SQLite Database와 ListView

2014-09-25

이승진

**목차**

[1. 예제 E03List #1 2](#_Toc399461803)

[1) 프로젝트 생성 2](#_Toc399461804)

[2. 레이아웃 리소스 만들기 2](#_Toc399461805)

[1) activity\_main.xml 수정 2](#_Toc399461806)

[2) strings.xml 수정 4](#_Toc399461807)

[3) list\_item.xml 생성 5](#_Toc399461808)

[3. Java 클래스 구현 10](#_Toc399461809)

[1) LectureDBOpenHelper.java 추가 10](#_Toc399461810)

[2) MainActivity.java 수정 13](#_Toc399461811)

**학습목표**

안드로이드의 SQLite 데이터베이스를 생성한다.

데이터베이스에 데이터를 삽입(insert) 한다.

데이터베이스에서 데이터를 조회한다.

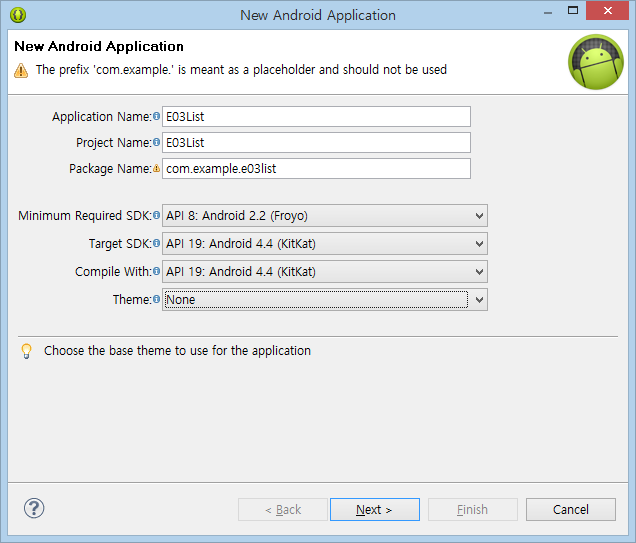
조회결과를 ListView에 채워서 화면에 표시한다.

# 예제 E03List #1

SQLite Database에 저장된 데이터 목록을 ListView로 출력해 보자.

## 프로젝트 생성

이클립스 메뉴: File – New – Andoid Applicaton Project



위와 같이 입력하고 Next를 계속 눌러서 진행하자.

# 레이아웃 리소스 만들기

## activity\_main.xml 수정

E02List 예제에서 만들었던 activity\_main.xml과 똑같이 만들어 주려고 한다.

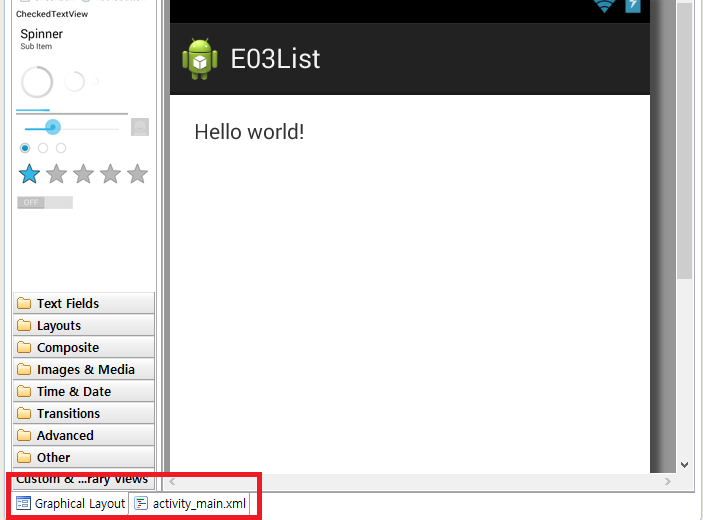
지금까지 eclipse의 시각적 편집기에서 layout xml 파일을 편집했었다.

이번에는 layout xml 파일을 텍스트 편집기에서 입력해 보자.

activity\_main.xml 파일의 편집창 아래를 보면 다음 그림과 같이 두 개의 탭이 있다.

Graphical Layout 탭은 시각적 편집기이고, 그 오른쪽의 activity\_main.xml 탭은 텍스트 편집기이다.

activity\_main.xml 탭을 클릭하자.



activity\_main.xml 파일의 텍스트 편집창에 아래 코드를 붙여 넣자.

|  |  |
| --- | --- |
| 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 | <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.e02list.MainActivity" >  <EditText  android:id="@+id/editText1"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentLeft="true"  android:layout\_alignParentTop="true"  android:layout\_toLeftOf="@+id/button1"  android:ems="10"  android:inputType="text" >  <requestFocus />  </EditText>  <Button  android:id="@+id/button1"  style="?android:attr/buttonStyleSmall"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignBottom="@+id/editText1"  android:layout\_alignParentRight="true"  android:text="@string/ok" />  <ListView  android:id="@+id/listView1"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_alignParentBottom="true"  android:layout\_alignParentLeft="true"  android:layout\_alignParentRight="true"  android:layout\_below="@+id/editText1" >  </ListView>  </RelativeLayout> |

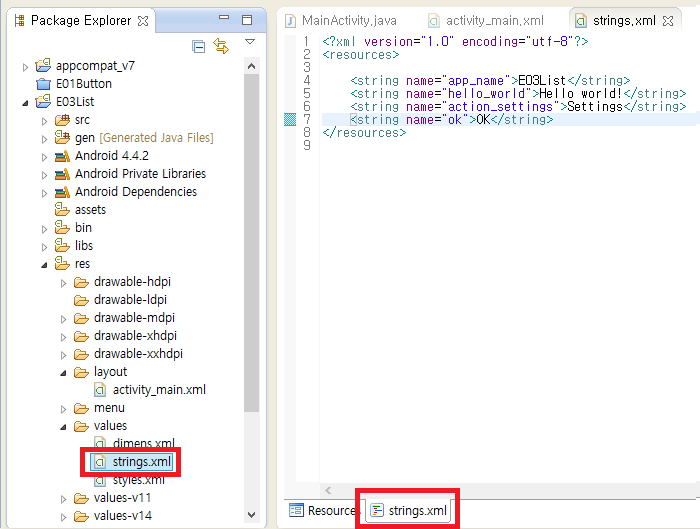
위 코드는 이전 E03List 예제의 activity\_main.xml 파일을 내용을 그대로 가져온 것이다.

(줄31)의 @string/ok에서 아마도 빨간색 에러 표시가 보일 것이다.

문자열 리소스(string resource) ok를 사용하고 있는데, 현재 프로젝트에는 이 문자열 리소스 ok를 아직 만들어 주지 않아서 없다는 에러이다.

## strings.xml 수정

문자열 리소스 ok를 만들자.



화면 왼쪽 Package Explorer 창에서 E03List - res - values 폴더 아래의 strings.xml가 문자열 리소르 파일이다.

더블 클릭하여 이 파일을 열고 편집창 아래의 strings.xml 탭을 클릭하면 위와 같이 텍스트 편집창이 열린다.

strings.xml 파일에 다음 항목을 추가하자.

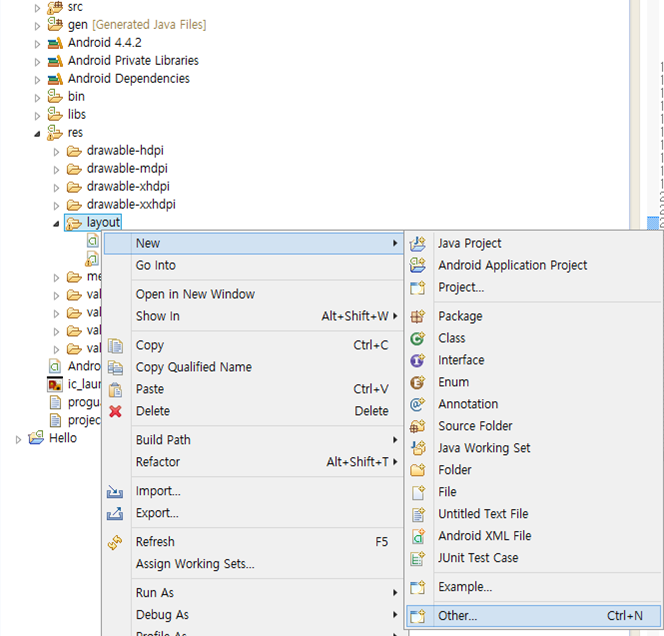
<string name="ok">OK</string>

Ctrl+S 키를 눌러서 파일을 저장하고 닫는다.

## list\_item.xml 생성

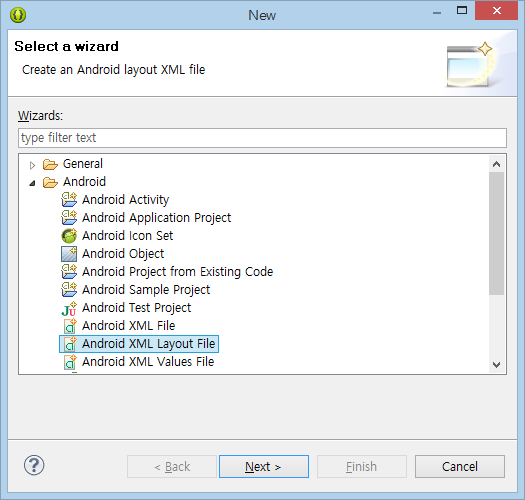
list\_item.xml 레이아웃 리소스 파일을 생성하자.

ListView의 데이터 항목 하나의 화면 배치(layout)를 지정하기 위한 레이아웃 리소스 파일이다.

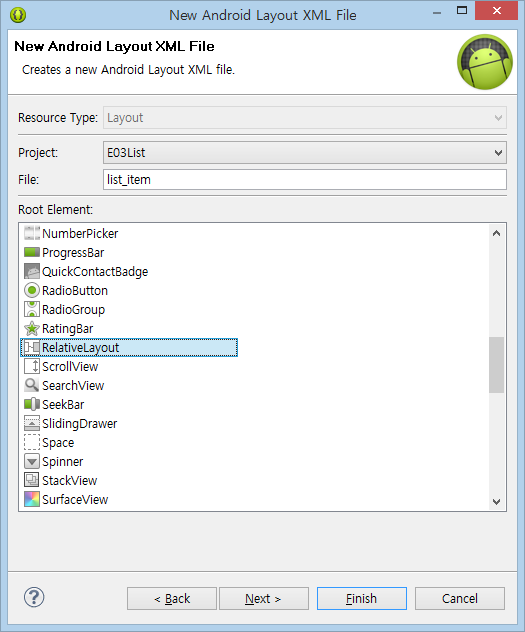


res 폴더 아래 layout 폴더를 우클릭하고

메뉴: New - Other...

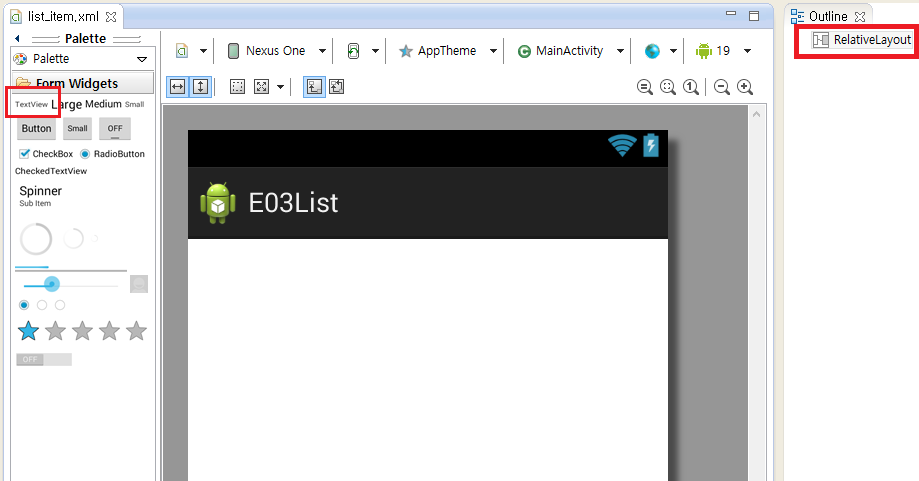


Android XML Layout File 항목 선택하고 Next 버튼 클릭



File: list\_item

RelativeLayout 선택하고 Finish 버튼 클릭



위와 같이 화면 배치(layout)을 만들기 위한 편집창이 열린다.

지금 여기서 우리가 편집해야 하는 레이아웃은 액티비티 화면 전체의 레이아웃이 아니고

ListView의 항목 하나를 화면에 표시하기 위한 레이아웃이다.

화면 오른쪽 Outline 창에 표시된 RelativeLayout에 주목하자.

Outline창은 현재 편집창에 배치된 뷰 객체들의 목록을 보여준다.

우리 눈에는 보이지 않지만, 지금 편집창에 RelativeLayout 객체가 하나 들어있다.

우리가 추가할 뷰 객체들은 이 RelativeLayout 객체의 자식 객체가 된다.

화면 왼쪽의 Palette 창의 TextView를 드래그 드롭하여 편집창에 세 번 추가하자.



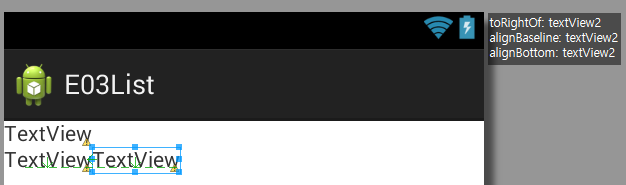
첫째 TextView는 위 그림과 같이 alignParentTop: true, alignParentLeft: true 가 되도록 배치해야 한다.

부모인 RelativeLayout의 위와 왼쪽에 붙어서 배치하라는 설정이다.

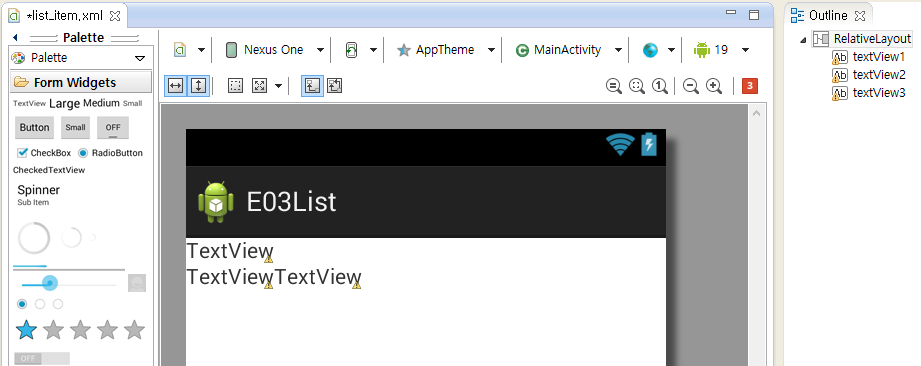


둘째 TextView는 위 그림과 같이 below: textView1, alignParentLeft: true 가 되도록 배치해야 한다.

textView1의 아래에, 그리고 부모의 왼쪽에 붙어서 배치하라는 설정이다.



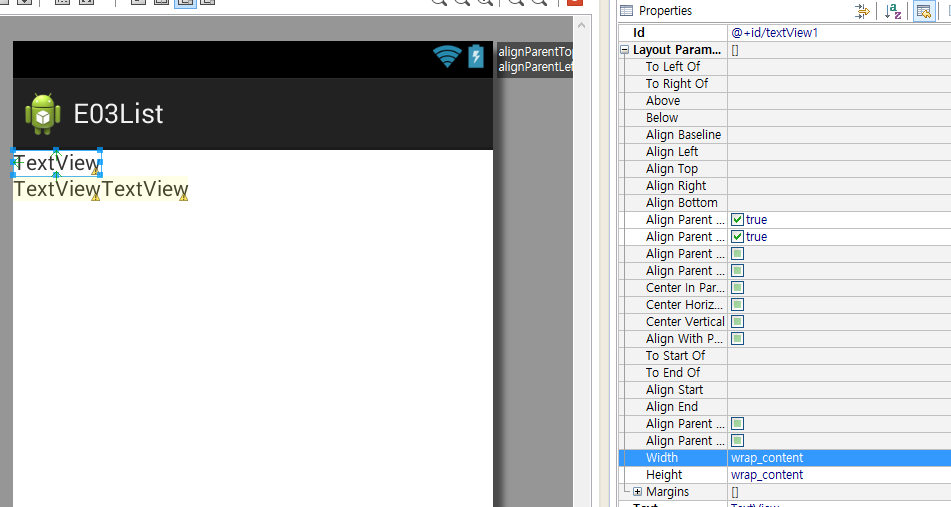
셋째 TextView는 toRightOf: textView2, alignBasline: textView2, alignBottom: textView2 가 되도록 배치해야 한다. textView2의 오른쪽에 붙어서, 그리고 textView와 아랫선을 맞춰서 배치하라는 설정이다.



화면 오른쪽의 Outline 창에 주목하자.

RelativeLayout 객체가 루트(root)이고, 방금 우리가 추가한 TextView 객체들이 RelativeLayout 객체의 자식으로 아래에 추가된 것을 확인할 수 있다. 그리고 이들의 id가 각각 textView1, textView2, textView3임을 확인할 수 있다.

textView1을 선택해 보자.



textView1을 선택하고 오른쪽 Properties창을 보면, 선택된 textView1의 속성값을 볼 수 있다.

속성창의 속성 목록에서 Layout Parameters 항목 아래를 펼쳐보면 width 속성값이 warp\_context이다.

이 속성값의 의미는 TextView의 폭(width)이 내부 문자열(content)의 크기와 같다는 것.

왼쪽 편집창을 봐도, TextView의 폭이 문자열의 크기와 같음을 알 수 있다.

이 width 속성값을 match\_parent로 바꾸자.

그리고 Text Size 속성값을 20sp로 입력하자.

textView1 속성 편집

|  |  |
| --- | --- |
| 속성 | 값 |
| Wdth | match parent |
| Text Size | 20sp |

textView2속성 편집

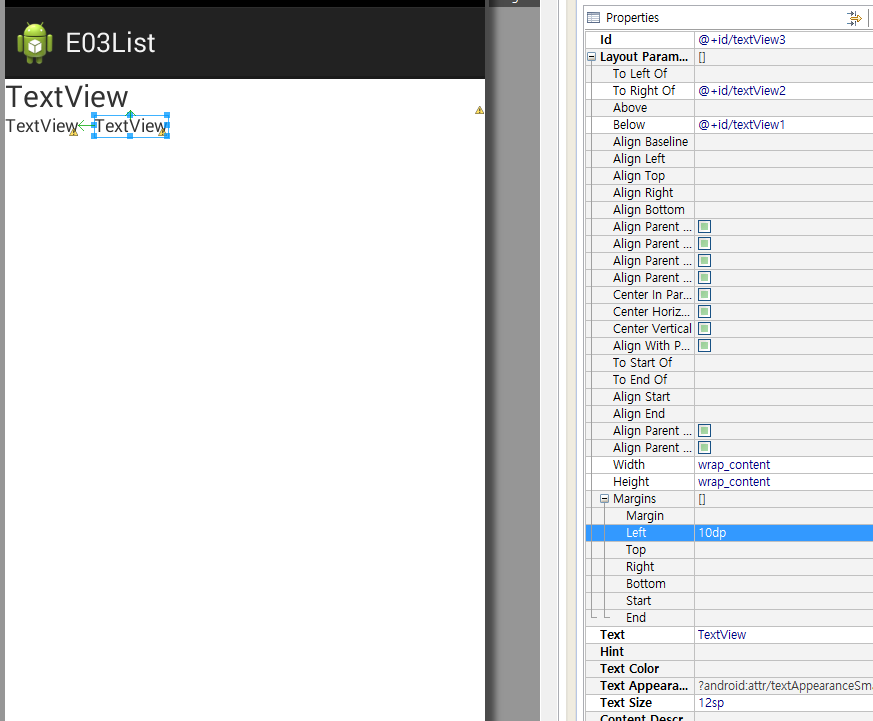
|  |  |
| --- | --- |
| 속성 | 값 |
| Text Size | 12sp |

textView3속성 편집

|  |  |
| --- | --- |
| 속성 | 값 |
| Text Size | 12sp |
| Margin Left | 10dp |

textView3의 Margin Left는 왼쪽 여백 크기를 지정하는 속성이다.

textView3를 선택하고 오른쪽 Properties창에서 Layout Parameters 항목 아래를 펼쳐보면 Margins 항목이 보인다. Margins 항목 아래를 펼쳐보면 Left 속성을 발견할 수 있을 것이다.

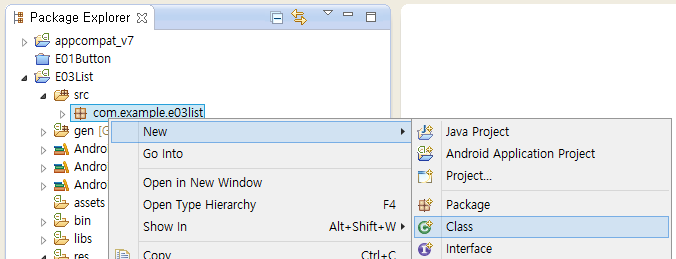


편집한 내용을 저장하고 편집창을 닫자.

# Java 클래스 구현

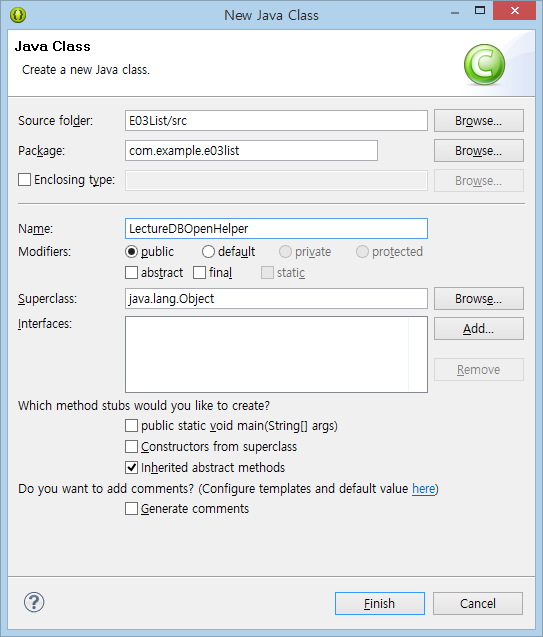
## LectureDBOpenHelper.java 추가

LectureDBOpenHelper 클래스를 추가하자.



위 그림과 같이 src 폴더 아래 com.example.e03list 패키지를 우클릭하고 New – Class 메뉴를 클릭하면

다음과 같은 대화상자가 표시된다.



Name: LectureDBOpenHelper

Finish 버튼을 클릭하면 src – com.example.e03list 폴더 아래에 LectureDBOpenHelper.java 파일이 생성된다.

다음과 같이 입력하고 저장하자.

|  |  |
| --- | --- |
| 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 | package com.example.e03list;  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 = 1;  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')) );";  public LectureDBOpenHelper(Context context) {  super(context, DB\_NAME, null, DB\_VERSION);  }  @Override  public void onCreate(SQLiteDatabase db) {  db.execSQL(SQL\_CREATE\_LISTITEM\_TABLE);  }  @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  }  } |

안드로이드는 SQLite 데이터베이스 엔진이 내장되어 있다.

앱에서 데이터베이스를 사용하려면, 먼저 데이터베이스 데이터 파일을 만들어야 한다.

데이터베이스 데이터 파일을 만들려면, (줄7)처럼 SQLiteOpenHelper 클래스를 상속받아서 LectureDBOpenHelper 클래스를 구현해야 한다.

지금 만들고 있는 E03List 앱은 파일명 Lecture.db인 데이터베이스 데이터 파일을 생성하고 여기에 데이터를 저장할 것이다.

(줄9) 데이터베이스 파일의 버전 번호를 정의한 int형 상수이다.

java에서 상수(constant)는 static final 멤버 변수로 정의한다.

(줄10)은 데이터베이스 파일명을 정의한 String형 상수이다.

(줄11)은 데이터베이스 파일 내부에 데이터베이스 테이블(table)을 생성하기 위한 SQL 문장을 정의한 String형 상수이다.

(줄15) LectureDBOpenHelper 클래스의 생성자이다. 생성자 파라미터는 Context 이다.

Context는 Activity 클래스의 부모 클래스이다.

(줄16) LectureDBOpenHelper 클래스의 부모인 SQLiteOpenHelper 클래스의 생성자를 호출한다.

생성자 파라미터로 Context와, 데이터베이스 파일의 버전과, 데이터베이스의 파일명을 전달해야 한다.

(줄20) SQLiteOpenHelper 클래스의 onCreate 메소드를 재정의한다.

이 메소드의 이름의 on으로 시작하므로 이벤트 핸들러임을 짐작할 수 있다.

데이터베이스 파일이 처음으로 생성(create)될 때 호출되는 메소드이다.

E03List 앱이 처음으로 실행될 때, SQLiteOpenHelper 클래스에 의해서 Lecture.db 데이터베이스 파일이 자동으로 생성된다. 이때 onCreate 메소드가 호출된다.

생성된 Lecture.db 데이터베이스에 해당하는 객체가 onCreate 메소드의 파라미터로 전달된다.

안드로이드에서 데이터베이스 파일에 해당하는 클래스는 SQLiteDatabase 이다.

나중에 E03List 앱을 삭제(uninstall)하면, Lecture.db 데이터베이스 파일도 같이 자동으로 삭제된다.

(줄21) onCreate 메소드에서 할 일은 데이터베이스 테이블을 생성하는 것이다.

SQLiteDatabase 클래스의 execSQL 메소드는 SQL 명령을 SQLite 엔진에서 실행하기 위한 메소드이다.

이 메소드의 파라미터는 SQL 문장에 해당하는 문자열이다.

줄11에서 문자열 상수로 정의된 SQL 문장을 줄21의 execSQL 메소드의 파라미터로 전달하고 있다.

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.

## 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  85  86  87  88 | package com.example.e03list;  import android.support.v4.widget.CursorAdapter;  import android.support.v4.widget.SimpleCursorAdapter;  import android.support.v7.app.ActionBarActivity;  import android.content.ContentValues;  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.view.View.OnClickListener;  import android.widget.Button;  import android.widget.EditText;  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);    Button b = (Button)findViewById(R.id.button1);  b.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View arg0) {  EditText e = (EditText)findViewById(R.id.editText1);  CharSequence s = e.getText();  e.setText("");  ContentValues value = new ContentValues(1);  value.put("title", s.toString());  db.insert("ListItem", null, value);  cursor.close();  cursor = db.rawQuery("SELECT \* FROM ListItem", null);  adapter.changeCursor(cursor);  }  });  }    @Override  protected void onResume() {  super.onResume();  helper = new LectureDBOpenHelper(this);  db = helper.getWritableDatabase();  cursor = db.rawQuery("SELECT \* FROM ListItem", 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;  }  return super.onOptionsItemSelected(item);  }  } |

(줄20) Lectude.db 데이터베이스 파일을 생성하는 코드를 구현한 LectureDBOpenHelper 클래스를 멤버 변수로 선언하였다.

(줄21) 데이터베이스 파일에 해당하는 SQLiteDatabase 클래스를 멤버 변수로 선언하였다.

(줄22) SQLite 데이터베이스에서 SELECT SQL 문장으로 조회한 결과 데이터에 해당하는 Cursor 클래스를 멤버 변수로 선언하였다.

(줄23) SQLite 데이터베이스에서 SELECT SQL 문장으로 조회한 결과 데이터에 해당하는 Cursor 클래스 객체를 ListView에 채워서 화면에 보여주기 위해 필요한 SimpleCursorAdapter 클래스를 멤버 변수로 선언하였다.

(줄26) 액티비티가 생성되자 마자 onCreate 이벤트 핸들러 메소드가 저절로 호출된다.

(줄27) 부모 클래스인 ActionBarActivity 클래스의 onCreate 메소드를 호출한다.

(줄28) R.layout.activity\_main 레이아웃 리소스의 내용대로 액티비티 화면을 만들라는 메소드 호출이다.

이 파일의 내용대로 EditText, Button, ListView 객체가 자동으로 만들어진다.

(줄30) SQLite 데이터베이스에서 SELECT SQL 문장으로 조회한 결과 데이터에 해당하는 Cursor 클래스 객체를 ListView에 채워서 화면에 보여주기 위해 필요한 SimpleCursorAdapter 클래스 객체를 생성하여 그 객체에 대한 참조를 멤버 변수 adapter에 대입한다.

SimpleCursorAdapter 클래스 생성자의 첫째 파라미터값으로 액티비티에 대한 참조를 전달한다.

(줄31) SimpleCursorAdapter 클래스 생성자의 2번째 파라미터는, ListView에서 데이터 항목 하나를 화면에 표시하기 위한 뷰(View) 객체들의 배치(layout)을 지정한 레이아웃 리소스(layout resource) XML 파일에 대한 id이다.

우리가 앞에서 만든 res/layout/list\_item.xml 파일의 id가 R.layout.list\_item 이다.

(줄32) SimpleCursorAdapter 클래스 생성자의 3번째 파라미터는, SQLite 데이터베이스에서 SELECT SQL 문장으로 조회한 데이터 목록에 해당하는 Cursor 클래스 객체이다. 아직 조회 결과 데이터가 없으므로 그냥 null을 전달한다.

(줄33) SimpleCursorAdapter 클래스 생성자의 4번째 파라미터는, SELECT SQL 문장으로 조회한 데이터의 필드명 목록이다. Lecture.db 데이터베이스의 ListItem 테이블에 \_id, title, modifiedTime 필드가 있다.

String 배열 객체를 생성하며 동시에 배열의 값도 지정하는 이 Java 문법에 주목하자.

(줄34) SimpleCursorAdapter 클래스 생성자의 5번째 파라미터는, 줄33에 명시한 title, modifiedTime, \_id 필드의 값이 각각 textView1, textView2, textView3에 자동으로 채워지도록 지정하기 위한, 뷰(View) 객체의 id 목록이다.

안드로이이드에서 뷰 객체의 id, 리소스의 id 값의 타입은 int 이다.

(줄35) SimpleCursorAdapter 클래스 생성자의 6번째 파라미터미로 그냥 이 값을 전달한다고 기억해 두자.

(줄30~36) SQLite 데이터베이스에서 SELECT SQL 문장으로 조회한 결과 목록의 데이터 각각에 대해서, R.layout.list\_item 레이아웃 리소스 파일에서 정의한 대로 textView1, textView2, textView3 뷰 객체를 생성하고,

이 뷰 객체들 조회 결과 데이터의 title, modifiedTime, \_id 필드값을 각각 채워서,

ListView에 전달해 주는 SimpleCursorAdapter 객체를 생성하였다.

그런데 아직 조회 결과 데이터는 넘겨주지 않고, 그냥 null을 전달하였다.

(줄37) id가 R.id.listView1인 ListView객체에 대한 참조를 얻는 메소드 호출이다.

이 ListView 객체는 줄28의 메소드 호출에 의해서 생성되었다.

(줄38) ListView 객체에 SimpleCursorAdapter 객체를 연결한다.

(줄40) id가 R.id.button1인 Button객체에 대한 참조를 얻는 메소드 호출이다.

이 Button 객체는 줄28의 메소드 호출에 의해서 생성되었다.

(줄41~54) 온클릭리스너(OnClickListener) 객체를 하나 생성하여, 버튼 객체에 연결한다.

이제 버튼이 클릭되면 연결된 온클릭리스너 객체의 onClick 메소드가 즉시 호출될 것이다.

여기서는 온클릭리스너를 버튼에 연결만 할 뿐이지, 아직 onClick 메소드는 호출되지 않는다.

(줄58) 액티비티가 화면에 표시되기 직전에 onResume 이벤트 핸들러 메소드가 자동으로 호출된다.

액티비티가 처음 실행될 때뿐만이 아니라, 액티비티가 화면이 내려갔다가 다시 보일 때도 언제나 onResume 메소드가 호출된다. onCreate 메소드가 onResume 메소드보다 먼저 호출된다.

(줄59) 부모 클래스의 onResume 메소드를 호출한다.

(줄60) Lectude.db 데이터베이스 파일을 생성하는 코드를 구현한 LectureDBOpenHelper 클래스 객체를 하나 생성하여, 그 객체에 대한 참조를 멤버 변수 helper에 대입한다.

(줄61) Lecture.db 데이터베이스 파일을 읽고 쓰기 모드로 열 때, SQLiteOpenHelper 클래스의 getWritableDatabase 메소드를 호출한다.

SQLiteOpenHelper 클래스는 LectureDBOpenHelper 클래스의 부모 클래스이다. 읽기 모드로 열 때는 getReadableDatabase 메소드를 호출한다.

이 메소드의 호출결과 Lecture.db 데이터베이스를 사용하기 위한 SQLiteDatabase 객체가 리턴된다.

이 객체에 대한 참조를 멤버 변수 db에 대입한다.

(줄62) 데이터베이스에서 조회하기 위한 SELECT SQL 문장을 실행하기 위해, SQLiteDatabase 클래스의 rawQuery 메소드를 호출한다. 조회 결과 데이터는 Cursor 클래스 객체로 리턴된다.

(줄63) 조회 결과 데이터인 커서 객체를 SimpleCursorAdapter에 전달하기 위해, SimpleCursorAdapter 클래스의 changeCusor 메소드를 호출한다.

이제 조회 결과 데이터가 SimpleCursorAdapter에 의해서 뷰(view) 객체에 채워져 ListView에 전달되어 화면에 보이게 된다.

사용자가 화면의 EditText에 문자열을 입력하고 버튼을 누르면 (줄43)의 onClick 메소드가 호출된다.

(줄43~53) 버튼이 클릭되면 호출되는 메소드이다.

(줄44) id가 R.id.editText1인 EditText 객체에 대한 참조를 얻는 메소드 호출이다.

이 EditText 객체는 줄28의 메소드 호출에 의해서 생성되었다.

(줄45) EditText 객체에 입력된 문자열을 얻는다.

(줄46) EditText에 입력된 문자열을 빈문자열로 교체한다. 즉 문자열 입력 텍스트 박스의 내용이 지워진다.

(줄47) SQLite 데이터베이스에 데이터를 추가(insert)하기 위한 필드값을 전달할 때 사용하는 ContentValues 클래스 객체를 하나 생성한다. 생성자 파라미터는 여기에 채울 필드값의 갯수이다.

Lecture.db 데이터베이스의 ListItem 테이블에서 \_id, modifiedTime 필드값은 자동으로 부여되므로, title 값만 전달하면 된다.

(줄48) EditText에 입력되었던 문자열을 title 필드값으로 전달하기 위해 ContextValues 객체에 채운다.

(줄49) SQLite 데이터베이스에 데이터를 추가(insert)하기 위한 메소드 호출이다.

여기서 db는 MainActivity 클래스의 멤버 변수이다.

화면에 액티비티가 표시되기 직전에 호출되는 onResume 메소드가 먼저 실행되었을 것이고, 이 메소드의 줄61에서 멤버 변수 db에 SQLiteDatabase 객체에 대한 참조가 대입되어 있을 것이다.

insert 메소드의 첫째 파라미터는 테이블명이고, 둘째 파라미터는 삽입(insert)할 필드값이 들어있는 ContentValues 객체이다.

데이터베이스 파일에 새 데이터가 추가되어도, 이전에 SELECT SQL 문장으로 조회한 데이터에 이 새 데이터가 저절로 추가되지 않는다. 새로 조회해 와야 한다.

(줄50) 아까 조회한 조회 결과 데이터가 이제 필요 없으므로, 새로 조회하기 전에 먼저 조회결과 데이터에 해당하는 Cursor 클래스의 close 메소드를 호출해야 한다.

데이터베이스 관련 객체들의 close 메소드는 반드시 호출해 주는 것이 바람직하다.

(줄51) 데이터베이스에서 다시 데이터를 조회한다.

(줄52) 조회 결과 데이터인 Cursor 객체를 SimpleCursorAdapter 객체에 채운다.

이제 새로 삽입된 데이터도 화면에 표시될 것이다.

(줄67) 액티비티가 종료될 때나 다른 액티비티 화면이 위로 올라올 때 등 여러 이유로 현재 액티비티의 화면이 아래로 내려가야 한다면, 언제나 onPause 메소드가 호출된다.

(줄68) 부모 클래스의 onPause 메소드를 호출한다.

(줄69~71) 데이터베이스 관련 객체들의 close 메소드를 호출한다.

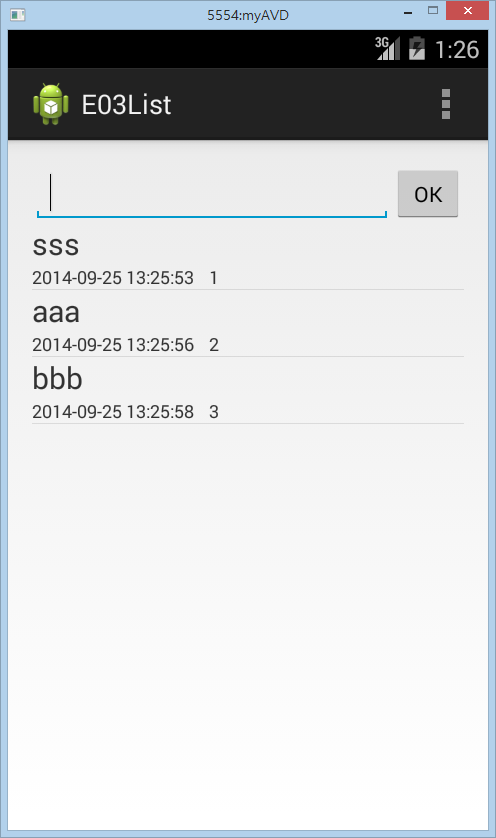
객체가 생성된 순서의 역순으로 close 메소드를 호출한다.

액티비티가 화면에 표시되기 직전에 줄60~63의 코드가 실행되어, 데이터베이스 조회 결과가 화면에 표시되고,

액티비티가 화면에서 내려가기 직전에 조회결과 데이터가 제거되고 데이터베이스도 닫힌다.

이렇게 구현하는 것이 여러가지 이유로 바람직하다.

# 실행결과



### 소스코드

E03List\_#1.zip