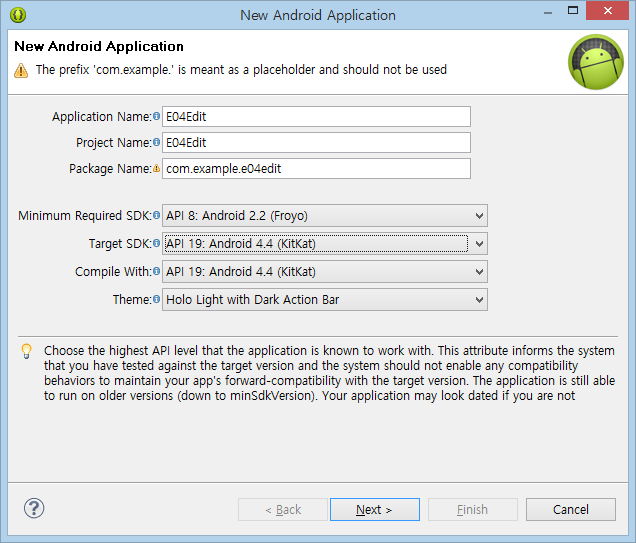
# 예제 E04Edit 프로젝트

액티비티 화면을 두개 만들어 보자

## 프로젝트 생성

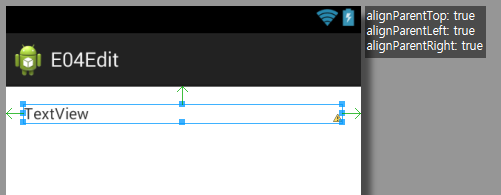
이클립스 메뉴: File – New – Android Application Project



|  |  |
| --- | --- |
| Project Name: | E04Edit |
| Minimum Required SDK: | API 8 |
| Target SDK: | API 19 |
| Compile Width: | API 19 |
| Theme: | Holo Light with Dark Action Bar |

# 레이아웃 리소스 만들기

## activity\_main.xml 수정



TextView의 오른쪽 경계를 오른쪽 끝까지 드래그하면 위와 같이 배치된다.

alignParentTop: true 부모의 위쪽 경계에 붙어서 배치된다

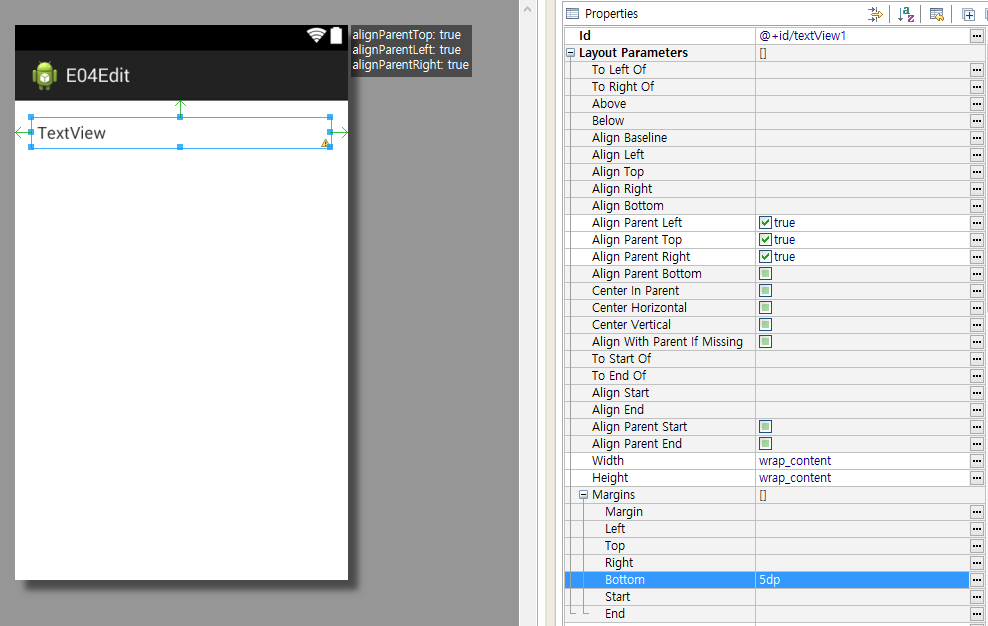
alignParentLeft: true 부모의 왼쪽 경계에 붙어서 배치된다

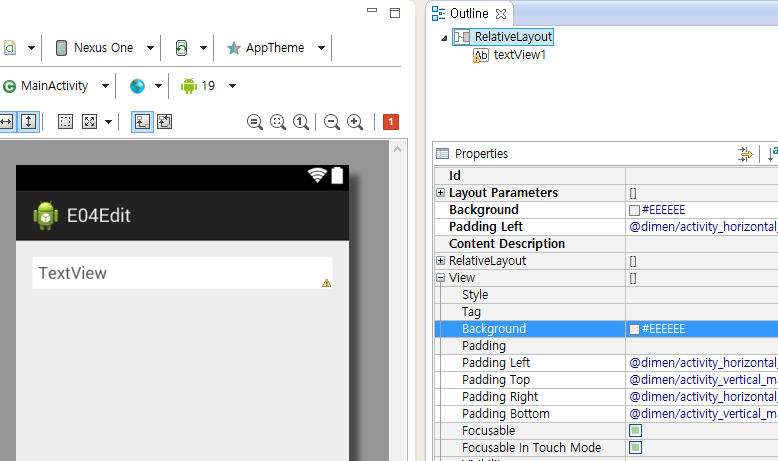
alignParentRight: true 부모의 오른쪽 경계에 붙어서 배치된다

왼쪽과 오른쪽 양쪽으로 붙어서 배치되었으므로 화면 크기에 따라 이 TextView의 폭은 늘어나게 된다.

화면 오른쪽의 Properties 창에서 textView1의 속성을 다음과 같이 입력

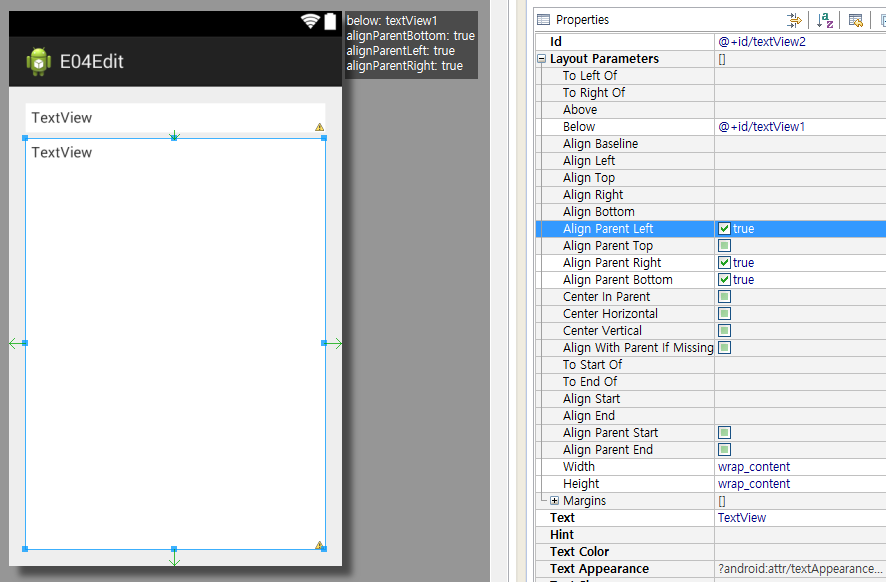
|  |  |
| --- | --- |
| Layout Parameters - Align Parent Left | true |
| Layout Parameters - Align Parent Top | true |
| Layout Parameters - Align Parent Right | true |
| Layout Parameters - Margins - Bottom | 5dp |
| View - Background | #FFFFFF |
| View - Padding | 5dp |





화면 오른쪽 위의 Outline 창에서 RelativeLayout을 선택하고, 그 아래 Properties 창에서

Bacground 속성값으로 #EEEEEE 입력



왼쪽의 Palette에서 TextView를 드래그 드롭하여 하나 더 추가하자.

그리고 새로 추가된 TextView의 위 아래 왼쪽 오른쪽 경계를 끝까지 드래그 하여 위와 같이 배치되도록 하자.

below: textView1 textView1의 아래에 붙어서 배치된다

alignParentBottom: true 부모의 아래쪽 경계에 붙어서 배치된다

alignParentLeft: true 부모의 왼쪽 경계에 붙어서 배치된다

alignParentRight: true 부모의 오른쪽 경계에 붙어서 배치된다

위와 아래로 붙어있고, 왼쪽 오른쪽도 붙어있으므로 화면의 크기에 따라 상하좌우로 늘어나게 된다.

화면 오른쪽의 Properties 창에서 textView1의 속성을 다음과 같이 입력

|  |  |
| --- | --- |
| Layout Parameters - Align Parent Left | true |
| Layout Parameters - Align Parent Right | true |
| Layout Parameters - Align Parent Bottom | true |
| View - Background | #FFFFFF |
| View - Padding | 5dp |

Ctrl+S 단축키를 눌러 저장하고 편집 창을 닫자.

# Java 클래스 구현

## MyMemo.java 구현

화면에 표시할 데이터에 해당하는 MyMemo 클래스를 구현한다.

화면 오른쪽의 Package Explorer 창에서 src 폴더를 우클릭하고

메뉴: New - Class

New Java Class 대화상자에서

Package: com.example.e03list

Class Name: MyMemo

### src/com.example.e04edit/MyMemo.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 | package com.example.e04edit;  public class MyMemo {  private int \_id;  private String title;  private String body;  private String modifiedTime;  public int getId() {  return \_id;  }  public void setId(int id) {  this.\_id = id;  }  public String getTitle() {  return title;  }  public void setTitle(String title) {  this.title = title;  }  public String getBody() {  return body;  }  public void setBody(String body) {  this.body = body;  }  public String getModifiedTime() {  return modifiedTime;  }  public void setModifiedTime(String modifiedTime) {  this.modifiedTime = modifiedTime;  }  } |

MyMemo 클래스의 멤버 변수 값들을 외부의 다른 클래스에서 직접 읽고 쓰는 것은 객체지향적 관점에서 바람직하지 않다. 그래서 멤버 변수 값을 외부에서 읽을 때 사용할 get 메소드, 멤버 변수 값을 외부에서 바꿀 때 사용할 set 메소드를 구현하였다.

예를들어 MyMemo 객체 내부의 title 멤버 변수값을 외부에서 가져가려면 getTitle 메소드를 호출한다.

title 멤버 변수값을 외부에서 대입하려면 setTitle 메소드를 호출한다.

## MyApplication.java 구현

여러 액티비티들 사이에 공유할 데이터를 저장하기 위해 MyApplication 클래스를 구현한다.

화면 오른쪽의 Package Explorer 창에서 src 폴더를 우클릭하고

메뉴: New - Class

New Java Class 대화상자에서

Package: com.example.e03list

Class Name: MyApplication

### src/com.example.e04edit/MyApplication.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | package com.example.e04edit;  import android.app.Application;  public class MyApplication extends Application {  private MyMemo myMemo;  public MyApplication() {  myMemo = new MyMemo();  myMemo.setTitle("The title of memo");  myMemo.setBody("The contents of memo ");  }  public MyMemo getMyMemo() {  return myMemo;  }  public void setMyMemo(MyMemo myMemo) {  this.myMemo = myMemo;  }  } |

(줄5) 안드로이드의 Application 클래스를 상속 받아 MyApplication을 구현한다.

안드로이드의 Application 클래스는 앱을 대표하는 클래스이다. 앱이 실행될 때, Application 클래스 객체가 하나 자동으로 생성된다.

(줄7) 액티비티들 사이에 공유해야 할 MyMemo 객체를 멤버 변수로 정의하였다.

(줄9~13) MyApplication 클래스의 생성자 메소드

MyMemo 객체를 하나 생성해고 그 객체에 대한 참조를 멤버 변수에 대입한다.

앱이 실행될 때 Application 클래스나 그 자식 클래스 객체가 하나 자동으로 생성되기 때문에,

액티비티들 사이에 공유할 데이터가 있다면 위와 같이 Application 클래스의 자식 클래스를 만들고

그 멤버 변수로 공유할 데이터를 구현하는 것이 좋다.

앱이 실행될 때 MyApplication 클래스 객체가 하나 자동으로 생성될 수 있으려면

MyApplication 클래스명을 AndroidManifest.xml 파일에 등록해주어야 한다.

### AndroidManifest.xml

|  |  |
| --- | --- |
| 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 | <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.example.e04edit"  android:versionCode="1"  android:versionName="1.0" >  <uses-sdk  android:minSdkVersion="8"  android:targetSdkVersion="19" />  <application  android:allowBackup="true"  android:icon="@drawable/ic\_launcher"  android:label="@string/app\_name"  android:theme="@style/AppTheme"  android:name="com.example.e04edit.MyApplication" >  <activity  android:name=".MainActivity"  android:label="@string/app\_name" >  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  </application>  </manifest> |

(줄16) Application을 상속받아 구현한 MyApplication 클래스명을 등록한다.

XML 파일에 클래스명을 적어줄 때는 거의 언제나 위와 같이 패키지(package)명까지 전부 적어야 한다.

AndroidManifest.xml 파일은 이 앱과 액티비티들에 대한 정보를 등록하는 파일이다.

## MainActivity.java 수정

### src/com.example.e04edit/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  35  36  37  38 | package com.example.e04edit;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.widget.TextView;  public class MainActivity extends ActionBarActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  TextView textView1 = (TextView)findViewById(R.id.textView1);  TextView textView2 = (TextView)findViewById(R.id.textView2);  MyApplication myApplication = (MyApplication)getApplicationContext();  MyMemo memo = myApplication.getMyMemo();  textView1.setText(memo.getTitle());  textView2.setText(memo.getBody());  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄15) id값이 R.id.textView1인 TextView 객체에 대한 참조를 얻어서 지역변수 textView1에 대입한다.

이 TextView 객체는 줄14의 setContentView 메소드에서 생성되었다.

(줄16) id값이 R.id.textView2인 TextView 객체에 대한 참조를 얻어서 지역변수 textView2에 대입한다.

이 TextView 객체는 줄14의 setContentView 메소드에서 생성되었다.

(줄18) 안드로이드 시스템에 의해 자동으로 생성된 MyApplication 객체에 대한 참조를 얻어서 지역변수 myApplication에 대입한다.

(줄19) MyMemo 객체에 대한 참조를 얻어서 지역변수 memo에 대입한다.

이 MyMemo 객체는 MyApplication 객체의 생성자에서 생성되었다.

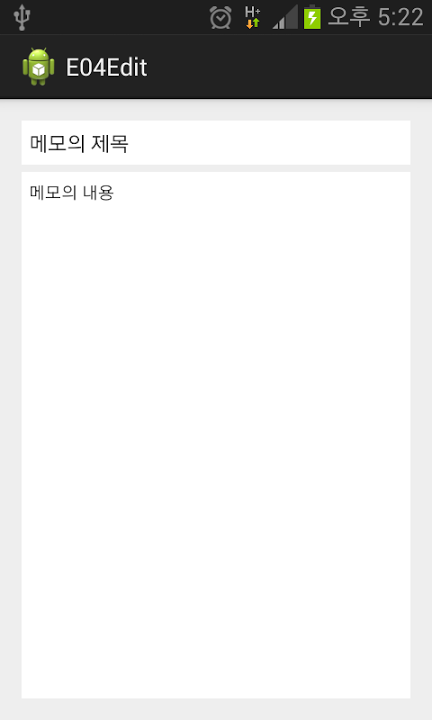
MyApplication 객체는 안드로이드 시스템에 의해 자동 생성되었다.

지역변수 memo에 대입되는 것은 객체 그 자체가 아니고 객체에 대한 참조값이다.

(줄20) MyMemo 객체의 title 멤버 변수 값을 구해서 textView1의 내용으로 등록한다.

(줄21) MyMemo 객체의 body 멤버 변수 값을 구해서 textView2의 내용으로 등록한다.

# 실행



### 소스코드

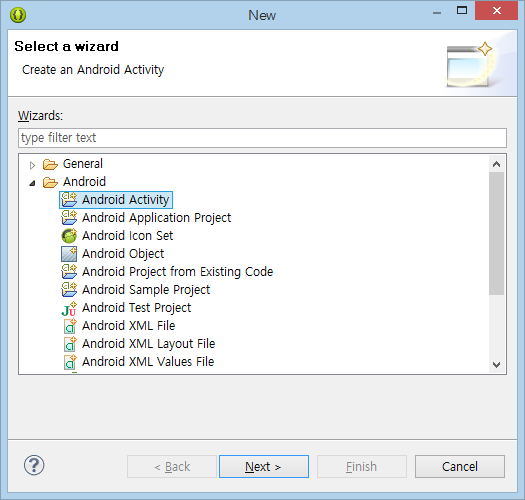
E04Edit\_#1.zip

# EditActivity 액티비티 추가 구현

## 프로젝트에 EditActivity 액티비티 추가

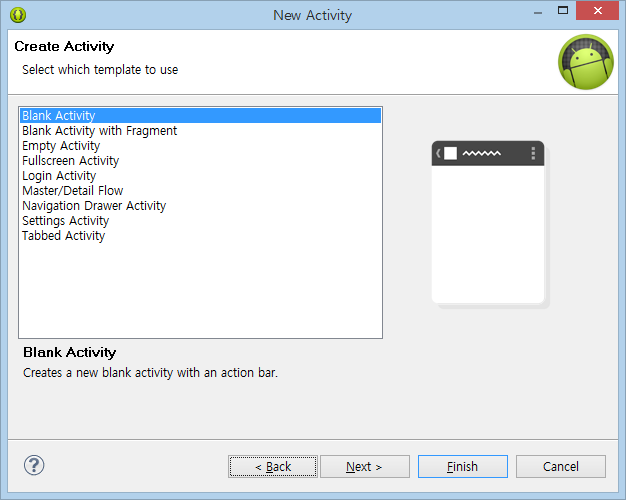
메모를 편집할 수 있는 편집 화면에 해당하는 EditActivity 클래스를 구현하자.

메뉴: File – New – Other



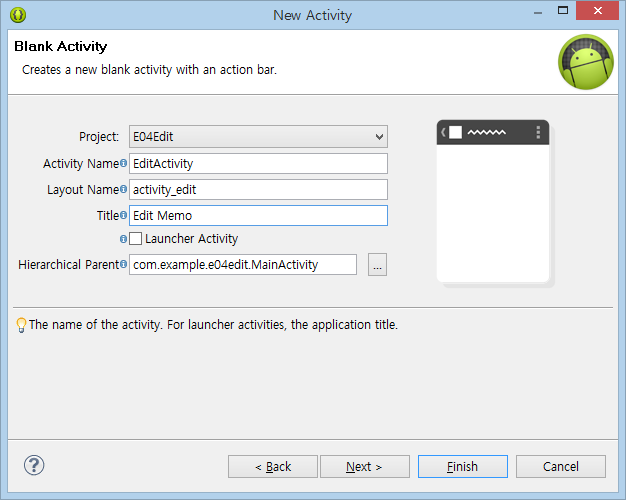
Android Activity 선택

Next 버튼 클릭



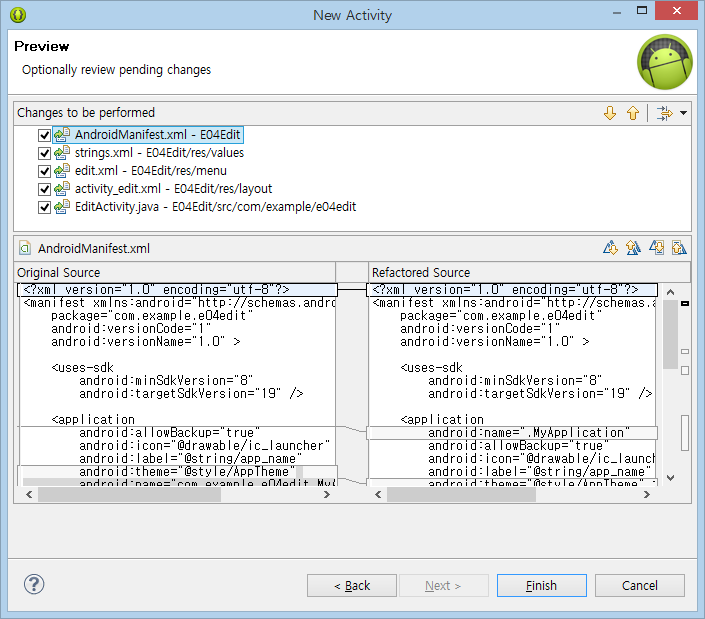
Blank Activity 선택

Next 버튼 클릭



|  |  |  |
| --- | --- | --- |
| 입력 항목 | 입력 값 | 입력 항목 설명 |
| Project: | E04Edit | 현재 프로젝트명 |
| Activity Name: | EditActivity | 액티비티 클래스명 |
| Layout Name: | activity\_edit | 레이아웃 리소스 XML 파일명 |
| Title: | Edit Memo | 액티비티 제목줄에 표시될 문자열 |
| Launcher Activity: | 체크 안함 | 이 액티비티를 앱을 시작할 때 처음 실행되는 시작 액티비티로 설정할 것인가? |
| Hierarchical Parent: | com.example.e04edit.MainActivity | 이 액티비티에서 백 버튼을 눌렀을 때 자동으로 넘어갈 목적지 액티비티 클래스 지정 |

위와 같이 입력하고 Next 버튼 클릭

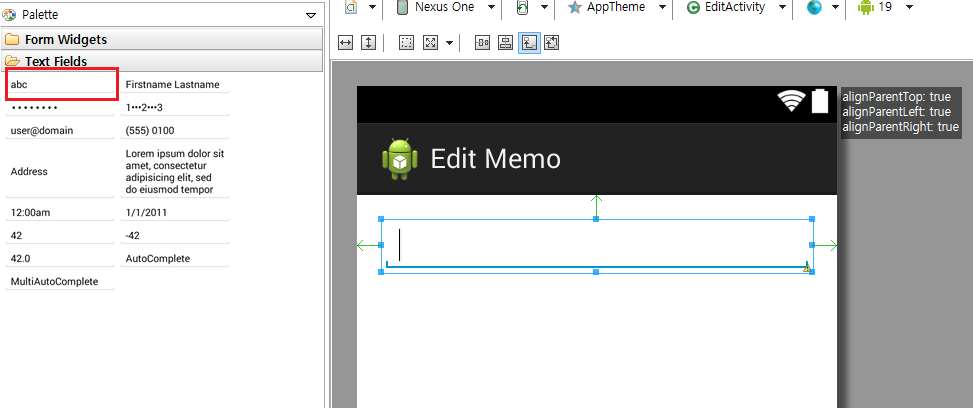


자동으로 수정될 소스 파일 목록과 변경 내용을 보여주고 있음.

그냥 Finish 버튼 클릭.

## activity\_edit.xml 수정

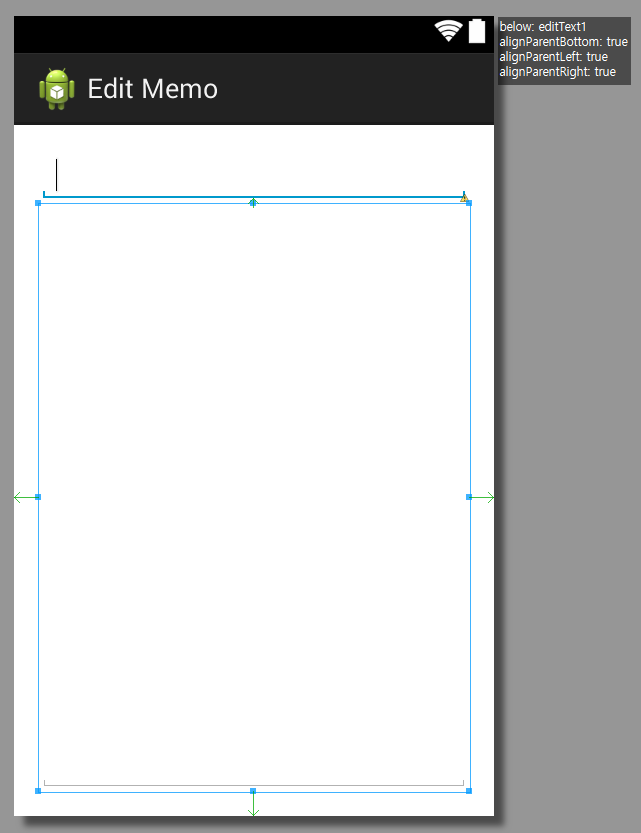
새로 추가된 EditActivity의 레이아웃 리소스 XML 파일인 activity\_edit.xml을 수정하자.



화면 왼쪽의 Palette 창에서 EditText를 드래그 드롭하여 위 그림처럼 추가하자.

editText1의 속성

|  |  |  |
| --- | --- | --- |
| 속성명 | 속성값 | 속성 설명 |
| alignParentTop: | true | 부모의 위 경계에 붙어서 배치된다. |
| alignParentLeft: | true | 부모의 왼쪽 경계에 붙어서 배치된다. |
| alignParentRight: | true | 부모의 오른쪽 경계에 붙어서 배치된다. |



EditText를 하나 더 드래그 드롭하여 추가하자.

추가된 EditText의 상하 좌우 경계를 부모의 경계까지 드래그하여 위 그림처럼 배치되도록 하자.

editText2의 속성

|  |  |  |
| --- | --- | --- |
| 속성명 | 속성값 | 속성 설명 |
| below: | editText1 | editText1의 아래애 붙어서 배치된다. |
| alignParentLeft: | true | 부모의 왼쪽 경계에 붙어서 배치된다. |
| alignParentRight: | true | 부모의 오른쪽 경계에 붙어서 배치된다. |
| alignParentBottom: | true | 부모의 아래쪽 경계에 붙어서 배치된다. |

단축키 Ctrl+S를 눌러서 저장하고 편집창을 닫자.

## EditActivity.java 수정

### src/com.example.e04edit/EditActivity.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 | package com.example.e04edit;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.widget.EditText;  public class EditActivity extends ActionBarActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_edit);  EditText editText1 = (EditText)findViewById(R.id.editText1);  EditText editText2 = (EditText)findViewById(R.id.editText2);  MyApplication myApplication = (MyApplication)getApplicationContext();  MyMemo memo = myApplication.getMyMemo();  editText1.setText(memo.getTitle());  editText2.setText(memo.getBody());  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.edit, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

MainActivity.java의 소스코드와 비교할 때, TextView가 EditText로 바뀐 것을 제외하고 동일하다.

MyMemo 객체 내부의 title과 body 값을 EditText에 채워서 화면에 보여주는 코드만 위 소스코드에 구현되었고,

EditText에서 수정된 문자열을 다시 MyMemo 객체의 title과 body에 저장하는 코드는 아직 구현되지 않았다.

이 코드는 나중에 추가하자.

# 다른 액티비티 호출하기

이제 MainActivity의 메뉴에서 '편집'을누르면 EditActivity 화면으로 넘어가는 기능을 구현하자.

MainActivity.java 파일의 줄26은 다음과 같다.

getMenuInflater().inflate(R.menu.main, menu);

EditActivity.java 파일의 줄26은 다음과 같다.

getMenuInflater().inflate(R.menu.edit, menu);

위 두 라인은 액티비티의 메뉴를 생성하는 일을 하고 있다.

getMenuInflater()는 액티비티의 메소드이다. 이 메소드는 메뉴 인플레이터 객체를 리턴해 준다.

메뉴 인플레이터 객체의 inflate 메소드가 바로 메뉴를 만드는 일을 한다.

inflate 메소드의 첫째 파라미터는 메뉴 리소스 XML 파일의 id이다.

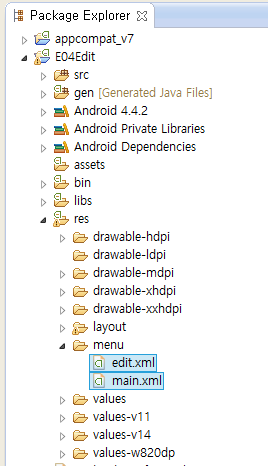
즉 MainActivity 액티비티의 메뉴 리소스 id는 R.menu.main 이고,

EditActivity 액티비티의 메뉴 리소스 id는 R.menu.edit 이다.

id가 R.menu.main 인 메뉴 리소스 파일은 res/menu/main.xml 이고,

id가 R.menu.edit 인 메뉴 리소스 파일은 res/menu/edit.xml 이다.

이클립스에서 액티비티가 생성될 때, 메뉴 리소스 XML 파일도 같이 생성되었다.



res/menu/main.xml 파일에 '편집' 메뉴를 추가해야 하고,

res/menu/edit.xml 파일에 '저장' 메뉴를 추가해야 한다.

## 메뉴 아이콘 만들기

메뉴 항목을 추가하기 전에 먼저 메뉴 항목에 표시할 아이콘을 만들자.

안드로이드 메뉴 항목에 표시할 아이콘의 공식 명칭은 액션바 아이콘(Action Bar Icon) 이다.

액티비티의 상단 제목줄의 공식 명칭이 액션바(Action Bar)이다.

다행스럽게도 안드로이드 개발자 사이트에서 액션바 아이콘 팩을 무료로 제공해 주고 있다.

다행스럽게도 안드로이드 개발자 사이트에서 "Action Bar Icon Pack"으로 검색하면 찾을 수 있다.

링크: <https://developer.android.com/design/downloads/index.html>

다운로드 받은 압축 파일을 풀면 Action Bar Icons 폴더 아래에 holo\_dark 폴더와 holo\_light 폴더가 있다.

액션바가 검정색을 때 holo\_dark 폴더 아래의 아이콘을, 액션바가 흰색일 대 holo\_light 폴더 아래의 아이콘을 사용하면 된다.

holo\_dark 폴더 아래에 144개의 폴더가 있다. 폴더 하나가 아이콘 하나이다. 즉 144개의 액션바 아이콘이 들어있다. 아이콘 폴더 아래에는 4개의 폴더가 있다 (drawable-hdpi, drawable-mdpi, drawable-xhdpi, drawable-xxhdpi). 하나의 아이콘이 해상도 별로 다른 크기의 4 파일로 제공된다.

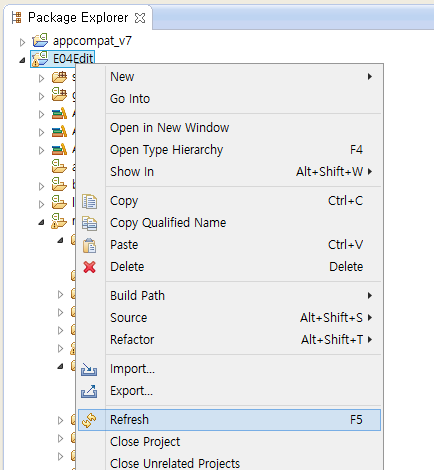
편집 메뉴의 아이콘으로 "Action Bar Icons/holo\_dark/01\_core\_edit" 아이콘을 사용하고,

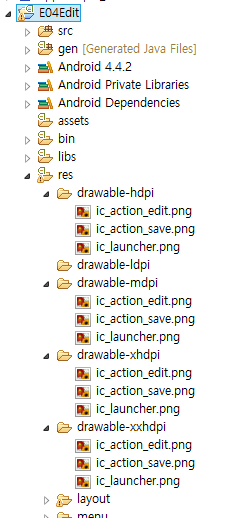
저장 메뉴의 아이콘으로 "Action Bar Icons/holo\_dark/05\_content\_save" 아이콘을 사용하자.

이 두 아이콘의 파일들을 프로젝트의 res 폴더 아래 drawable-hdpi, drawable-mdpi, drawable-xhdpi, drawable-xxhdpi 폴더에 복사하자.

복사한 다음 이클립스의 Package Explorer 창에서 E04Edit 프로젝트를 우클릭하고 Refresh 메뉴를 클릭하면

추가된 아이콘 파일들이 Package Explorer 창에 표시될 것이다.





이렇게 추가된 아이콘 파일들의 아이디는

Java 소스코드에서 R.drawable.ic\_action\_edit, R.drawable.ic\_action\_save 이고

XML 파일에서 @drawable/ic\_action\_edit, @drawable/ic\_action\_save 이다.

## 메뉴 제목 문자열 리소스 만들기

앱이 다국어를 지원할 수 있으려면 메뉴, 버튼 등 화면에 표시되는 문자열을 문자열 리소스로 구현해야 한다.

문자열 리소스 파일은 res/values/strings.xml 이다.

문자열 리소스를 사용해서 구현한 경우에 다국어를 지원하기 위해서 추가로 할 일은 strings.xml 파일만 여러 나라 언어로 여러개 만들기만 하면 되니 무척 간단하다.

### res/values/strings.xml 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?xml version="1.0" encoding="utf-8"?>  <resources>  <string name="app\_name">E04Edit</string>  <string name="hello\_world">Hello world!</string>  <string name="action\_settings">Settings</string>  <string name="title\_activity\_edit">Edit Memo</string>  <string name="action\_edit">Edit</string>  <string name="action\_save">Save</string>  </resources> |

(줄7~8)의 문자열 리소스가 추가되었다.

이 문자열 리소스의 아이디는 Java 소스코드에서 R.string.action\_edit, R.string.action\_save 이고,

XML 파일에서 @string/action\_edit, @string/action\_save 이다.

## 편집 메뉴 만들기

MainActivity 액티비티의 메뉴 파일인 res/menu/main.xml 파일에 메뉴 항목을 추가하자.

### res/menu/main.xml 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <menu xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  tools:context="com.example.e04edit.MainActivity" >  <item android:id="@+id/action\_edit"  android:icon="@drawable/ic\_action\_edit"  android:title="@string/action\_edit"  app:showAsAction="ifRoom" />  <item  android:id="@+id/action\_settings"  android:orderInCategory="100"  android:title="@string/action\_settings"  app:showAsAction="never"/>  </menu> |

(줄6~9)의 메뉴 항목이 추가되었다.

|  |  |  |
| --- | --- | --- |
| 항목 | 값 | 항목 설명 |
| android:id | @+id/action\_edit | 이 메뉴 항목의 id를 정의한다.  Java 코드에서는 R.id.action\_edit 이다 |
| android:icon | @drawable/ic\_action\_edit | 메뉴 항목에 표시할 아이콘의 리소스 id를 지정한다 |
| android:title | @string/action\_edit | 메뉴 제목으로 표시할 문자열 리소스 id를 지정한다 |
| app:showAsAction | ifRoom | 액션바의 공간에 여유가 있으면 메뉴 항목의 아이콘을 액션바에 표시하라는 뜻 |

액티비티 상단의 제목줄의 공식 명칭은 액션바(Action Bar)이다.

이 액션바의 공간에 여유가 있을 때, 메뉴 항목의 아이콘이 표시될 수 있다.

액션바에 표시된 메뉴 항목의 공식 명칭은 액션(Action)이다.

옛날 안드로이드에는 액션바가 없어서 전부 메뉴로 표시되었다.

안드로이드 3.0부터 액션바가 생겼고 메뉴를 액션이라고 부르기 시작했다.

즉 옵션 메뉴, 메뉴, 액션이 모두 같은 말이다.

안드로이드 3.0 이전에는 옵션 메뉴(option menu), 메뉴(menu)라고 불렀고,

안드로이드 3.0 부터는 액션이라고 부른다.

app:showAsAction 값이 ifRoom이면 액션바에 여유 공간(room)이 있을 때 액션으로 표시하라는 뜻이고,

줄14에서 처럼 never이면 액션으로 표시하지 말라는 뜻이고,

always이면 언제나 표시하라는 뜻이다.

## 메뉴 이벤트 핸들러 구현하기

메뉴가 클릭되면 메뉴 이벤트 핸들러 메소드가 호출된다.

액션바의 액션이 클릭되어도 메뉴 이벤트 핸들러 메소드가 호출된다.

메뉴 이벤트 핸들러는 액티비티의 onOptionsItemSelected 메소드이다.

### src/com.example.e04edit/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  38  39  40  41  42  43 | package com.example.e04edit;  import android.support.v7.app.ActionBarActivity;  import android.content.Intent;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.widget.TextView;  public class MainActivity extends ActionBarActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  TextView textView1 = (TextView)findViewById(R.id.textView1);  TextView textView2 = (TextView)findViewById(R.id.textView2);  MyApplication myApplication = (MyApplication)getApplicationContext();  MyMemo memo = myApplication.getMyMemo();  textView1.setText(memo.getTitle());  textView2.setText(memo.getBody());  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  } else if (id == R.id.action\_edit) {  Intent intent = new Intent(this, EditActivity.class);  startActivity(intent);  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄36~40)의 코드가 추가되었다. 이 추가된 라인을 제외하고 onOptionsItemSelected 메소드의 나머지 라인은 자동으로 생성되었다.

(줄32) 메뉴 항목이나 액션바의 액션이 클릭되면 액티비티의 onOptionsItemSelected 메소드가 호출된다.

이때 클릭된 메뉴 항목이 파라미터로 전달된다.

(줄33) 메뉴 항목 객체의 getItemId 메소드를 호출하여 메뉴 항목에 대한 id를 얻는다.

이 id는 메뉴 리소스 XML 파일에서 android:id 속성으로 부여한 값이다.

res/menu/main.xml 파일의 두 메뉴 항목의 아이디는 각각 @+id/action\_edit, @+id/action\_settings 이다.

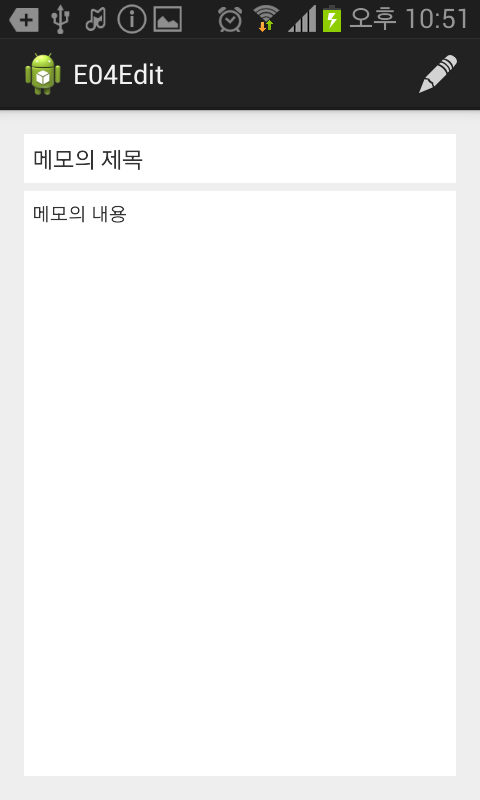
이 아이디는 Java 코드에서 각각 R.id.action\_edit, R.id.action\_settings 이다.

(줄37~38) EditActivity 액티비티를 호출할 때는 이와 같이 구현해야 한다고 일단 외우고 넘어가자.

다른 액티비티를 호출할 때는 Intent 객체를 생성하고 startActivity 메소드를 호출해야 한다.

startActivity 메소드는 액티비티 클래스의 메소드이다.

# 실행

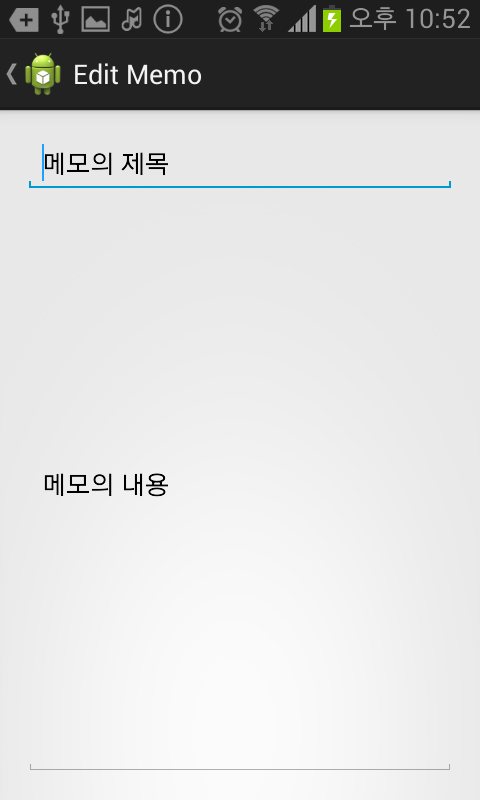


액션바에 편집 액션의 아이콘이 표시되었다.

이 아이콘을 클릭하면 EditActivity가 호출된다.

### 소스코드

E04Edit\_#2.zip



EditActivity 화면이다.

액션바의 왼쪽 끝 < 아이콘을 클릭하거나, 안드로이드 기기의 백(back) 버튼을 누르면 MainActivity로 돌아간다.

EditActivity를 생성할 때, Hierarchical Parent 속성으로 MainActivity를 지정해 주었기 때문에, 액션바의 왼쪽 끝 < 아이콘을 클릭하면 MainActivity로 돌아간다.

위 화면에서 메모의 내용이 수직 방향으로 한가운데 보이는 것이 어색하다.

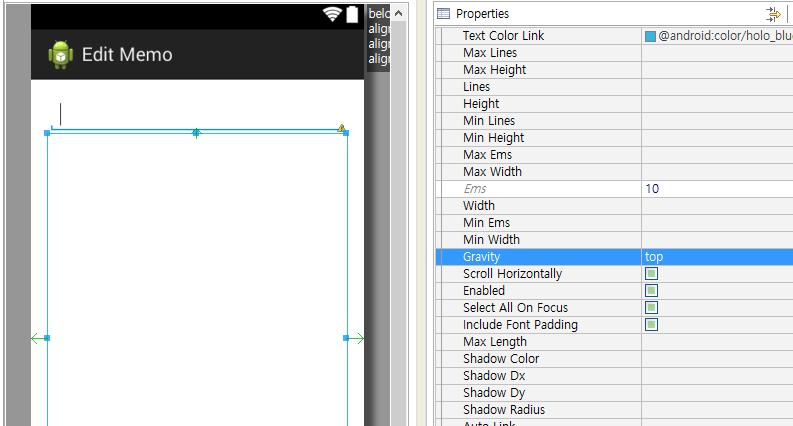
EditText의 상단에 표시되도록 수정하자.

그리고 EditActivity에 저장 메뉴를 추가하자.

# EditActivity에 메뉴 추가

## activity\_edit.xml 수정

### res/layout/activity\_edit.xml



editText2를 선택하고 오른쪽 Properties 창에서

TextView – Gravity 속성값을 center\_vertical 에서 top으로 변경하자.

이제 editText2의 내용이 수직 방향으로 한가운데가 아니고 위에 붙어서 표시된다.

Gravity 속성은 TextView나 EditText의 내용 문자열의 수평, 수직 방향 배열을 지정한다.

## 저장 메뉴 만들기

### res/menu/edit.xml 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <menu xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  tools:context="com.example.e04edit.EditActivity" >  <item android:id="@+id/action\_save"  android:icon="@drawable/ic\_action\_save"  android:title="@string/action\_save"  app:showAsAction="ifRoom" />  <item  android:id="@+id/action\_settings"  android:orderInCategory="100"  android:title="@string/action\_settings"  app:showAsAction="never"/>  </menu> |

(줄6~9)의 메뉴 항목이 추가되었다.

|  |  |  |
| --- | --- | --- |
| 항목 | 값 | 항목 설명 |
| android:id | @+id/action\_save | 이 메뉴 항목의 id를 정의한다.  Java 코드에서는 R.id.action\_edit 이다 |
| android:icon | @drawable/ic\_action\_save | 메뉴 항목에 표시할 아이콘의 리소스 id를 지정한다 |
| android:title | @string/action\_save | 메뉴 제목으로 표시할 문자열 리소스 id를 지정한다 |
| app:showAsAction | ifRoom | 액션바의 공간에 여유가 있으면 메뉴 항목의 아이콘을 액션바에 표시하라는 뜻 |

## 메뉴 이벤트 핸들러 구현하기

메뉴 항목이 액션바의 액션이 클릭되면 메뉴 이벤트 핸들러 메소드가 호출된다.

메뉴 이벤트 핸들러는 액티비티의 onOptionsItemSelected 메소드이다.

### src/com.example.e04edit/EditActivity.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  38  39  40  41  42  43  44  45  46  47 | package com.example.e04edit;  import android.support.v4.app.NavUtils;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.widget.EditText;  public class EditActivity extends ActionBarActivity {  EditText editText1;  EditText editText2;  MyMemo memo;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_edit);  editText1 = (EditText)findViewById(R.id.editText1);  editText2 = (EditText)findViewById(R.id.editText2);  MyApplication myApplication = (MyApplication)getApplicationContext();  memo = myApplication.getMyMemo();  editText1.setText(memo.getTitle());  editText2.setText(memo.getBody());  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.edit, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  } else if (id == R.id.action\_save) {  memo.setTitle(editText1.getText().toString());  memo.setBody(editText2.getText().toString());  NavUtils.navigateUpFromSameTask(this);  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄12~14) editText1, editText2, memo를 onCreate와 onOptionsItemSelected 메소드에서 사용해야 해서 멤버 변수로 구현하였다.

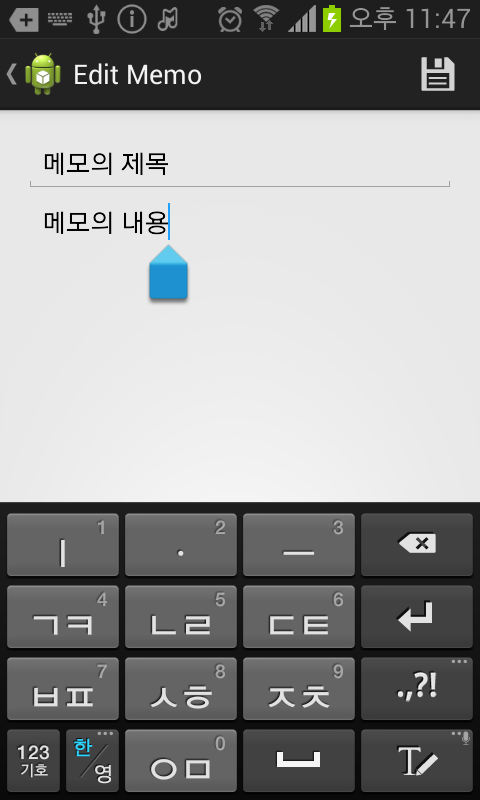
(줄20, 21, 34) editText1, editText2, memo 가 멤버 변수이므로 여기서 대입된 값을 나중에 메뉴가 클릭되어 실행되는 onOptionsItemSelected 메소드에서도 사용할 수 있다.

(줄40, 41) editText1, editText2에 입력된 값을 꺼내서 멤버 변수 memo가 참조하는 객체에 저장한다.

EditText의 getText() 메소드의 리턴 타입은 CharSequence이다. String 타입의 문자열을 얻기 위해 toString() 메소드를 호출하였다.

(줄42) Hierarchical Parent 속성으로 지정된 Activity로 돌아가기 위한 메소드 호출이다. EditActivity를 생성할 때, Hierarchical Parent 속성으로 MainActivity를 지정했으므로 MainActivity로 돌아가게 된다.

# 실행



제목이나 내용을 수정하고 액션바의 저장 아이콘을 클릭하면 수정된 내용이 MyMemo 객체에 저장되고

MainActivity로 넘어가게 된다.

### 소스코드

E04Edit\_#3.zip

### 개선 필요

한 화면이 넘어갈 정도로 여러 줄의 텍스트를 입력해 보자.

EditText에는 스크롤 기능이 있어서 여러 줄의 텍스트를 입력하는데 지장이 없다.

그런데 RelativeLayout과 TextView에는 스크롤 기능이 없어서 한 화면이 넘어가는 텍스트는 짤려서 보인다.

# 화면 스크롤 기능 추가

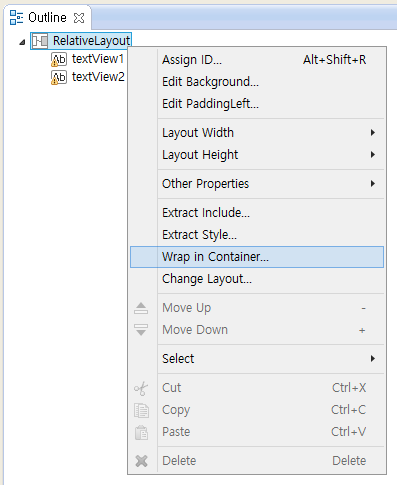
한 화면이 넘어가는 텍스트도 스크롤하며 볼 수 있도록 MainActivity 화면에 스크롤 기능을 추가하자.

화면의 배치를 담당하는 RelativeLayout 클래스에는 스크롤 기능이 없다.

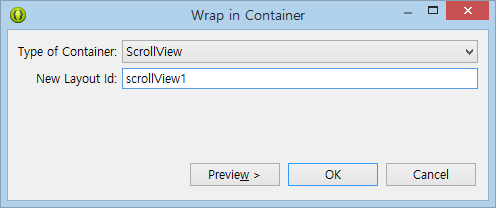
스크롤 기능이 없는 RelativeLayout 위에 ScrollView 클래스를 추가하면, 즉 ScrollView가 루트(root)가 되고 그 자식으로 RelativeLayout이 내려가면, 스크롤 기능이 추가된다.

activity\_main.xml 레이아웃 리소스 XML 파일을 수정하여 ScrollView를 RelativeLayout의 부모로 추가하자.

## activity\_main.xml 수정

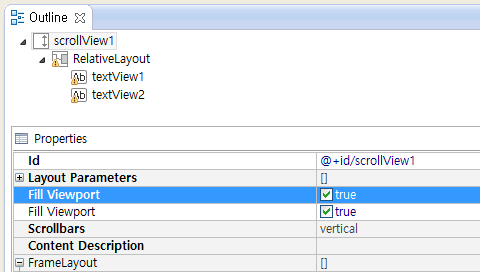


Outline 창에서 RelativeLayout을 우클릭하고 'Wrap In Container' 메뉴 클릭.



|  |  |  |
| --- | --- | --- |
| 입력 항목 | 값 | 설명 |
| Type of Container: | ScrollView | 부모로 추가할 클래스 선택 |
| New Layout ud: | scrollView1 | 새 아이디 입력 |

추가된 scrollView1의 FillViewport 속성을 true로 변경



## 실행

한 화면이 넘어갈 정도로 여러 줄의 텍스트를 입력해 보고 MainActivity에서 스크롤 해보자.

### 소스코드

E04Edit\_#4.zip

# 서식 변경

MainActivity의 화면 모양과 EditActivity의 화면 모양을 통일하려고 함.

배경색, 글자크기, 여백 등의 서식 속성값을 동일하게 주면 같은 모양이 된다.

그래서 두 액티비티의 RelativeLayout 객체 서식 속성값을 동일하게 주려고 함.

EditText 클래스는 TextView 클래스의 자식 클래스이다.

그래서 TextView 클래스의 속성이 EditText에도 상속되어 있다.

그래서 MainActivity의 TextView의 서식 속성값을 EditActivity의 EditText에도 동일하게 주려고 함.

## activity\_main.xml 수정

### res/layout/activity\_main.xml 변경 전

|  |  |
| --- | --- |
| 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 | <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:id="@+id/scrollView1"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:fillViewport="true" >  <RelativeLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="#EEEEEE"  android:paddingBottom="@dimen/activity\_vertical\_margin"  android:paddingLeft="@dimen/activity\_horizontal\_margin"  android:paddingRight="@dimen/activity\_horizontal\_margin"  android:paddingTop="@dimen/activity\_vertical\_margin"  tools:context="com.example.e04edit.MainActivity" >  <TextView  android:id="@+id/textView1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_alignParentTop="true"  android:layout\_marginBottom="5dp"  android:background="#FFFFFF"  android:padding="5dp"  android:text="TextView" />  <TextView  android:id="@+id/textView2"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentBottom="true"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_below="@+id/textView1"  android:background="#FFFFFF"  android:padding="5dp"  android:text="TextView" />  </RelativeLayout>  </ScrollView> |

(줄12~15) padding 속성은 경계선 안쪽 여백을 의미한다.

상,하,좌,우 4방향의 여백을 다르게 주려면 paddingTop, paddingBottom, paddingLeft, paddingRight 속성을 사용한다. 4방향의 여백을 동일하게 주려면 padding 속성을 사용하면 된다.

padding 속성값을 5dp로 변경하자.

(줄18, 30) TextView에 textSize 속성값 18sp을 추가하자.

### res/layout/activity\_main.xml 변경 후

|  |  |
| --- | --- |
| 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 | <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:id="@+id/scrollView1"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:fillViewport="true" >  <RelativeLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="#EEEEEE"  android:padding="5dp"  tools:context="com.example.e04edit.MainActivity" >  <TextView  android:id="@+id/textView1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_alignParentTop="true"  android:layout\_marginBottom="5dp"  android:background="#FFFFFF"  android:padding="5dp"  android:textSize="18sp"  android:text="TextView" />  <TextView  android:id="@+id/textView2"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentBottom="true"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_below="@+id/textView1"  android:background="#FFFFFF"  android:padding="5dp"  android:textSize="18sp"  android:text="TextView" />  </RelativeLayout>  </ScrollView> |

(줄12) padding 속성이 하나로 줄었다

(줄25,38) textSize 속성이 추가되었다.

## activity\_edit.xml 수정

### res/layout/activity\_edit.xml 변경 전

|  |  |
| --- | --- |
| 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 | <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:paddingBottom="@dimen/activity\_vertical\_margin"  android:paddingLeft="@dimen/activity\_horizontal\_margin"  android:paddingRight="@dimen/activity\_horizontal\_margin"  android:paddingTop="@dimen/activity\_vertical\_margin"  tools:context="com.example.e04edit.EditActivity" >  <EditText  android:id="@+id/editText1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_alignParentTop="true"  android:ems="10" >  <requestFocus />  </EditText>  <EditText  android:id="@+id/editText2"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentBottom="true"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_below="@+id/editText1"  android:ems="10"  android:gravity="top" />  </RelativeLayout> |

### res/layout/activity\_edit.xml 변경 후

|  |  |
| --- | --- |
| 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 | <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="#EEEEEE"  android:padding="5dp"  tools:context="com.example.e04edit.EditActivity" >  <EditText  android:id="@+id/editText1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_alignParentTop="true"  android:layout\_marginBottom="5dp"  android:background="#FFFFFF"  android:padding="5dp"  android:textSize="18sp"  android:ems="10" >  <requestFocus />  </EditText>  <EditText  android:id="@+id/editText2"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentBottom="true"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_below="@+id/editText1"  android:background="#FFFFFF"  android:padding="5dp"  android:textSize="18sp"  android:ems="10"  android:gravity="top" />  </RelativeLayout> |

(줄5~6) 추가됨

(줄16~19) 추가됨

(줄33~35) 추가됨

## 액티비티 화면 전환 애니메이션

액티비티의 화면이 다른 액티비티 화면으로 전환할 때 애니메이션 효과가 기본으로 실행되고 있다.

어떤 경우에는 애니메이션 효과 없이 신속하게 화면 전환되는 것이 바람직하다.

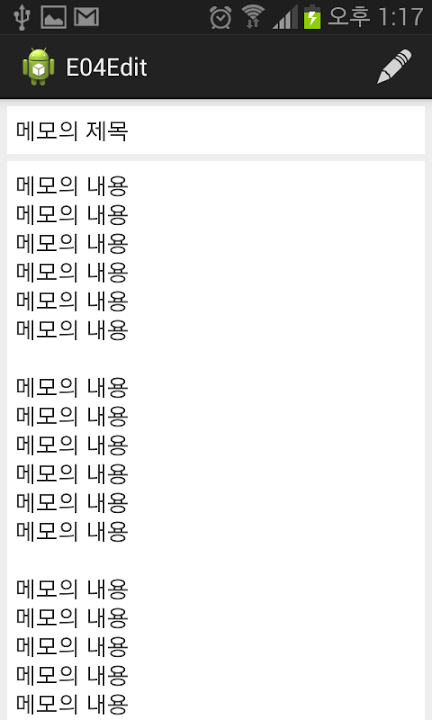
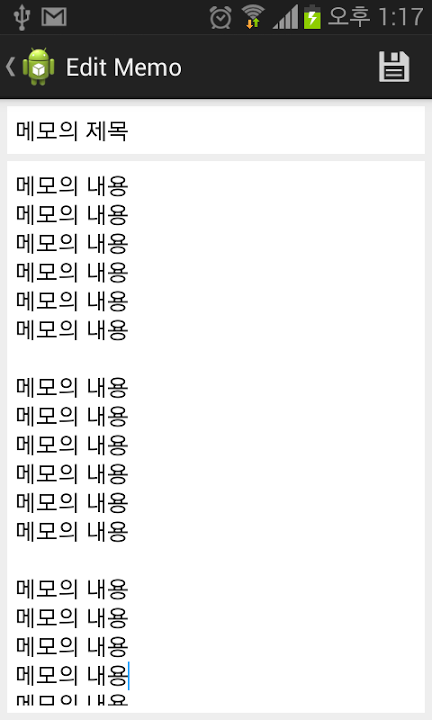
이 애니메이션 효과를 제거하자.

### res/values/style.xml 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | <resources>  <!--  Base application theme, dependent on API level. This theme is replaced  by AppBaseTheme from res/values-vXX/styles.xml on newer devices.  -->  <style name="AppBaseTheme" parent="Theme.AppCompat.Light">  <!--  Theme customizations available in newer API levels can go in  res/values-vXX/styles.xml, while customizations related to  backward-compatibility can go here.  -->  </style>  <!-- Application theme. -->  <style name="AppTheme" parent="AppBaseTheme">  <!-- All customizations that are NOT specific to a particular API-level can go here. -->  <item name="android:windowAnimationStyle">@null</item>  </style>  </resources> |

(줄18) windowAnimationStyle 속성값을 null로 설정함.

# 결과

서식을 통일한 덕분에 두 액티비티의 화면 형태가 같다.

화면 전환이 빠르기 때문에, 사용자는 화면이 전환되었다고 느끼기 보다는, 편집 모드에 진입했다고 느낄 것이다.

### 소스코드

E04Edit\_#5.zip