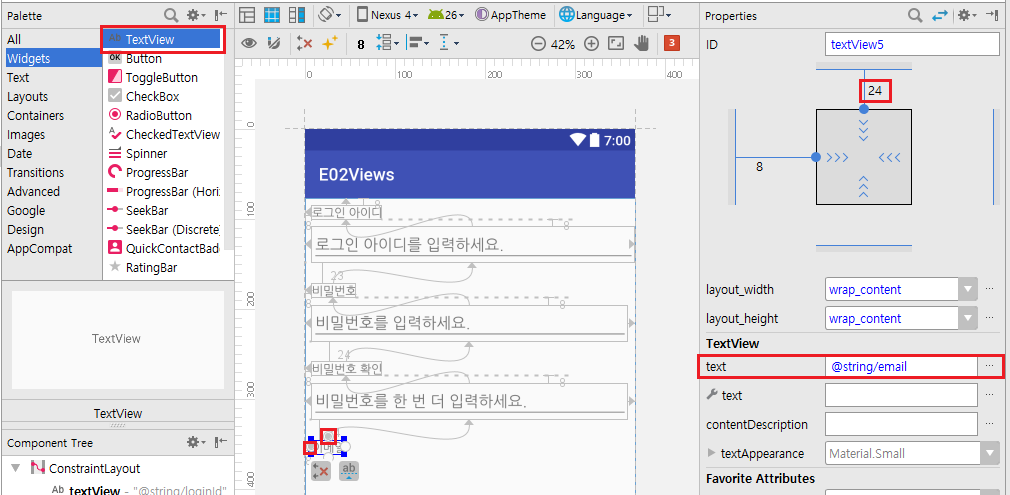
ID 속성에 editText\_password2 를 입력하자.

layout\_width 속성 값을 0dp로 지정하자.

hint 속성 값으로 아래와 같은 문자열 리소스를 설정하자.



## 이메일 TextView 추가



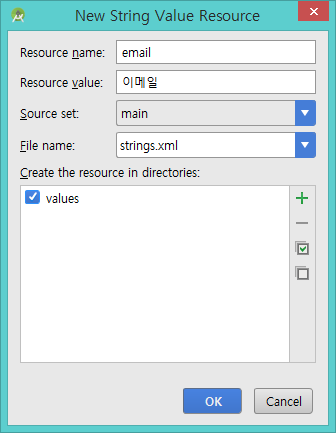
Palette 창에서 Plain TextView 항목을 액티비티 화면에 드래그 드롭하자.

위쪽 constraint 연결선을 EditText 아래에 붙이자.

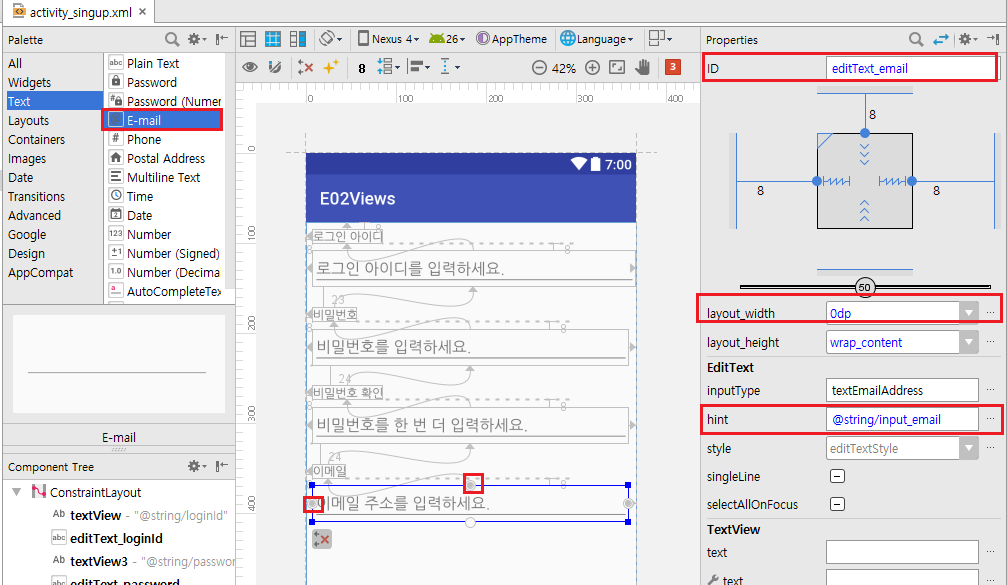
왼쪽 constraint 연결선을 화면 왼쪽 끝에 붙이자.

위쪽 여백(margin) 값을 24로 수정하자.

text 속성 옆의 ... 버튼을 클릭하여 문자열 리소스를 만들자.



## 이메일 입력 EditText 추가



Palette 창에서 E-mail 항목을 액티비티 화면에 드래그 드롭하자.

E-mail 항목은 EditText 클래스 객체이다.

위쪽 constraint 연결선을 TextView 아래에 연결하고

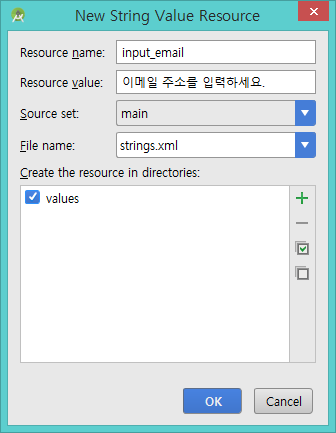
왼쪽 constraint 연결선을 화면 왼쪽 끝에 연결하자.

오른쪽 constraint 연결선을 화면 오른쪽 끝에 연결하자.

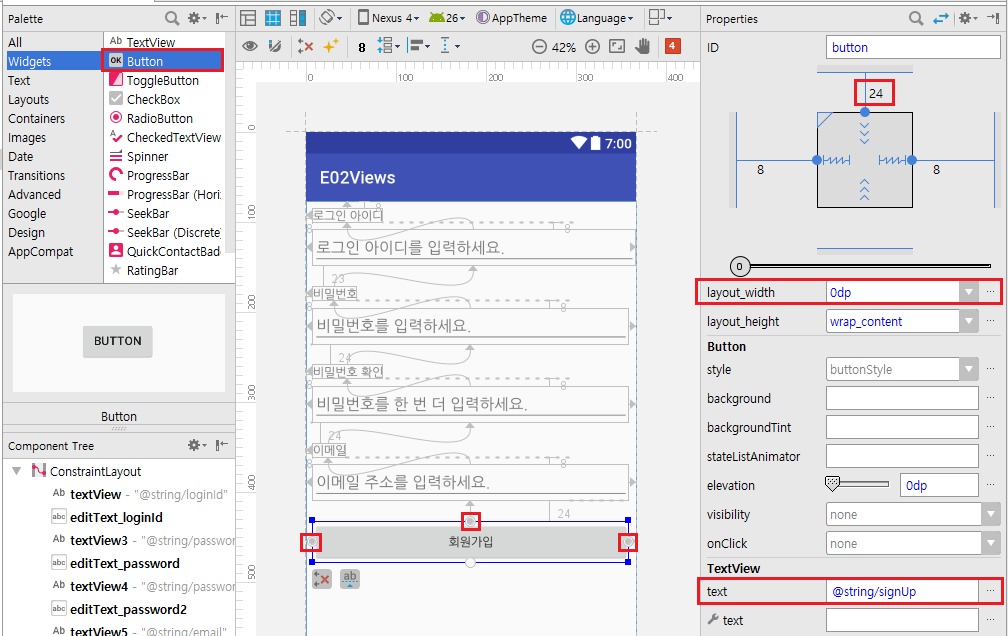
ID 속성에 editText\_email 을 입력하자.

layout\_width 속성 값을 0dp로 지정하자.

hint 속성 값으로 아래와 같은 문자열 리소스를 설정하자.



## 버튼 추가



Palette 창에서 Button 항목을 액티비티 화면에 드래그 드롭하자.

위쪽 constraint 연결선을 EditText 아래에 연결하고

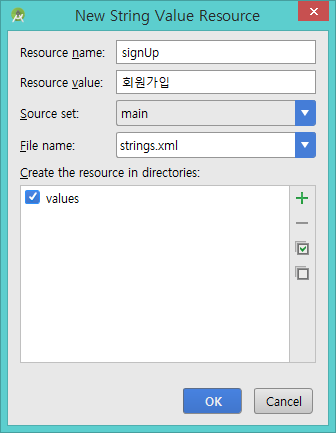
왼쪽 constraint 연결선을 화면 왼쪽 끝에 연결하자.

오른쪽 constraint 연결선을 화면 오른쪽 끝에 연결하자.

위쪽 여백(margin) 값을 24로 수정하자.

layout\_width 속성 값을 0dp로 지정하자.

text 속성 값으로 아래와 같은 문자열 리소스를 설정하자.



# 회원 가입 Java 코드

## SignupActivity.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | **package** net.skhu.e02views;  **import** android.support.v7.app.AppCompatActivity;  **import** android.os.Bundle;  **import** android.util.Log;  **import** android.view.View;  **import** android.widget.Button;  **import** android.widget.EditText;  **import** android.widget.Toast;  **public class** SignupActivity **extends** AppCompatActivity {  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_signup***);  Button button = (Button)findViewById(R.id.***button***);  View.OnClickListener listener = **new** View.OnClickListener() {  @Override  **public void** onClick(View view) {  EditText editText\_loginId = (EditText)findViewById(R.id.***editText\_loginId***);  String loginId = editText\_loginId.getText().toString();  **if** (*isEmptyOrWhiteSpace*(loginId))  editText\_loginId.setError(**"로그인 아이디를 입력하세요"**);  EditText editText\_password = (EditText)findViewById(R.id.***editText\_password***);  String password = editText\_password.getText().toString();  **if** (*isEmptyOrWhiteSpace*(password))  editText\_password.setError(**"비밀번호를 입력하세요"**);  EditText editText\_password2 = (EditText)findViewById(R.id.***editText\_password2***);  String password2 = editText\_password2.getText().toString();  **if** (password.equals(password2) == **false**)  editText\_password2.setError(**"비밀번호가 일치하지 않습니다"**);  EditText editText\_email = (EditText)findViewById(R.id.***editText\_email***);  String email = editText\_email.getText().toString();  *// 회원 가입 데이터를 서버에 전송하는 코드를 구현해야 함.*  String msg = **"회원가입 성공: "** + loginId + **" "** + email;  Toast.*makeText*(SignupActivity.**this**, msg, Toast.***LENGTH\_LONG***).show();  }  };  button.setOnClickListener(listener);  }  **static boolean** isEmptyOrWhiteSpace(String s) {  **if** (s == **null**) **return true**;  **return** s.trim().length() == 0;  }  } |

|  |
| --- |
| EditText editText\_loginId = (EditText)findViewById(R.id.***editText\_loginId***); |

현재 액티비티의 내부에서 id가 editText\_loginId 인 EditText 객체에 대한 참조를 구한다.

|  |
| --- |
| String loginId = editText\_loginId.getText().toString(); |

EditText 클래스의 getText 메소드가 리턴하는 것은 String 객체가 아니고 Editable 객체이다.

Editable 클래스의 equals 메소드는 identity를 비교하기 때문에 바람직하지 않다.

Editable 클래스의 toString 메소드를 호출하여 String 객체로 변환하는 것이 좋다.

|  |
| --- |
| **static boolean** isEmptyOrWhiteSpace(String s) {  **if** (s == **null**) **return true**;  **return** s.trim().length() == 0;  } |

String 문자열이 null 이거나, 빈 문자열이거나, 공백 문자 뿐이라면 true를 리턴한다.

|  |
| --- |
| **if** (*isEmptyOrWhiteSpace*(loginId))  editText\_loginId.setError(**"로그인 아이디를 입력하세요"**); |

로그인 아이디가 입력되지 않았다면, 화면에 에러 메시지를 표시한다.

EditText 클래스의 setError 메소드를 사용하여, 입력 오류 에러 메시지를 화면에 표시한다.

|  |
| --- |
| Toast.*makeText*(SignupActivity.**this**, msg, Toast.***LENGTH\_LONG***).show(); |

Toast.makeText(...) 메소드의 첫째 파라미터는 Context 객체이어야 한다.

액티비티 클래스의 부모 클래스가 Context 클래스이다.

즉 대부분의 경우 Toast.makeText(...) 메소드의 첫째 파라미터는 액티비티 객체이어야 한다.

줄21~44의 onClick 메소드는 SignupActivity 클래스의 메소드가 아니고,

SignupActivity 클래스 내부에서 구현된 anonymous inner class의 메소드이다.

이 inner class의 메소드에서 this는 물론 자기 자신인 anonymous inner class 객체이다.

inner class의 메소드에서 바깥쪽 클래스(outer class)의 this를 사용할 수 있다.

아웃터클래스명.this 형태로 사용하면 된다. (예: 줄43의 SignupActivity.this )

줄43에서 Toast.makeText(...) 메소드의 첫째 파라미터로, 그냥 this를 전달하면, 컴파일 에러가 발생한다.

this가 액티비티 객체가 아니기 때문이다. SignupActivity.this를 전달해야 한다.

Inner class의 메소드에서 outer class의 this를 사용할 수 있다.

단, 그냥 this라고 하면 자기 자신인 inner class 객체의 this 이다.

Outer클래스이름.this 라고 해야 outer class 객체의 this 이다.

(줄18)

Button button = (Button)findViewById(R.id.button); // Ok

Button button = (Button)this.findViewById(R.id.button); // Ok

줄18에서 호출되는 findViewById 메소드는 Activity 클래스에서 상속된 메소드이다.

줄18은 SignupActivity 클래스의 메소드이므로 부모 클래스의 메소드를 호출할 수 있다.

줄18에서 findViewById 메소드를 호출할 때 this가 생략되었다.

(줄22)

EditText editText\_loginId = (EditText)findViewById(R.id.editText\_loginId); // Ok

EditText editText\_loginId = (EditText)this.findViewById(R.id.editText\_loginId); // Error

EditText editText\_loginId = (EditText)SignupActivity.this.findViewById(R.id.editText\_loginId); // Ok

줄22는 anonymous inner class 메소드이다. 여기서 this는 anonymous inner class 객체의 this 이다.

inner class 메소드에서 outer class 메소드와 멤버 변수를 사용할 수 있다.

그래서 줄22에서 outer class인 액티비티의 findViewById 메소드를 호출할 수 있다.

그런데, 줄22에서 생략된 것은 this 가 아니다. 줄22에서 this는 anonymous inner class 객체이기 때문이다.

findViewById는 outer class인 액티비티의 메소드이므로, 줄22에서 생략된 this는 SignupActivity.this 이다.

### inner class의 this

(1) onClick 메소드는 annoymous inner class 메소드이다.

(2) onClick 메소드 내부의 this는 annoymous inner class 객체를 가르킬 수도 있고,

outter class 객체를 가르킬 수도 있다. (SignupActivity 클래스)

(3) onClick 메소드 내부의 findViewById 메소드 호출 앞에 this. 이 생략되었다.

(4) 따라서 findViewById 메소드는 annoymous inner class 메소드가 아니고

SignupActivity 클래스의 부모 클래스 메소드이다.

명제 (4)가 false일 수 밖에 없는 이유.

## Editable 인터페이스

Java 표준 라이브러리에서 문자열에 해당하는 클래스는 3개다. String, StringBuilder, StringBuffer.

이 클래스들의 부모는 CharSequance 인터페이스다.

String 클래스의 특징은, 객체가 생성된 후 내부 문자열을 수정할 수 없다는 것이다.

이런 클래스를 immutable 클래스라고 한다.

객체를 생성한 후 내부 문자열을 수정할 수 있는 클래스는 StringBuilder 이다.

StringBuffer 클래스의 사용법은 StringBuilder 클래스와 같다.

StringBuffer 클래스의 특징은 thread safe 하다는 점이다. 여러 스레드들이 StringBuffer 객체 하나를 공유해도 안전하다. thread safe 해야 하기 때문에 StringBuffer는 StringBuilder 보다 느리다.

안드로이드 표준 라이브러리에 문자열 클래스들이 몇 개 더 들어있다.

SpannableString 클래스와 SpannableStringBuilder 클래스가 안드로이드 라이브러리에 들어있다.

이 클래스들도 String 클래스 처럼 CharSequence 인터페이스를 implements 했다.

EditText 클래스의 getText 메소드가 리턴하는 문자열은 SpannableStringBuilder 클래스이다.

SpannableStringBuilder 클래스는 Java의 CharSequance 인터페이스를 implements 했고,

안드로이드의 Editable 인터페이스도 implements 했다.

EditText 클래스의 getText 메소드의 리턴 타입은 Editable 이다.

Editable getText();

## 화면 전환 기능 구현

MainActivty 화면에서 회원가입 메뉴를 클릭하면 SignupActivity 화면으로 넘어가는 기능을 구현하자.

**MainActivity.java 수정**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | **package** net.skhu.e02views;  **import** android.content.Intent;  **import** android.support.v7.app.AppCompatActivity;  **import** android.os.Bundle;  **import** android.view.Menu;  **import** android.view.MenuItem;  **import** android.widget.Toast;  **public class** MainActivity **extends** AppCompatActivity {  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  }  @Override  **public boolean** onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.***menu\_main***, menu);  **return true**;  }  @Override **public boolean** onOptionsItemSelected(MenuItem item) {  **int** id = item.getItemId();  **if** (id == R.id.***action\_signUp***) {  Intent intent = **new** Intent(**this**, SignupActivity.**class**);  startActivity(intent);  **return true**;  } **else if** (id == R.id.***action\_memo***) {  Toast.*makeText*(**this**, **"메모장 메뉴 클릭"**, Toast.***LENGTH\_SHORT***).show();  **return true**;  }  **return super**.onOptionsItemSelected(item);  }  } |

|  |
| --- |
| Intent intent = **new** Intent(**this**, SignupActivity.**class**);  startActivity(intent); |

다른 액티비티를 호출하려면,

먼저 Intent 클래스 객체를 생성해야 한다.

Intent 클래스의 생성자의 첫째 파라미터는 Context 객체이다.

둘째 파라미터는 호출할 액티비티의 **클래스이름**.class 이다.

그다음 startActivity 메소드를 호출해야 한다.

이 메소드의 파라미터는 Intent 객체이다.

### 클래스명.class, 객체.getType()

Java 코드가 실행될 때, 클래스마다 그 클래스 구조에 대한 정보를 담은

객체가 하나씩 만들어진다. 이 객체를 메타 클래스 객체라고 부른다.

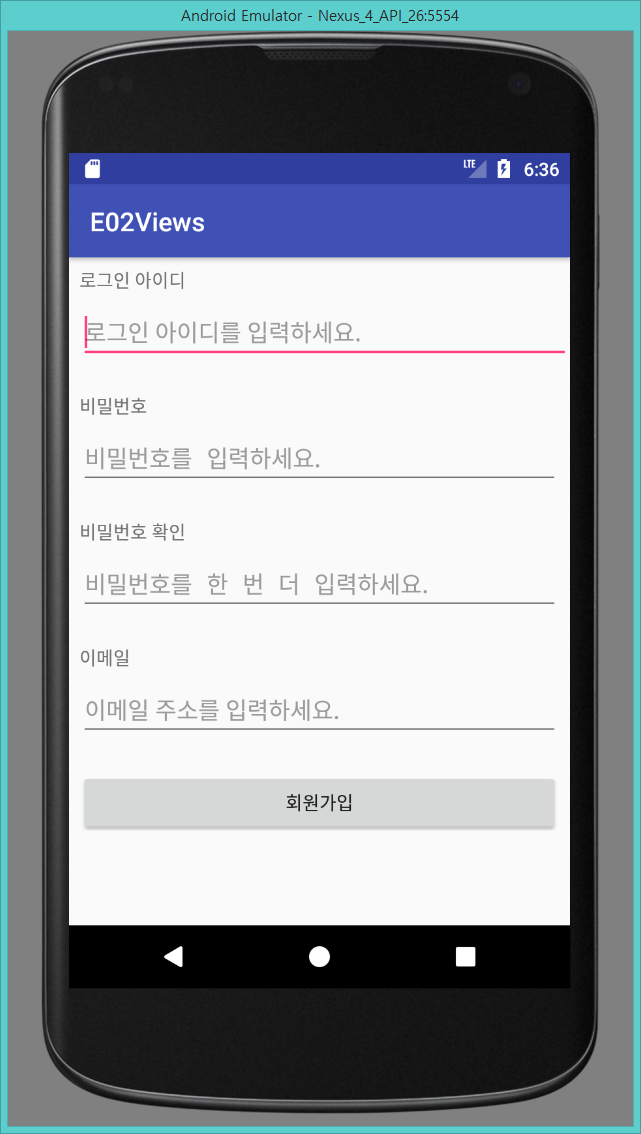
그 객체에 대한 참조를 얻으려면 클래스명.class 코드를 사용하면 된다.

메타 클래스 객체가 속한 클래스 이름은 Class 이다.

클래스 구조: 멤버 변수, 메소드 목록..

## 결과 소스코드

E02Views\_src1.zip

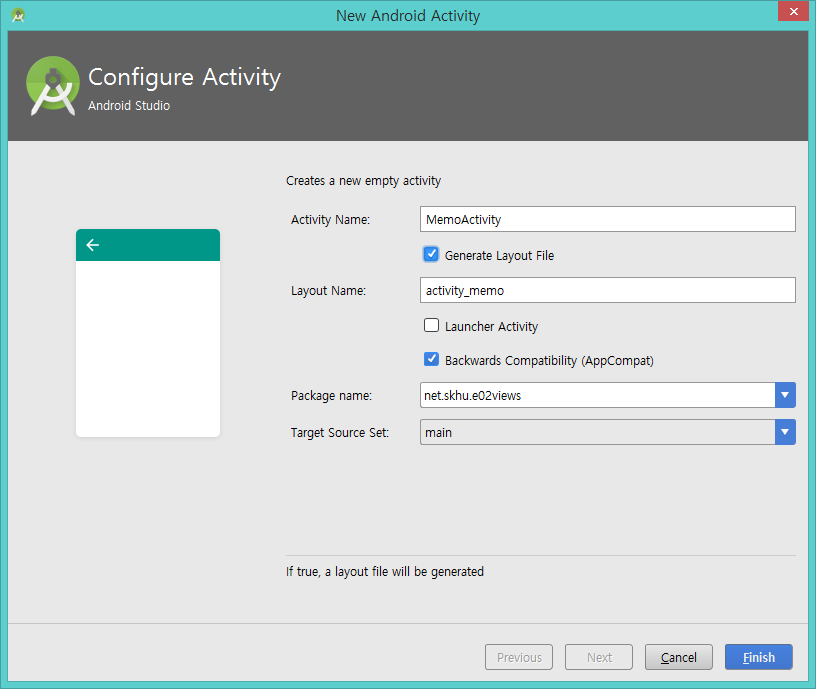


# 메모장 화면 만들기

## 새 Activity 생성

Project 창에서 app 폴더를 마우스 오른쪽 버튼으로 클릭하고

메뉴에서 New - Activity - Empty Activity 클릭



### 파일 생성

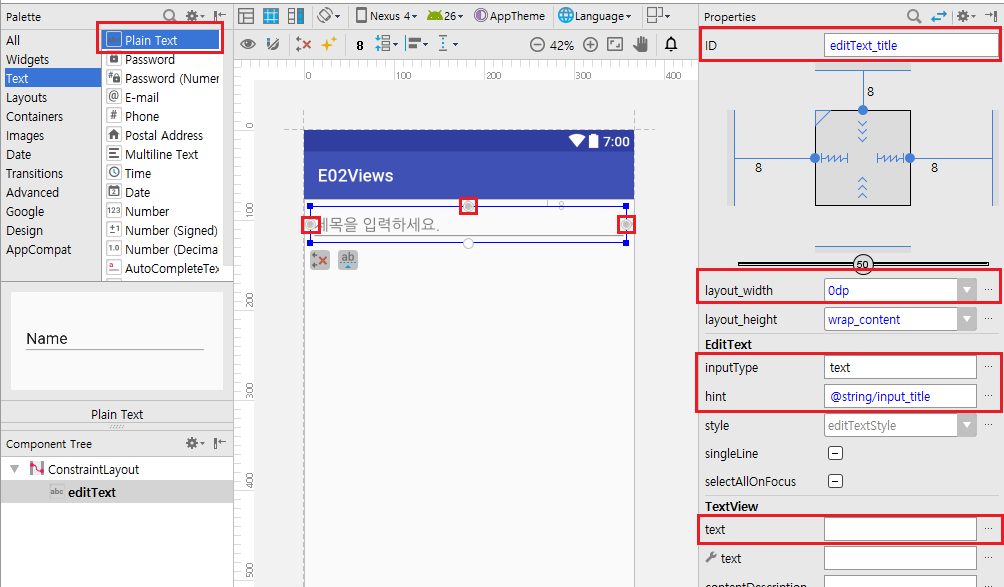
Activity Name: MemoActivity.java

Layout Name: activity\_memo.xml

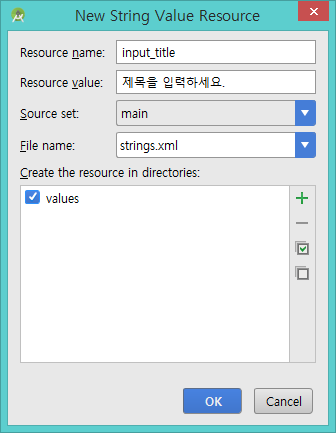
## 메모 화면 레이아웃

activity\_memo.xml 파일 수정

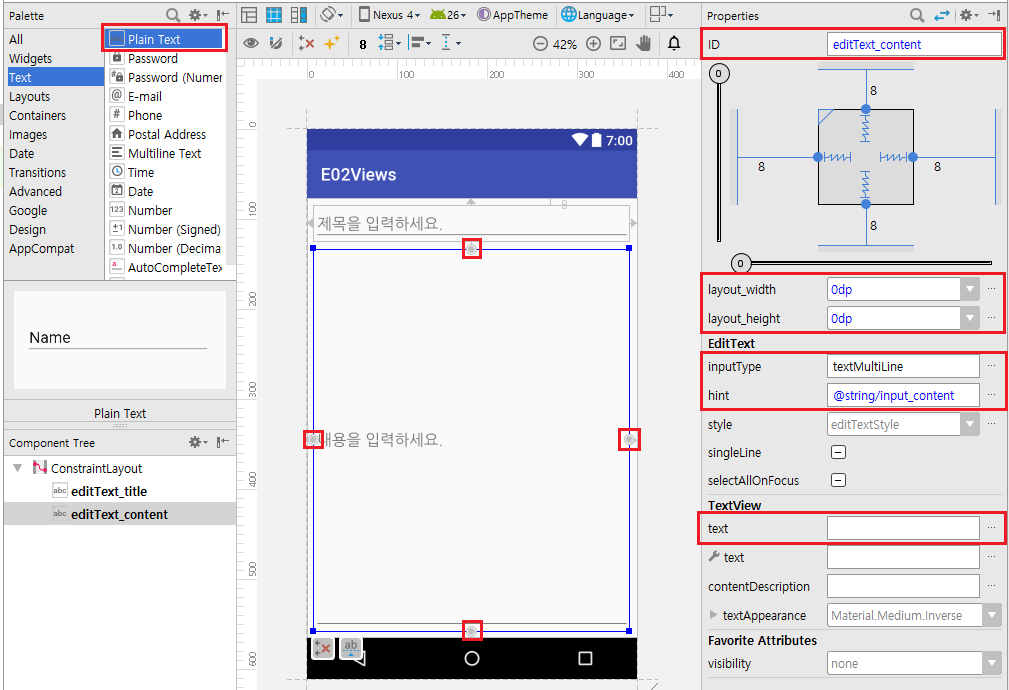
### 제목 EditText 추가하기



화면 왼쪽 Palette 창에서 Plain Text 항목을 액티비티 화면에 드래그 드롭하여 EditText를 추가한다.

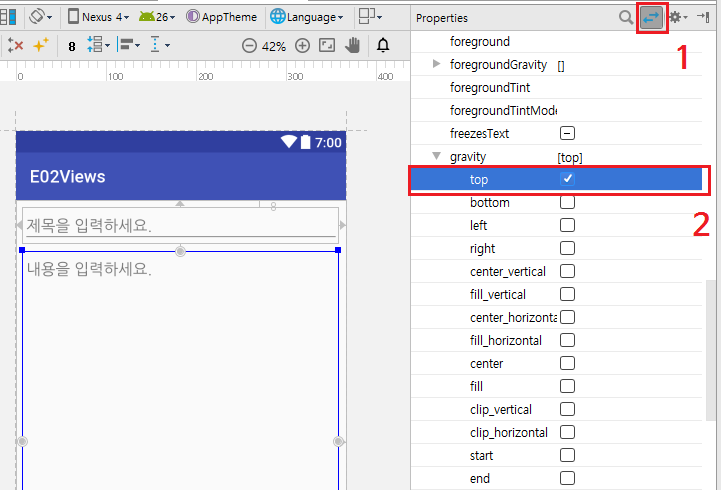


### 내용 EditText 추가하기



내용 문자열이 수직 방향으로 가운데 위치하는 것이 기본이다.

상단에 위치하도록 수정하자.

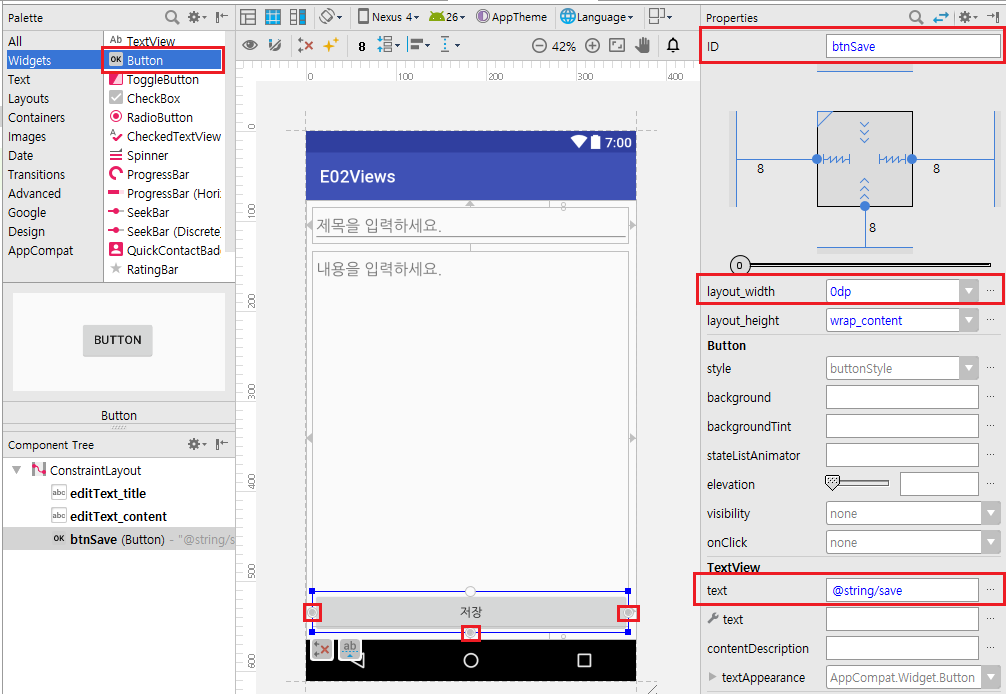


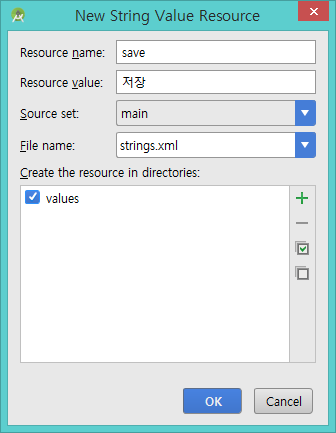
1번 버튼을 클릭하면, 전체 속성(properties)을 볼 수 있다.

gravity 속성 값을 top 으로 수정하면,

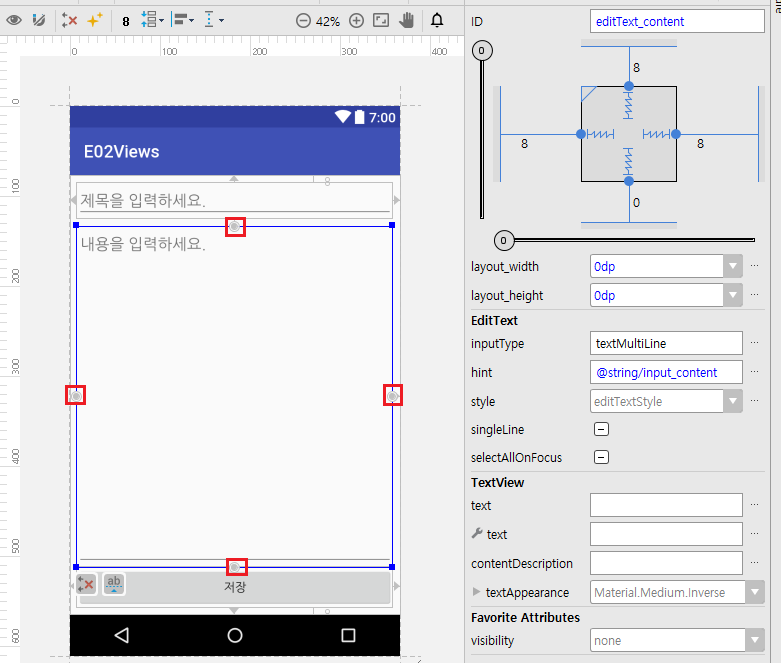
문자열이 EditText 상단에 위치하게 된다.

### 저장 버튼 추가하기





### 내용 EditText의 constraint 수정하기



내용 EditText의 아래쪽 constrinat 연결선을 버튼의 위쪽에 연결하자.

# 메모장 Java 코드

## MemoActivity.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | **package** net.skhu.e02views;  **import** android.support.v7.app.AppCompatActivity;  **import** android.os.Bundle;  **import** android.view.View;  **import** android.widget.Button;  **import** android.widget.EditText;  **import** android.widget.Toast;  **public class** MemoActivity **extends** AppCompatActivity {  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_memo***);  Button button = (Button) findViewById(R.id.***btnSave***);  View.OnClickListener listener = **new** View.OnClickListener() {  @Override  **public void** onClick(View view) {  EditText editText\_title = (EditText) findViewById(R.id.***editText\_title***);  String title = editText\_title.getText().toString();  **if** (*isEmptyOrWhiteSpace*(title))  editText\_title.setError(**"제목을 입력하세요"**);  EditText editText\_content = (EditText) findViewById(R.id.***editText\_content***);  String content = editText\_content.getText().toString();  **if** (*isEmptyOrWhiteSpace*(content))  editText\_content.setError(**"내용을 입력하세요"**);  *// 메모 데이터를 서버에 전송하는 코드를 구현해야 함.*  String msg = **"저장 성공: "** + title;  Toast.*makeText*(MemoActivity.**this**, msg, Toast.***LENGTH\_LONG***).show();  }  };  button.setOnClickListener(listener);  }  **static boolean** isEmptyOrWhiteSpace(String s) {  **if** (s == **null**) **return true**;  **return** s.toString().trim().length() == 0;  }  } |

Toast.makeText 메소드의 첫째 파라미터는 액티비티 객체에 대한 참조이어야 한다.

Toast.makeText 메소드의 첫째 파라미터로 MemoActivity 를 가르키는 참조를 전달해야 한다.

줄34은 inner class의 메소드 내부이므로, this 는 inner class 객체를 가르킨다.

바깥쪽 클래스(outer class) 이름을 붙인 MemoActivity.this 는 MemoActivity 객체를 가르킨다.

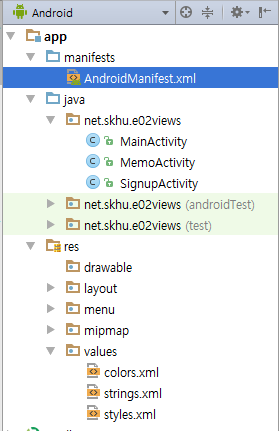
## 화면 전환 기능 구현

MainActivty 화면에서 메모장 메뉴를 클릭하면 MemoActivity 화면으로 넘어가는 기능을 구현하자.

**MainActivity.java 수정**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | **package** net.skhu.e02views;  **import** android.content.Intent;  **import** android.support.v7.app.AppCompatActivity;  **import** android.os.Bundle;  **import** android.view.Menu;  **import** android.view.MenuItem;  **import** android.widget.Toast;  **public class** MainActivity **extends** AppCompatActivity {  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  }  @Override  **public boolean** onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.***menu\_main***, menu);  **return true**;  }  @Override **public boolean** onOptionsItemSelected(MenuItem item) {  **int** id = item.getItemId();  **if** (id == R.id.***action\_signUp***) {  Intent intent = **new** Intent(**this**, SignupActivity.**class**);  startActivity(intent);  **return true**;  } **else if** (id == R.id.***action\_memo***) {  Intent intent = **new** Intent(**this**, MemoActivity.**class**);  startActivity(intent);  **return true**;  }  **return super**.onOptionsItemSelected(item);  }  } |

## 화면 제목 변경하기



AndroidManifest.xml 파일 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | *<?***xml version="1.0" encoding="utf-8"***?>*  <**manifest xmlns:android="http://schemas.android.com/apk/res/android"**  **package="net.skhu.e02views"**>  <**application**  **android:allowBackup="true"**  **android:icon="@mipmap/ic\_launcher"**  **android:label="@string/app\_name"**  **android:roundIcon="@mipmap/ic\_launcher\_round"**  **android:supportsRtl="true"**  **android:theme="@style/AppTheme"**>  <**activity android:name=".MainActivity"**>  <**intent-filter**>  <**action android:name="android.intent.action.MAIN"** />  <**category android:name="android.intent.category.LAUNCHER"** />  </**intent-filter**>  </**activity**>  <**activity android:name=".SignupActivity" android:label="@string/action\_signUp"** />  <**activity android:name=".MemoActivity" android:label="@string/action\_memo"** />  </**application**>  </**manifest**> |

@string/action\_signUp 과 @string/action\_memo 은 문자열 리소스의 ID 이다.

문자열 리소스들은 res/values/strings.xml 파일에 들어있다.

이 파일에서 @string/action\_signUp 과 @string/action\_memo 항목을 확인하자.

## 결과 소스코드

E02Views\_src2.zip

