메모 앱 만들기 #1

2014-11-06

이승진

**목차**

[1. MyMemo 프로젝트 2](#_Toc404313166)

[1) 프로젝트 생성 2](#_Toc404313167)

[2) 런처 아이콘 변경하기 2](#_Toc404313168)

[2. MainActivity 수정 3](#_Toc404313169)

[1) activity\_main.xml 파일 수정 3](#_Toc404313170)

[2) list\_item.xml 파일 생성 4](#_Toc404313171)

[3) 메뉴 아이콘 만들기 4](#_Toc404313172)

[4) 메뉴 제목 문자열 리소스 만들기 4](#_Toc404313173)

[5) New 메뉴 만들기 5](#_Toc404313174)

[6) LectureDBOpenHelper.java 생성 6](#_Toc404313175)

[7) MainActivity.java 수정 8](#_Toc404313176)

[3. EditActivity 액티비티 생성 10](#_Toc404313177)

[1) 프로젝트에 EditActivity 액티비티 추가 생성 10](#_Toc404313178)

[2) activity\_edit.xml 수정 11](#_Toc404313179)

[3) 저장 메뉴 만들기 13](#_Toc404313180)

[4) EditActivity.java 수정 14](#_Toc404313181)

[5) 실행 14](#_Toc404313182)

[4. 편집 기능 추가 15](#_Toc404313183)

[1) MainActivity.java 수정 15](#_Toc404313184)

[2) EditActivity.java 수정 17](#_Toc404313185)

[3) 실행 19](#_Toc404313186)

[5. 삭제 기능 추가 20](#_Toc404313187)

[1) 메뉴 아이콘 만들기 20](#_Toc404313188)

[2) 메뉴 제목 문자열 리소스 만들기 20](#_Toc404313189)

[3) Remove 메뉴 만들기 20](#_Toc404313190)

[4) EditActivity.java 수정 21](#_Toc404313191)

[5) 실행 22](#_Toc404313192)

**학습목표**

간단한 메모장 앱을 만든다.

다음과 같은 기능을 구현한다

- 메모를 데이터베이스에 저장

- 메모 목록 보기

- 새 메모 작성 기능

- 메모 편집

- 메모 삭제

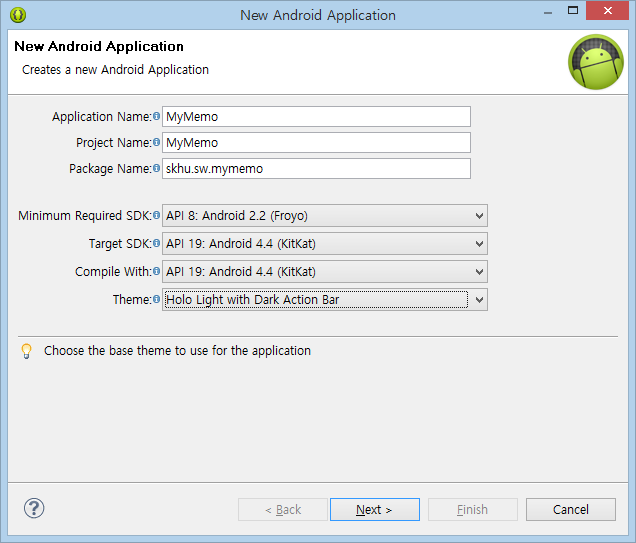
# MyMemo 프로젝트

간단한 메모 앱을 만들어보자.

먼저 메모 목록을 보여주는 액티비티부터 만들자

## 프로젝트 생성

이클립스 메뉴: File – New – Andoid Applicaton Project



위와 같이 입력하고 Next를 계속 눌러서 진행하자.

## 런처 아이콘 변경하기

런처 아이콘을 변경하려면, 먼저 drawable-XXX 폴더들에 들어있는 ic\_launcher.png 파일을 모두 삭제한다.

그리고 144 x 144 픽셀 크기의 png 파일을 만들어서, drawable-xxhdpi 폴더에 ic\_launcher.png 파일명으로 저장한다.



강의노트 웹사이트의 소스코드/MyMemo 폴더 아래에서 이 파일을 다운로드 받을 수 있다.

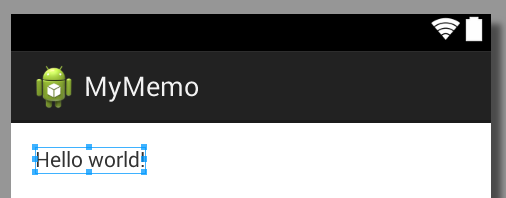
MyMemo 예제 프로젝트의 런처 아이콘을 이 파일로 교체하자.

런처 아이콘의 배경은 흰색이 아니라 투명해야 한다.

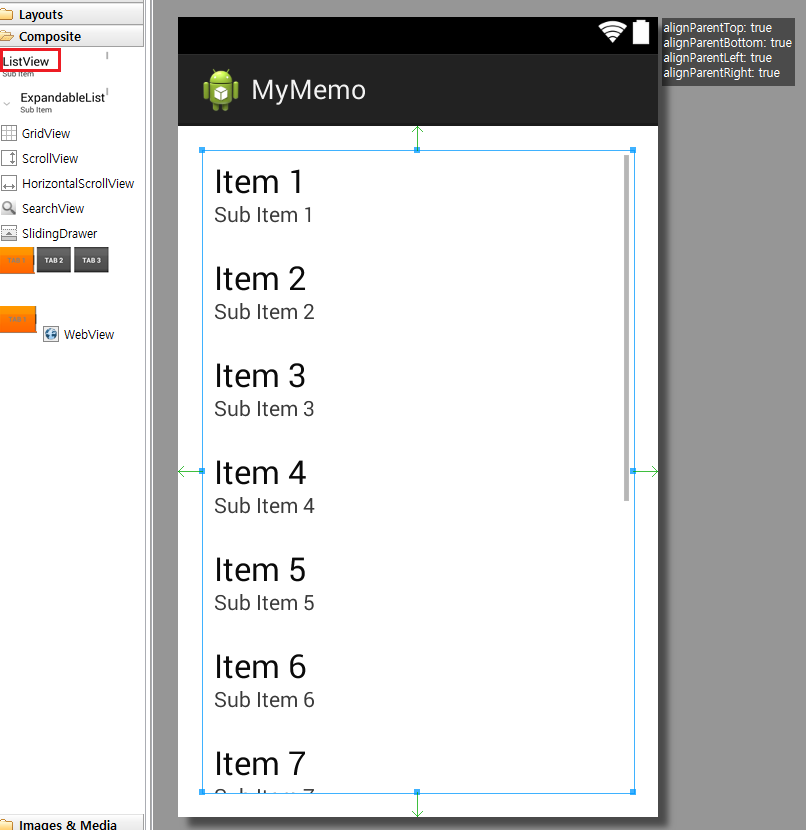
# MainActivity 수정

## activity\_main.xml 파일 수정

MainActivity의 레이아웃 리소스 파일인 activity\_main.xml 파일을 열어서 수정하자



이미 만들어져 있는 "Hello world" TextView를 클릭하여 선택하고 Delete 키를 눌러서 제거하자.



eclipse 화면 왼쪽의 Palette 창에서 Composite 탭 아래의 ListView를 드래그 드롭하여 액티비티에 추가한다.

그리고 ListView의 위, 아래, 왼쪽, 오른쪽 4개의 변(border)를 마우스로 드래그하여 액티비티의 끝까지 끌어다 놓으면 위 화면 왼쪽에 보이는 것처럼 레이아웃 속성이 설정된다.

alignParentTop: true

alignParentBottom: true

alignParentLeft: true

alignParentRight: true

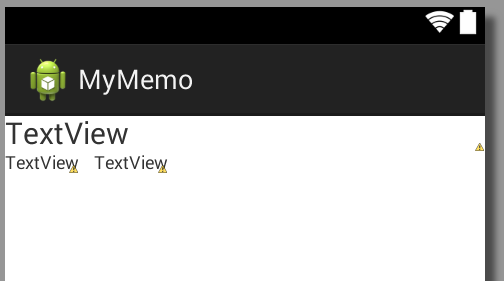
activity\_main.xml 파일을 저장하고 닫는다.

## list\_item.xml 파일 생성

MainActivity의 ListView에 표시할 데이터 항목 하나를 표시하기 위한 뷰(View)들의 레이아웃을 정의하자.

list\_item.xml 레이아웃 리소스 파일을 생성하자.

"03 SQLite Database.docx" 파일의 5페이지에서 list\_item.xml 생성을 설명한 것과 똑같은 방법으로 list\_item.xml 파일을 생성하자.



똑같은 방법으로 다시 만드는 것이 귀찮으면, list\_item.xml 파일을 생성한 다음 내용을 클립보드로 복사해서 붙혀넣기 해도 된다.

## 메뉴 아이콘 만들기

"05 Intent.docx" 파일의 16 페이지에서 설명한 것과 똑같은 방법으로 ic\_action\_edit.png 파일과 ic\_action\_save.png 파일을 프로젝트의 리소스에 추가하자

그리고 추가로 "Action Bar Icons/holo\_dark/01\_core\_new" 아래의 ic\_action\_new.png 파일도 프로젝트 리소스에 추가하자.

## 메뉴 제목 문자열 리소스 만들기

"05 Intent.docx" 파일의 17 페이지에서 설명한 것과 똑같은 방법으로 Edit, Save 문자열 리소스를 만들자.

그리고 다음 항목도 추가하자

<string name="action\_new">New</string>

## New 메뉴 만들기

MainActivity 액티비티의 메뉴에 새 메모 작성 메뉴를 추가하자.

### res/menu/main.xml 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <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="skhu.sw.mymemo.MainActivity" >  <item  android:id="@+id/action\_new"  android:icon="@drawable/ic\_action\_new"  android:title="@string/action\_new"  app:showAsAction="ifRoom"/>  <item  android:id="@+id/action\_settings"  android:orderInCategory="100"  android:title="@string/action\_settings"  app:showAsAction="never"/>  </menu> |

(줄6~10)의 메뉴 항목이 추가되었다.

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

## LectureDBOpenHelper.java 생성

eclipse Package Explorer 창에서 MyMemo 프로젝트 아래 src 폴더 아래 skhu.sw.mymemo 패키지를 우클릭하고 New – Class 메뉴를 클릭하면 New Java Class 생성 대화상자가 표시된다.

LectureDBOpenHelper 클래스를 생성하자.

### src/skhu.sw/LectureDBOpenHelper.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 skhu.sw.mymemo;  import android.content.Context;  import android.database.sqlite.SQLiteDatabase;  import android.database.sqlite.SQLiteOpenHelper;  public class LectureDBOpenHelper extends SQLiteOpenHelper {  private static final int DB\_VERSION = 2;  private static final String DB\_NAME = "Lecture.db";  private static final String SQL\_CREATE\_LISTITEM\_TABLE =  "CREATE TABLE ListItem ( \_id INTEGER PRIMARY KEY AUTOINCREMENT, "  + "title TEXT, modifiedTime DATETIME DEFAULT (datetime('now','localtime')) );";  private static final String SQL\_CREATE\_MYMEMO\_TABLE =  "CREATE TABLE MyMemo ( \_id INTEGER PRIMARY KEY AUTOINCREMENT, "  + "title TEXT, body TEXT, modifiedTime DATETIME DEFAULT (datetime('now','localtime')) );";    public LectureDBOpenHelper(Context context) {  super(context, DB\_NAME, null, DB\_VERSION);  }  @Override  public void onCreate(SQLiteDatabase db) {  db.execSQL(SQL\_CREATE\_LISTITEM\_TABLE);  db.execSQL(SQL\_CREATE\_MYMEMO\_TABLE);  }  @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  if (oldVersion == 1)  db.execSQL(SQL\_CREATE\_MYMEMO\_TABLE);  }  } |

안드로이드 앱에서 데이터베이스를 사용하려면, (줄7)처럼 SQLiteOpenHelper 클래스를 상속받아서 자식 클래스를 만들고, 생성자 메소드, onCreate 메소드, 그리고 onUpdate 메소드를 구현해야 한다.

(줄19) 생성자 메소드 구현

(줄20) 부모 클래스인 SQLiteOpenHelper 클래스의 생성자 호출이다.

이 생성자 메소드의 두번째 파라미터가 데이터베이스 파일명이고, 마지막 파라미터는 데이터베이스 파일의 버전이다.

(줄9) 데이터베이스 파일의 버전을 정의한 상수 멤버 변수이다

(줄10) 데이터베이스 파일명을 정의한 상수 멤버 변수이다.

Java 언어에서 상수는 static final 멤버 변수로 정의한다.

데이터베이스 파일명은 이전 예제에서 사용했던 그 파일명이고, 파일의 버전은 1에서 2로 올라갔다.

(줄24) onCreate 메소드

이 앱이 실행되는 안드로이드 폰에 "Lecture.db" 데이터베이스 파일이 없다면, 줄24의 onCreate 메소드가 호출된다. onCreate 메소드가 호출되기 직전에 "Lecture.db" 데이터베이스 파일은 자동 생성된다. onCreate 메소드에서는 테이블을 생성하는 SQL 문장을 실행해야 한다.

(줄25) ListItem 테이블을 생성하는 SQL 문장을 실행한다.

(줄26) MyMemo 테이블을 생성하는 SQL 문장을 실행한다.

(줄30) onUpgrade 메소드

폰에 "Lecture.db" 데이터베이스 파일은 이미 있는데, 버전이 2보다 낮다면, 즉 파일의 버전이 1이라면 줄30의 onUpgrade 메소드가 호출된다. upUpgrade 메소드에서는 이전 버전의 데이터베이스를 현재 버전의 구조로 만들기 위한 SQL 문장을 실행해야 한다.

이전 예제에서 버전 1의 Lecture.db에 ListItem 테이블만 만들었었다. 여기에 MyMemo 테이블만 추가로 생성하면 버전 2의 구조가 된다.

(줄32) MyMemo 테이블을 생성하는 SQL 문장을 실행한다.

(줄11~13) ListItem 테이블을 생성하는 SQL 문장을 상수 멤버 변수로 정의하였다.

Java의 문자열에 들어있는 이 SQL 문장을 알아보기 쉽게 정리하면 다음과 같다.

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE ListItem (  \_id INTEGER PRIMARY KEY AUTOINCREMENT,  title TEXT,  modifiedTime DATETIME DEFAULT (datetime('now','localtime'))  ); |

(줄1) 생성할 테이블의 이름은 ListItem 이다.

(줄2) 이름이 \_id인 INTEGER형 필드(field)를 기본키(primary key)로 생성한다.

AUTOINCREMENT는 1,2,3,4... 일련번호 값이 자동으로 부여된다는 것을 뜻한다.

이 테이블에 새 레코드를 추가(insert)할 때 \_id 값은 자동으로 부여된다.

SQLite 엔진에서 정수형 필드 타입은 INTEGER 이다.

(줄3) 이름이 title인 TEXT형 필드를 생성한다.

SQLite 엔진에서 문자열 필드 타입은 TEXT 이다.

문자열은 Unicode로 저장된다.

(줄4) 이름이 modifiedTime인 DATETIME형 필드를 생성한다.

DEFAULT (datetime('now','localtime'))가 의미하는 것은, 이 테이블에 새 레코드를 추가(insert)할 때,

이 필드 값이 현재 시각으로 자동 부여된다는 것이다.

\_id. modifiedTime 두 필드의 값이 SQLite 엔진에 의해 자동으로 부여될 것이므로,

이 테이블에 새 레코드를 추가(insert)할 때 title 필드값만 전달하면 된다.

(줄5) SQLite의 SQL 문장의 끝에 세미콜론(;) 문자가 필요하다.

(줄14~16) MyMemo 테이블을 생성하는 SQL 문장을 상수 멤버 변수로 정의하였다.

이 SQL 문장을 보기 좋게 정리하면 아래와 같다.

CREATE TABLE ListItem (

\_id INTEGER PRIMARY KEY AUTOINCREMENT,

title TEXT,

body TEXT,

modifiedTime DATETIME DEFAULT (datetime('now','localtime'))

);

## MainActivity.java 수정

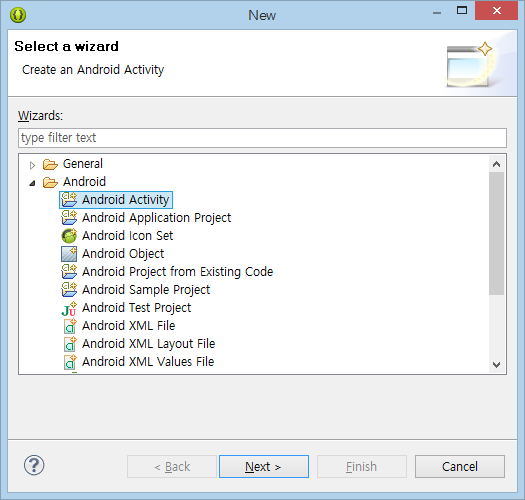
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73 | package skhu.sw.mymemo;  import android.support.v4.widget.CursorAdapter;  import android.support.v4.widget.SimpleCursorAdapter;  import android.support.v7.app.ActionBarActivity;  import android.content.Intent;  import android.database.Cursor;  import android.database.sqlite.SQLiteDatabase;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.widget.ListView;  public class MainActivity extends ActionBarActivity {    LectureDBOpenHelper helper;  SQLiteDatabase db;  Cursor cursor;  SimpleCursorAdapter adapter;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);    adapter = new SimpleCursorAdapter(this,  R.layout.list\_item,  null,  new String[] { "title", "modifiedTime", "\_id" },  new int[] { R.id.textView1, R.id.textView2, R.id.textView3 },  CursorAdapter.FLAG\_REGISTER\_CONTENT\_OBSERVER  );  ListView listView = (ListView)findViewById(R.id.listView1);  listView.setAdapter(adapter);  }  @Override  protected void onResume() {  super.onResume();  helper = new LectureDBOpenHelper(this);  db = helper.getWritableDatabase();  cursor = db.rawQuery("SELECT \* FROM MyMemo", null);  adapter.changeCursor(cursor);  }    @Override  protected void onPause() {  super.onPause();  cursor.close();  db.close();  helper.close();  }  @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\_new) {  Intent intent = new Intent(this, EditActivity.class);  startActivity(intent);  return true;  }  return super.onOptionsItemSelected(item);  }  } |

# EditActivity 액티비티 생성

## 프로젝트에 EditActivity 액티비티 추가 생성

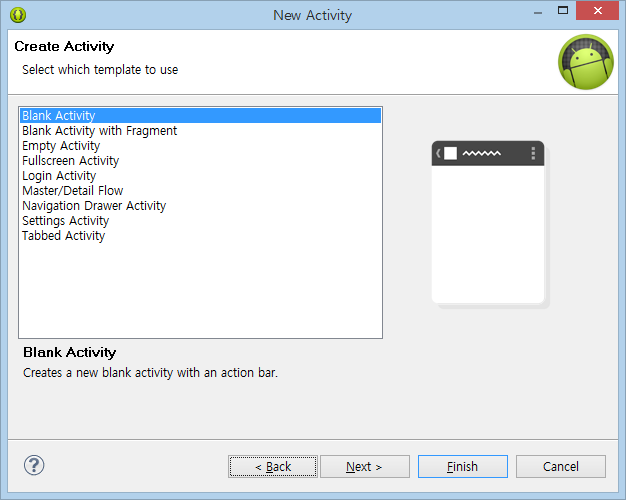
메모를 편집할 수 있는 편집 화면에 해당하는 EditActivity 클래스를 구현하자.

메뉴: File – New – Other



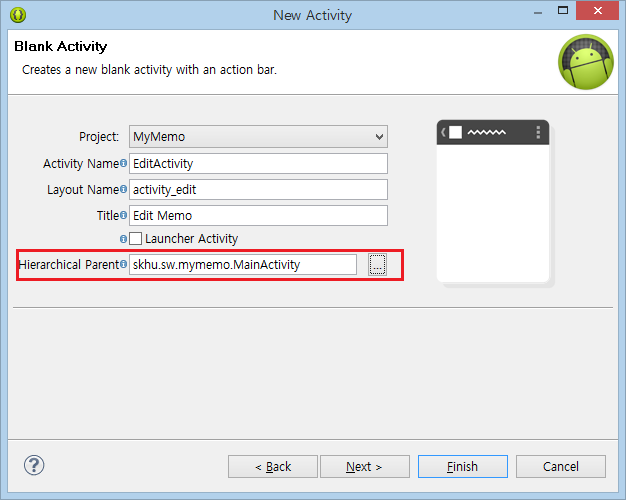
Android Activity 선택

Next 버튼 클릭



Blank Activity 선택

Next 버튼 클릭



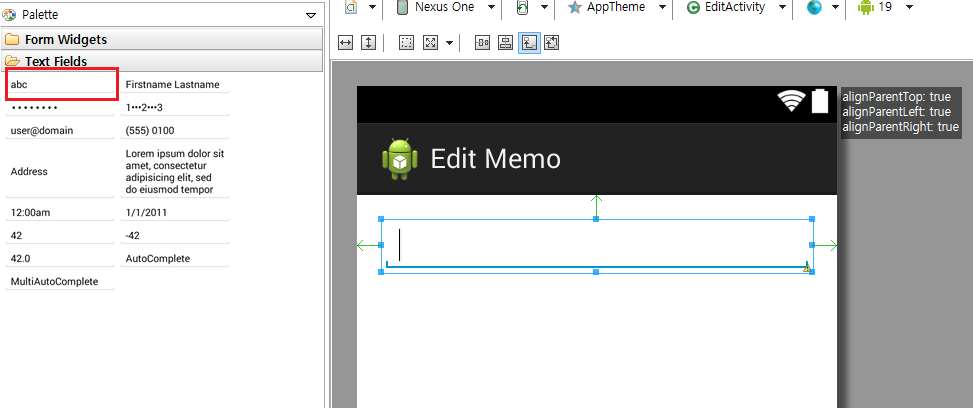
|  |  |  |
| --- | --- | --- |
| 입력 항목 | 입력 값 | 입력 항목 설명 |
| Project: | MyMemo | 현재 프로젝트명 |
| Activity Name: | EditActivity | 액티비티 클래스명 |
| Layout Name: | activity\_edit | 레이아웃 리소스 XML 파일명 |
| Title: | Edit Memo | 액티비티 제목줄에 표시될 문자열 |
| Launcher Activity: | 체크 안함 | 이 액티비티를 앱을 시작할 때 처음 실행되는 시작 액티비티로 설정할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

위와 같이 입력하고 Next 버튼 클릭

다음 화면에서 그냥 Finish 버튼을 클릭.

## activity\_edit.xml 수정

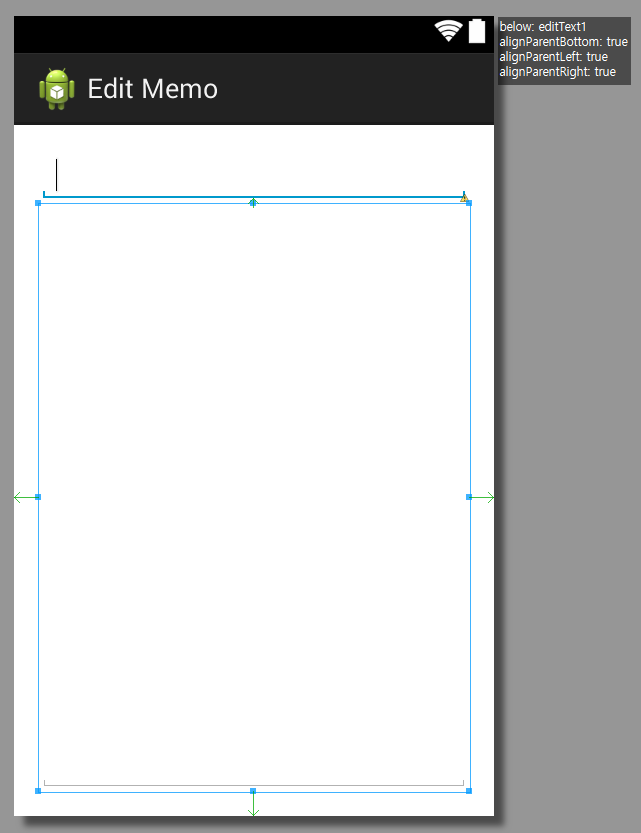
새로 추가된 EditActivity의 레이아웃 리소스 XML 파일인 activity\_edit.xml을 수정하자.



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 아래에 붙어서 배치된다. |
| alignParentLeft: | true | 부모의 왼쪽 경계에 붙어서 배치된다. |
| alignParentRight: | true | 부모의 오른쪽 경계에 붙어서 배치된다. |
| alignParentBottom: | true | 부모의 아래쪽 경계에 붙어서 배치된다. |
| Gravity | top | EditText의 내용 문자열이 수직 방향으로 상단에 붙어서 배치된다. |

단축키 Ctrl+S를 눌러서 저장하고 편집창을 닫자.

## 저장 메뉴 만들기

EditActivity 액티비티의 메뉴에 저장 메뉴를 추가하자.

### res/menu/edit.xml 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <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 |

# 편집 기능 추가

MainActivity에서 리스트뷰의 항목을 하나 클릭하면, EditActivity로 넘어가야 한다.

이때 그냥 넘어가는 것이 아니고, 클릭된 항목의 기본키(primary key)인 \_id 필드값을 EditActivity에 전달해야 한다.

## 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  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84 | package skhu.sw.mymemo;  import android.support.v4.widget.CursorAdapter;  import android.support.v4.widget.SimpleCursorAdapter;  import android.support.v7.app.ActionBarActivity;  import android.content.Intent;  import android.database.Cursor;  import android.database.sqlite.SQLiteDatabase;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.widget.AdapterView;  import android.widget.AdapterView.OnItemClickListener;  import android.widget.ListView;  public class MainActivity extends ActionBarActivity {    LectureDBOpenHelper helper;  SQLiteDatabase db;  Cursor cursor;  SimpleCursorAdapter adapter;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);    adapter = new SimpleCursorAdapter(this,  R.layout.list\_item,  null,  new String[] { "title", "modifiedTime", "\_id" },  new int[] { R.id.textView1, R.id.textView2, R.id.textView3 },  CursorAdapter.FLAG\_REGISTER\_CONTENT\_OBSERVER  );  ListView listView = (ListView)findViewById(R.id.listView1);  listView.setAdapter(adapter);  listView.setOnItemClickListener(new OnItemClickListener() {  @Override  public void onItemClick(AdapterView<?> parentView, View view, int position, long id) {  Intent intent = new Intent(MainActivity.this, EditActivity.class);  intent.putExtra("\_id", id);  startActivity(intent);  }  });  }  @Override  protected void onResume() {  super.onResume();  helper = new LectureDBOpenHelper(this);  db = helper.getWritableDatabase();  cursor = db.rawQuery("SELECT \* FROM MyMemo", null);  adapter.changeCursor(cursor);  }    @Override  protected void onPause() {  super.onPause();  cursor.close();  db.close();  helper.close();  }  @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\_new) {  Intent intent = new Intent(this, EditActivity.class);  startActivity(intent);  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄38~45) listView의 항목 하나를 클릭하면 즉시 호출되는 onItemClick 메소드를 구현한 OnItemClickListener 객체를 생성하여 listView에 등록하였다.

(줄40) 이제 리스트뷰의 항목 하나를 클릭하면, 줄40의 onItemClick 메소드가 즉시 호출된다.

이 메소드의 마지막 파라미터 long id 는 클릭한 항목에 표시되는 레코드의 기본키(primary key)인 \_id 필드값이다.

SQLiteDatabase에서 모든 테이블의 기본키는 long 타입의 \_id 필드이다.

(줄41) EditActivity 액티비티를 호출하기위한 Intent 객체를 생성한다.

(줄42) 이 Intenet 객체에 id 변수값을 데이터로 저장한다. 이 데이터의 이름은 "\_id" 이다.

이 데이터는 EditActivity에 전달된다.

Intent 객체를 사용하여 다른 액티비를 호출할 때, 그 호출되는 액티비티에 데이터를 넘겨주려면,

Intent 클래스의 putExtra("*데이터\_이름* ", *전달할\_데이터*); 메소드를 사용한다.

(줄43) EditActivity 객체를 호출한다.

데이터 항목을 클릭하여 EditActivity가 호출된 경우에는 \_id 필드값이 전달된다.

이때는 그 데이터 항목을 편집해서 데이터베이스에 저장(update)해야 한다.

그런데 R.id.action\_new 메뉴를 클릭하여 EditActivity가 호출된 경우에는 (줄76~78)

\_id 필드값이 전달되지 않는다.

이때는 새 데이터 항목을 입력 받아서 데이터베이스에 추가(insert)해야 한다.

## 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  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73 | package skhu.sw.mymemo;  import android.support.v4.app.NavUtils;  import android.support.v7.app.ActionBarActivity;  import android.content.ContentValues;  import android.content.Intent;  import android.database.Cursor;  import android.database.sqlite.SQLiteDatabase;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.widget.EditText;  public class EditActivity extends ActionBarActivity {    long \_id;  EditText editText1;  EditText editText2;    @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);  Intent intent = getIntent();  \_id = intent.getLongExtra("\_id", 0);  if (\_id > 0) {  LectureDBOpenHelper helper = new LectureDBOpenHelper(this);  SQLiteDatabase db = helper.getReadableDatabase();  Cursor cursor = db.rawQuery("SELECT title, body FROM MyMemo WHERE \_id = " + \_id, null);  if (cursor.moveToFirst()) {  editText1.setText(cursor.getString(0));  editText2.setText(cursor.getString(1));  }  cursor.close();  db.close();  helper.close();  }  }  @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) {  LectureDBOpenHelper helper = new LectureDBOpenHelper(this);  SQLiteDatabase db = helper.getWritableDatabase();  ContentValues value = new ContentValues(2);  value.put("title", editText1.getText().toString());  value.put("body", editText2.getText().toString());  if (\_id == 0)  db.insert("MyMemo", null, value);  else  db.update("MyMemo", value, "\_id = " + \_id, null);  db.close();  helper.close();  NavUtils.navigateUpFromSameTask(this);  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄27) MainActivity가 EditActivity를 호출할 때 사용한 Intent 객체를 꺼낸다.

(줄28) 그 Intent 객체에서 MainActivity가 전달한 데이터를 꺼낸다.

데이터의 이름 "\_id"가 정확히 일치해야 한다.

이 데이터는 클릭된 데이터 항목의 기본키(primary key)인 \_id 필드값이다.

꺼낸 데이터 값을 멤버 변수 \_id에 대입한다.

Intent 클래스의 getLongExtra("*데이터의 이름* ", *디폴트값* ) 메소드를 사용하여 전달된 데이터를 꺼낸다.

전달된 데이터가 없는 경우에 두번째 파라미터 디폴트 값이 리턴된다.

(줄30) 꺼낸 데이터 값이 디폴트 값인 0이 아니라면, 즉 전달된 데이터가 있다면 true가 되어 줄31~40을 실행한다.

(줄33) 기본키인 \_id 필드값이 멤버 변수 \_id에 대입된 값과 같은 레코드를 조회하는 SQL 문장을 실행한다.

"SELECT title, body FROM MyMemo WHERE \_id = " + \_id

여기서 + \_id 부분이 의미하는 것은 멤버 변수 \_id의 값을 문자열에 덧붙이라는 것이다.

예를 들어 멤버 변수 \_id의 값이 2라면,

위 코드의 결과는 ""SELECT title, body FROM MyMemo WHERE \_id = 2" 가 된가.

즉 \_id 필드값이 2인 레코드를 조회하는 SQL 문장이다.

SQLiteDatabase 클래스의 rawQuery 메소드는 파라미터로 주어진 SQL 문장을 실행하고 조회 결과를 Cursor 객체로 리턴한다.

(줄34) Cursor 클래스의 moveToFirst() 메소드는 첫 레코드를 받아온다. 그리고 받아온 레코드가 있으면 true를 리턴한다.

(줄35,36) Cursor 클래스의 getStirng(*인덱스* ) 메소드는 조회 결과로 받아온 현재 레코드의 *인덱스* 번째 필드 값을 문자열로 리턴한다.

(줄61~62) MainActivity로부터 전달받은 데이터를 대입해 놓은 \_id 멤버 변수 값이 0이라면,

MainActivity로부터 전달받은 데이터가 없다는 것을 의미한다.

이때의 저장은 새 데이터 추가(insert)이다.

(줄63~64) \_id 멤버 변수 값이 0보다 크다면, 이 값은 MainActivity에서 클릭된 데이터 항목의 기본키 \_id 필드값을 의미한다. 이때의 저장은 그 데이터 항목의 수정(update)이다.

(줄64) SQLiteDatabase 클래스의 update 메소드

update(*테이블명* , *저장할필드값* , *WHERE조건식* , *WHERE조건식파라미터* ) 메소드

테이블명 : 데이터를 저장할 테이블 이름

저장할필드값 : 저장할 데이터의 필드명과 필드값을 저장한 ContentValues 객체

WHERE조건식 : UPDATE할 레코드를 조회하는 SQL 문장의 WHERE 조건식 구문이다.

거의 대부분의 경우 기본키은 \_id 필드값을 비교하는 "\_id = 3" 형태의 문자열이 값이다.

WHERE조건식파라미터 : WHERE조건식에 물음표(?) 문자가 사용되었다면, 그 물음표 문자 대신 끼워 넣을

파라미터 값이다. String[] 타입의 값이어야 한다.

줄64의 WHERE조건식 파라미터에 물음표가 없으므로 이 파라미터는 null이다.

## 실행

실행해보고 기 입력한 메모 항목을 수정해보자

### 결과 MyMemo#2.zip

# 삭제 기능 추가

## 메뉴 아이콘 만들기

"05 Intent.docx" 파일의 16 페이지에서 설명한 것과 똑같은 방법으로

"Action Bar Icons/holo\_dark/01\_core\_remove" 아래의 ic\_action\_remove.png 파일을 프로젝트 리소스에 추가하자.

## 메뉴 제목 문자열 리소스 만들기

"05 Intent.docx" 파일의 17 페이지에서 설명한 것과 똑같은 방법으로 Remove 문자열 리소스를 만들자.

<string name="action\_remove">Remove</string>

## Remove 메뉴 만들기

EditActivity 액티비티의 메뉴에 새 메모 작성 메뉴를 추가하자.

### res/menu/edit.xml 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <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\_remove"  android:icon="@drawable/ic\_action\_remove"  android:title="@string/action\_remove"  app:showAsAction="ifRoom"/>  <item  android:id="@+id/action\_settings"  android:orderInCategory="100"  android:title="@string/action\_settings"  app:showAsAction="never"/>  </menu> |

(줄11~15)의 메뉴 항목이 추가되었다.

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

## 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  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | package skhu.sw.mymemo;  import android.support.v4.app.NavUtils;  import android.support.v7.app.ActionBarActivity;  import android.content.ContentValues;  import android.content.Intent;  import android.database.Cursor;  import android.database.sqlite.SQLiteDatabase;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.widget.EditText;  public class EditActivity extends ActionBarActivity {    long \_id;  EditText editText1;  EditText editText2;    @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);  Intent intent = getIntent();  \_id = intent.getLongExtra("\_id", 0);  if (\_id > 0) {  LectureDBOpenHelper helper = new LectureDBOpenHelper(this);  SQLiteDatabase db = helper.getReadableDatabase();  Cursor cursor = db.rawQuery("SELECT title, body FROM MyMemo WHERE \_id = " + \_id, null);  if (cursor.moveToFirst()) {  editText1.setText(cursor.getString(0));  editText2.setText(cursor.getString(1));  }  cursor.close();  db.close();  helper.close();  }  }  @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) {  LectureDBOpenHelper helper = new LectureDBOpenHelper(this);  SQLiteDatabase db = helper.getWritableDatabase();  ContentValues value = new ContentValues(2);  value.put("title", editText1.getText().toString());  value.put("body", editText2.getText().toString());  if (\_id == 0)  db.insert("MyMemo", null, value);  else  db.update("MyMemo", value, "\_id = " + \_id, null);  db.close();  helper.close();  NavUtils.navigateUpFromSameTask(this);  return true;  } else if (id == R.id.action\_remove) {  if (\_id > 0) {  LectureDBOpenHelper helper = new LectureDBOpenHelper(this);  SQLiteDatabase db = helper.getWritableDatabase();  db.execSQL("delete from MyMemo where \_id = " + \_id);  db.close();  helper.close();  }  NavUtils.navigateUpFromSameTask(this);  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄69~79) 삭제(Remove) 메뉴가 클릭되었을 때 실행될 코드이다.

(줄70) \_id 멤버 변수 값이 0이라는 것은 현재 새 메모를 입력 중임을 뜻한다.

입력 중인 새 메모를 데이터베이스에 추가(insert)하지 않고 줄77,78을 실행하여 MainActivity로 리턴하면

새 메모는 삭제되는 셈이다.

(줄71~75) MyMemo 테이블의 기본키인 \_id 필드값으로 조회해서 레코드를 삭제하는 DELETE SQL 문장을 실행한다.

## 실행

실행해 보고 기존의 메보를 삭제해 보자

### 결과

MyMemo#3.zip