Toast와 ListView의 클릭 이벤트

2014-09-25

이승진

**목차**

[1. 런처 아이콘 변경 2](#_Toc399771088)

[1) 런쳐 아이콘 (laucher icon) 2](#_Toc399771089)

[2) 해상도 (resolution) 2](#_Toc399771090)

[3) 드로워블 리소스(drawable resource) 2](#_Toc399771091)

[4) 런쳐 아이콘 크기 3](#_Toc399771092)

[5) 런처 아이콘 변경하기 3](#_Toc399771093)

[2. ListView의 클릭 이벤트 4](#_Toc399771094)

[1) OnItemClick 이벤트 4](#_Toc399771095)

[2) OnItemLonClick 이벤트 4](#_Toc399771096)

[3. Toast 클래스 4](#_Toc399771097)

[4. MainActivity.java 수정 5](#_Toc399771098)

[5. ActionBar 제목 변경 9](#_Toc399771099)

[1) 문자열 리소스 만들기 9](#_Toc399771100)

[2) AndroidManifest.xml 파일 수정 10](#_Toc399771101)

[6. 실행 11](#_Toc399771102)

[7. 과제 11](#_Toc399771103)

**학습목표**

앱의 런쳐 아이콘을 변경한다.

ActionBar의 제목을 변경한다.

ListView의 ItemClick, ItemLongClick 이벤트 핸들러를 구현한다.

Toast 클래스를 사용하여 화면에 문자열을 잠시 보여준다.

# 런처 아이콘 변경

## 런쳐 아이콘 (laucher icon)

안드로이드 앱의 아이콘을 런쳐 아이콘(launcher icon)이라고 한다.

앱 아이콘 목록을 보여주고, 아이콘을 터치하면 앱을 실행시켜주는 SW를 앱 런처(app launcher)라고 부른다.

이 런처에 표시되기 위한 아이콘이라는 의미로 런처 아이콘이라고 한다.

## 해상도 (resolution)

화면에서 색상을 표시하는 가장 작은 단위인 점을 우리말로 화소라고 하고, 영어로 픽셀(pixel) 이라고 한다.

이 픽셀의 크기가 작고 세밀할 수록 고해상도이고, 픽셀의 크기가 크고 거칠수록 저해상도이다.

해상도의 단위는 DPI (dot per inch) 이다. 1인치에 들어있는 점의 수를 의미한다.

고해상도일 수록 점의 크기가 작으므로, 1인치에 저행상도보다 더 많은 점이 들어있다.

예를들어 100 dpi인 화면에서 3인치 크기에는 300 픽셀이 들어있고, 300 dpi 화면에서 3인치에는 900 픽셀이 들어있다. 즉 저해상도 화면과 고해상도 화면에서 동일한 크기로 보이려면, 고해상도에서는 더 많은 픽셀로 그려야 한다.

## 드로워블 리소스(drawable resource)

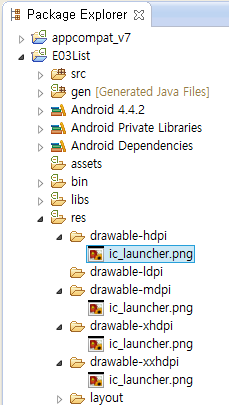
화면에 그려질 수 있는 데이터 파일을 드로워블 리소스(drawable resource)라고 한다.

대표적인 드로워블 리소스는 아이콘 파일, 비트맵 파일이다.

다양한 해상도의 안드로이드 기기에서 동일한 크기로 아이콘이 그려지려면, 고해상도의 화면에는 큰 아이콘을, 저해상도의 화면에는 작은 아이콘을 그려야 한다.

드로워블 리소스 파일을 저장하기 위한 폴더가 해상도별로 여러개이다.

|  |  |
| --- | --- |
| res/drawable-ldpi | low dpi 저해상도 |
| res/drawable-mdpi | medium dpi 중간 해상도 |
| res/drawable-hdip | high dpi 고해상도 |
| res/drawable-xhdpi | extra high dpi 초고해상도 |
| res/drawable-xxhdpi | extra extra high dpi 초초고해상도 |



## 런쳐 아이콘 크기

|  |  |
| --- | --- |
| res/drawable-ldpi | 36 x 36 픽셀 |
| res/drawable-mdpi | 48 x 48 픽셀 |
| res/drawable-hdip | 72 x 72 픽셀 |
| res/drawable-xhdpi | 96 x 96 픽셀 |
| res/drawable-xxhdpi | 144 x 144 픽셀 |

런쳐 아이콘을 해상도별로 따로 정교하게 만들려면, 위 표의 크기로 여러개 만들어서 해상도별로 해당 드로워블 리소스 폴더에 넣어야 한다. 아이콘을 해상도별로 따로 만들지 않고, 그냥 크기만 변경해서 저장할 것이라면, 해상도 별로 따로 만들지 않아도 된다. 144 x 144 픽셀 크기의 아이콘 파일 하나만 res/drawable-xxhdpi 폴더에 넣어 놓으면, 다른 크기의 아이콘이 필요할 경우에 안드로이드 시스템이 알아서 크기 조정해서 사용한다.

## 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. png: 색상의 수가 몇개 안될때(압축률이 좋다) jpg: 색상의 수가 많다(사진등)



위 아이콘의 크기가 144 x 144 픽셀이다.

이 아이콘 파일의 URL은 다음과 같다.

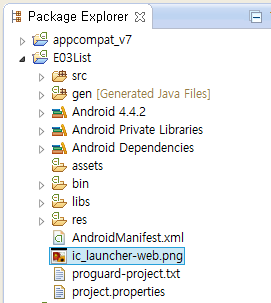
<https://onedrive.live.com/embed?cid=C704E2ECDA408483&resid=C704E2ECDA408483%219274&authkey=ACThB7-AmXjQ82U>

E03List 예제 프로젝트의 런처 아이콘을 이 파일로 교체하자.

런처 아이콘의 배경은 흰색이 아니라 투명해야 한다.

## Google Play용 런처 아이콘

앱을 Google Play에 등록할 때 512 x 512 크기의 런처 아이콘 파일이 필요하다.



프로젝트에 ic\_launcher-web.png 파일이 바로 512 x 512 크기의 런처 아이콘 파일이다.

웹사이트용 아이콘이다. 앱에선 안쓰인다.

이 파일은 안드로이드 앱에서 사용되지 않는다. 따라서 안드로이드 앱을 빌드하여 생성되는

설치파일(\*.apk)에 포함되지도 않는다.

프로젝트에 포함된 ic\_launcher-web.png 파일의 역할은

앱을 Google Play에 등록할 때 이 크기의 런처 아이콘 파일이 필요하다고 알려주는 것 뿐이다.

따라서 그냥 삭제해 버려도 무방하다.

# ListView의 클릭 이벤트

## ItemClick 이벤트

ListView의 항목을 클릭하면 ItemClick 이벤트가 발생한다.

ListView의 항목이 클릭될 때 실행될 코드를 구현하는 방법은 다음과 같다.

- OnItemClickListener의 자식 클래스를 만들고

- 그 자식 클래스에서 onItemClick 메소드를 재정의 한다.

- 그 자식 클래스 객체를 생성(new)하여서

- ListView의 setOnItemClickListener 메소드를 호출하여 그 자식 클래스 객체를 ListView에 등록한다.

## ItemLongClick 이벤트

ListView의 항목을 길게 꾸욱 눌러 클릭하면 ItemLongClick 이벤트가 발생한다.

ListView의 항목이 길게 클릭될 때 실행될 코드를 구현하는 방법은 다음과 같다.

- OnItemLongClick의 자식 클래스를 만들고

- 그 자식 클래스에서 onItemLongClick 메소드를 재정의 한다.

- 그 자식 클래스 객체를 생성(new)하여서

- ListView의 setOnItemLongClickListener 메소드를 호출하여 그 자식 클래스 객체를 ListView에 등록한다.

# Toast 클래스

Toast 클래스는 화면에 문자열을 잠깐 표시할 때 사용한다.

Toast 클래스의 makeText 메소드를 호출하여 Toast 객체를 생성한다.

public static Toast makeText(Context context, int resid, int duration); //아이디를 넘겨줌

public static Toast makeText(Context context, CharSequence text, int duration);//문자열을 넘겨줌

new는 오퍼레이터(단항연산자)

a= new SomeClass().amethod();->a= (new SomeClass()).amethod();

new SomeClass().amethod();

연산자 new , . , ()<메소드 호출 연산자>

\*p++ 이면 ++가 높다. 둘다 단항연산자면, 오른쪽부터한다.

\*나 %연산자는 같으면 왼쪽부터한다.

위 네모안 메소드는 팩토리 메소드(factory method)

makeText 메소드는 public static 메소드이다. 즉 클래스 메소드이다.

이 메소드는 Toast 객체를 생성하여 리턴한다.

첫째 파라미터 값으로 보통 액티비티에 대한 참조를 전달한다. Context는 Activity의 부모 클래스이다.

둘째 파라미터 값으로 화면에 표시할 문자열을 전달한다.

이 값의 타입이 int이면 문자열 리스소의 아이디이고, 이 값의 타입이 CharSequence이면 문자열 객체이다.

CharSequence는 Java 언어의 모든 문자열 클래스들의 부모 클래스이다.

셋째 파라미터는 화면에 표시할 시간의 길이다.

Toast.LENGTH\_SHORT, Toast.LENGTH\_LONG 두 값중의 하나를 전달한다.

LENGTH\_SHORT이면 문자열이 아주 잠깐 표시되다 저절로 사라지고, LENGTH\_LONG이면 몇초 정도 표시되다 저절로 사라진다. public static final int!

# MainActivity.java 수정

### src/com.example.e03list/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  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  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118 | package com.example.e03list;  import java.util.Locale;  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.AdapterView;  import android.widget.AdapterView.OnItemClickListener;  import android.widget.AdapterView.OnItemLongClickListener;  import android.widget.Button;  import android.widget.EditText;  import android.widget.ListView;  import android.widget.TextView;  import android.widget.Toast;  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) {  TextView textView1 = (TextView)view.findViewById(R.id.textView1);  TextView textView2 = (TextView)view.findViewById(R.id.textView2);  TextView textView3 = (TextView)view.findViewById(R.id.textView3);  String s = String.format(Locale.KOREA,  "title:%s modifiedTime:%s \_id:%s position:%d id:%d",  textView1.getText(), textView2.getText(), textView3.getText(), position, id);  Toast toast = Toast.makeText(MainActivity.this, s, Toast.LENGTH\_LONG);  toast.show();  }  });  listView.setOnItemLongClickListener(new OnItemLongClickListener() {  @Override  public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {  db.execSQL("delete from ListItem where \_id = " + id);  cursor.close();  cursor = db.rawQuery("SELECT \* FROM ListItem", null);  adapter.changeCursor(cursor);  return true;  }  });  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);  }  } |

1. 자바문법으로 어떤 의미를 갖는가? 2. 안드로이드 문법으로 어떤 의미를 갖는가?

자료 타입 : 기본자료형, 클래스형

up casting, : 자식클래스타입의 참조를 부모클래스의 변수에 넣는게 자연스럽다.

CharSequentce s;

String s1;

StringBuilder s2;

s=s1, s=s2;

?대신에 object class를 적어줘도 된다. 다 업캐스팅 된다.

뷰의 역할이 자식들을 거느려서 배치를 보여주는 놈이 뷰 그룹

추가된 코드는 줄46~68

(줄46~58) ListView의 항목 하나를 클릭했을 때 발생하는 ItemClick 이벤트에 대한 리스너(Listener) 객체를 생성하여, ListView에 등록한다.

MainActivity 클래스의 onCreate 메소드를 실행할 때, 줄46~58을 실행할 때 일어나는 일은 다음과 같다.

1. OnItemClickListener 클래스의 자식 클래스가 생성된다. 이 자식 클래스의 이름은 주어지지 않았다.

2. 이름 없는 자식 클래스에서 onItemClick 메소드를 재정의(override)한다.

3. 이름 없는 자식 클래스의 객체가 하나 생성된다.

4. listView의 setOnItemClickListener 메소드가 호출되면서, 이름 없는 자식 클래스 객체에 대한 참조가 파라미터 값으로 전달된다. 이제 앞으로 listView 객체의 항목 하나가 클릭되면, 이 자식 객체의 onItemClick 메소드가 호출된다.

(줄48~57) MainActivity 클래스의 onCreate 메소드가 실행될 때 이 onItemClick 메소드는 호출되거나 실행되지 않는다. MainActivity 클래스의 onCreate 메소드가 실행을 마치어 리턴된 후에, listView의 항목이 클릭되면 그때 onItemClick 메소드가 호출된다.

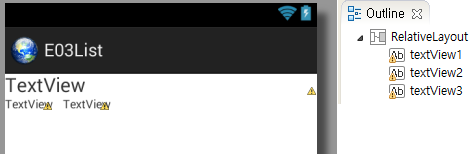
(줄48) listView의 항목이 클릭되면 onItemClick 메소드가 호출된다.

onItemClick 메소드의 첫째 파라미터는 클릭된 ListView 객체에 대한 참조이다.

파라미터 변수 타입인 AdapterView 클래스는 ListView 클래스의 부모 클래스이다.

onItemClick 메소드의 두번째 파라미터는 클릭된 항목에 해당하는 뷰 객체에 대한 참조이다.

데이터 항목 하나에 해당하는 뷰 객체의 형태는 res/layout/list\_item.xml 레이아웃 리소스 파일로 정의되었다.



위 그림과 같이 RelativeLayout 객체가 부모 객체가 되고, 그 아래 TextView 객체 3개가 생성된다.

SimpleCursorAdapter 객체가 이 레이아웃 리소스의 내용대로 뷰 객체들을 만들어서 ListView 객체에 넘겨준다.

넘겨주기 전에 먼저 TextView 객체들에 값을 채운다. textView1에는 데이터 항목의 title 필드값을, textView2에는 modifiedTime 필드값을, textView3에는 \_id 필드값을 채운다.

어떤 뷰 객체에 어떤 필드값을 채울지는 SimpleCursorAdapter의 생성자 파라미터로 명시되었다 (줄40,41)

onItemClick 메소드의 두번째 파라미터는 클릭된 항목에 해당하는 뷰 객체에 대한 참조이다.

뷰 객체들의 부모인 RelativeLayout 객체에 대한 참조가 전달될 것이다.

onItemClick 메소드의 3번째 파라미터는 클릭된 항목이 목록에서 몇번째 항목인지를 알려준다. 클릭된 항목이 목록에서 첫째 항목이었다면 3번째 파라미터 값으로 0 이 전달되고, 그 다음 항목이었다면 1 이 전달된다.

onItemClick 메소드의 4번째 파라미터는 클릭된 항목에 해당하는 데이터의 기본키(primary key)인 \_id 필드 값이다. 이렇게 ListView에서 \_id 필드값을 사용하기 때문에, ListView에 채울 데이터를 조회하기 위한 SQL SELECT 문은 반드시 \_id 필드를 포함해야 한다.

(줄49) RelativeLayout 객체의 findViewById 메소드를 호출한다. 이 메소드 호출은 RelativeLayout 객체가 거느리고 있는 객체들 중에서 id가 textView1인 뷰 객체를 찾아서 그 객체에 대한 참조를 리턴한다. 리턴된 참조 값을 지역변수 textView1에 저장한다.

(줄52) String 클래스의 format 메소드를 호출한다. 이 메소드는 클래스 메소드(static method)이다.

이 메소드는 파라미터 값으로 문자열을 하나 생성하여 리턴한다.

이 메소드의 사용법은 C언어의 printf 함수와 유사하다. 즉 C언어의 printf 함수호출로 화면에 출력될 내용이, String 클래스의 format 메소드의 리턴값인 문자열로 리턴된다.

(줄52) Locale.KOREA

날짜, 시각, 돈의 단위 등 나라마다 다른 형태로 출력하는 값들의 출력 형태를 지정하기 위한 국가 코드이다.

(줄53) "title:%s modifiedTime:%s \_id:%s position:%d id:%d"

C언어의 printf의 첫번째 파라미터와 내용이 같은 포멧 스트링(format string)이다.

(줄52~54) TextView에 채워졌던 데이터 항목의 값과 파라미터 변수값들을 줄53의 포멧 스트링의 해당 위치에 끼워 넣어서 문자열을 만든다. 그 문자열에 대한 참조가 지역변수 s에 저장된다.

(줄55) Toast.makeText 메소드를 호출한다. 이 메소드는 클래스 메소드(static method)이다.

이 메소드는 Toast 객체를 생성해서 리턴한다.

Toast 객체는 화면에 잠깐 메시지를 표시할 때 사용하는 객체이다.

makeToast 메소드의 2번째 파라미터가 화면에 표시할 문자열이다.

makeToast 메소드의 3번째 파라미터는 메시지를 화면에 얼마동안 표시할지를 결정한다.

Toast.LENGTH\_LONG 이면 길게, Toast.LENGTH\_SHORT 이면 짧게 표시한다.

LENGTH\_LONG과 LENGTH\_SHORT는 Toast 클래스의 public static final int 멤버이다.

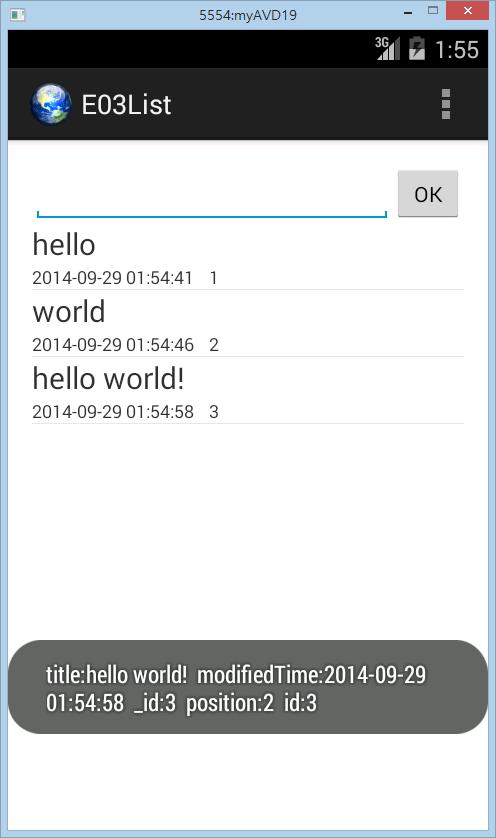
makeToast 메소드의 1번째 파라미터는 액티비티에 대한 참조이다.

onItemClick 메소드는 MainActivity 클래스의 메소드가 아니고, OnItemClickListener의 이름 없는 자식 클래스의 메소드이다. 따라서 onItemClick 메소드 내부의 this는 MainActivity가 아니고 OnItemClickListener의 이름 없는 자식 클래스의 객체를 참조한다.

OnItemClickListener의 이름 없는 자식 클래스는 MainActivity 클래스의 내부에서 정의된 안쪽(inner) 클래스이고, MainActivity 클래스는 바깥쪽(outter) 클래스이다. 안쪽 클래스의 메소드에서 바깥쪽 클래스의 멤버 변수와 메소드를 사용할 수 있고, 바깥쪽 클래스의 this도 사용할 수 있다. 안쪽 클래스에서 그냥 this는 안쪽 클래스 객체를 가르키고, 바깥쪽클래스명.this는 바깥쪽 클래스 객체를 가르킨다. 줄55의 MainActivity.this는 바깥쪽 클래스인 MainActivity 클래스 객체를 가르킨다.

(줄55) makeToast 메소드가 생성하여 리턴한 Toast 객체에 대한 참조를 지역 변수 toast에 저장한다.

(줄56) Toast 클래스의 show 메소드를 호출한다. Toast의 생성자 파라미터로 전달된 문자열이 화면에 잠깐 표시된다.



(줄59~68) ListView 항목 하나를 꾸욱 눌러서 길게 클릭했을 때 발생하는 ItemLongClick 이벤트에 대한 리스너 객체를 생성하여, ListView에 등록한다.

(줄61~66) MainActivity 클래스의 onCreate 메소드가 실행될 때 이 onItemClick 메소드는 호출되거나 실행되지 않는다. MainActivity 클래스의 onCreate 메소드가 실행을 마치어 리턴된 후에, listView의 항목이 클릭되면 그때 onItemClick 메소드가 호출된다.

(줄61) onItemLongClick 메소드의 파라미터는 OnItemClickListener의 onItemClick 메소드의 그것과 같다.

(줄62) 클릭된 데이터 항목의 기본키 \_id 필드값을 파라미터로 DELETE SQL 문을 실행한다.

해당 레코드가 DB에서 삭제된다.

(줄63) DB에서 다시 조회하기 전에, 이제 필요없어진 이전의 조화결과인 Cursor 객체의 Close 메소드를 호출하여 닫는다. 여기서 cursor는 MainActivity 클래스의 멤버 변수이다. 안쪽 클래스인 OnItemLongClickListener의 자식 클래스에서 바깥 클래스인 MainActivity의 멤버 변수를 사용할 수 있다.

(줄64) DB에서 ListItem 테이블을 다시 조회 하기 위한 SELECT SQL 문장을 실행하기 위해, SQLiteDatabase 클래스의 rawQuery 메소드를 호출한다. 조회 결과 데이터는 Cursor 클래스 객체로 리턴된다.

조회결과로 리턴된 Cursor 객체에 대한 참조를 멤버 변수 cursor에 저장한다.

(줄65) 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.

(줄66) LongClick 이벤트 처리가 완료되었다는 의미로 true를 리턴한다.

# ActionBar 제목 변경

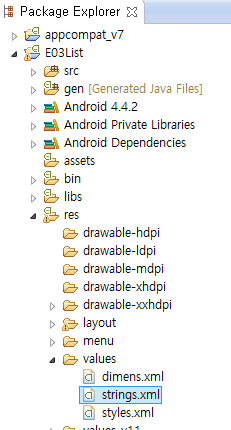
화면 상단에 앱의 아이콘과 앱의 제목이 표시된 영역을 ActionBar라고 한다.

액션바에 표시되는 제목의 디폴트 값은 앱의 제목이다.

액션바의 제목을 변경해 보자.

## 문자열 리소스 만들기

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



이 파일을 아래와 같이 수정하자.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?xml version="1.0" encoding="utf-8"?>  <resources>  <string name="app\_name">E03List</string>  <string name="hello\_world">Hello world!</string>  <string name="action\_settings">Settings</string>  <string name="ok">OK</string>  <string name="mainActivityTitle">Data List</string>  </resources> |

(줄7) MainActivity의 제목으로 표시될 문자열 리소스 항목이 추가되었다.

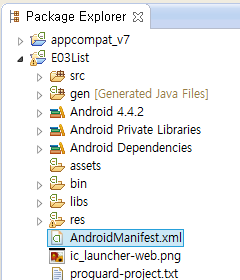
화면에 표시될 문자열은 전부 문자열 리소스(string resource)로 정의하는 것이 바람직하다.

문자열 리소스를 사용하면 다국어 버전을 만들기 쉽기 때문이다.

이 문자열 리소스의 아이디는 Java 소스코드에서 사용할 때 R.string.mainActivityTitle 이고, XML 파일에서 사용할 때는 @string/mainActivityTitle 이다.

## AndroidManifest.xml 파일 수정

액티비티에 대한 기본 정보는 AndroidManifest.xml 파일에 들어있다.



이 파일을 다음과 같이 수정하자.

|  |  |
| --- | --- |
| 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 | <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.example.e03list"  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" >  <activity  android:name=".MainActivity"  android:label="@string/mainActivityTitle" >  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  </application>  </manifest> |

(줄18) activity 태그의 andoid:label 항목이 바로 actionBar에 표시되는 제목이다.

이 제목은 @string/mainActivityTitle 문자열 리소스 아이디로 변경하였다.

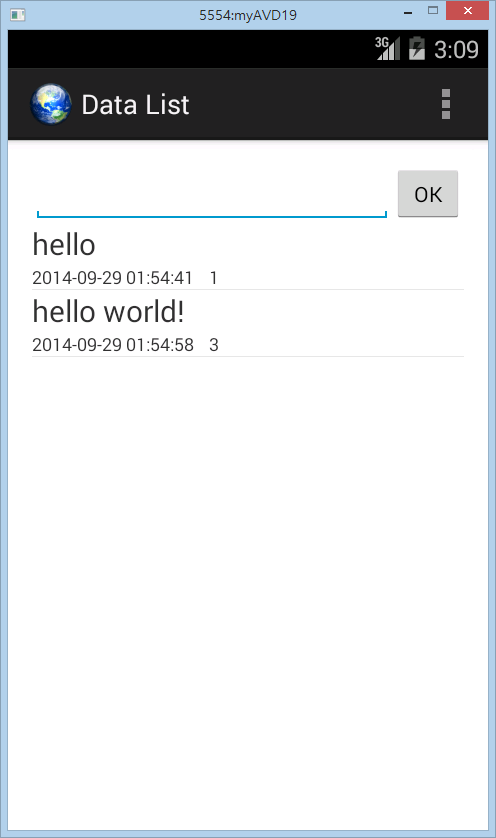
# 실행

앱의 아이콘이 변경되었음을 확인하자.

액티비티의 제목줄이 변경되었음을 확인하자.

ListView의 항목을 클릭해서 Toast 메시지 출력을 확인하자.

Listview의 항목을 길게 클릭해서 삭제해 보자.



### 소스코드

E03List\_#2.zip

# 과제 #1

길게 클릭하여 데이터 항목이 삭제되면 "Deleted" 라는 Toast 메시지가 짧게 표시되도록 구현하자.

# MainActivity.java 코드 중복 제거

MainActivity.java에서 아래 코드가 3번 들어있다.

cursor.close();

cursor = db.rawQuery("SELECT \* FROM ListItem", null);

adapter.changeCursor(cursor);

(1) ItemList 테이블의 레코드가 삭제된 후 다시 조회하기 위해서 실행함

(2) ItemList 테이블의 레코드가 추가된 후 다시 조회허가 위해서 실행함

(3) onResume 메소드에서 데이터를 처음 화면에 보여주기 위해 최초로 조회하기 위해서 실행함.

동일한 소스코드가 여러번 반복되는 것은 바람직하지 않다.

동일한 소스코드를 아래와 같이 loadDataToAdpater 메소드로 추출하고, 이 메소드를 호출하여 실행하는 것이 바람직하다.

private void loadDataToAdapter() {

cursor.close();

cursor = db.rawQuery("SELECT \* FROM ListItem", null);

adapter.changeCursor(cursor);

}

그런데 onResume 메소드에서 위 메소드를 호출할 때는 cursor.close()를 생략해야 한다.

onResume 메소드에서 loadDataToAdpater() 메소드를 호출했을 때는 cursor.close() 가 생략되도록 다음과 같이 수정하자.

private void loadDataToAdapter() {

if (cursor != null) cursor.close();

cursor = db.rawQuery("SELECT \* FROM ListItem", null);

adapter.changeCursor(cursor);

}

onResume 메소드에서 loadDataToAdapter 메소드를 호출할 때에만 cursor 멤버 변수 값이 null 이고, 다른 곳에서 loadDataToAdapter 메소드를 호출할 때에는 cursor 멤버 변수 값이 null 이 아니면, onResume 메소드에서 호출할 때에만 cursor.close()가 생략된다.

아래 소스 코드를 실행 흐름의 순서로 읽으면서 멤버 변수 cursor 값이 어떻게 변하는지 분석해 보자.

### src/com.example.e03list/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  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  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119 | package com.example.e03list;  import java.util.Locale;  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.AdapterView;  import android.widget.AdapterView.OnItemClickListener;  import android.widget.AdapterView.OnItemLongClickListener;  import android.widget.Button;  import android.widget.EditText;  import android.widget.ListView;  import android.widget.TextView;  import android.widget.Toast;  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) {  TextView textView1 = (TextView)view.findViewById(R.id.textView1);  TextView textView2 = (TextView)view.findViewById(R.id.textView2);  TextView textView3 = (TextView)view.findViewById(R.id.textView3);  String s = String.format(Locale.KOREA,  "title:%s modifiedTime:%s \_id:%s position:%d id:%d",  textView1.getText(), textView2.getText(), textView3.getText(), position, id);  Toast toast = Toast.makeText(MainActivity.this, s, Toast.LENGTH\_LONG);  toast.show();  }  });  listView.setOnItemLongClickListener(new OnItemLongClickListener() {  @Override  public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {  db.execSQL("delete from ListItem where \_id = " + id);  loadDataToAdapter();  return true;  }  });  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);  loadDataToAdapter();  }  });  }    private void loadDataToAdapter() {  if (cursor != null) 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();  loadDataToAdapter();  }  @Override  protected void onPause() {  super.onPause();  cursor.close(); cursor = null;  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);  }  } |

### 소스코드

E03List\_#3.zip

# 과제 #2

위 소스 코드를 실행 흐름 순서를 따라 읽고 분석하라.

예를 들어, 앱을 실행 한 후 1초 뒤에 데이터를 하나 입력하고 OK 버튼을 클릭하고,

또 2초 뒤에 데이터 항목 하나를 길게 클릭하여 삭제하고,

잠시 다른 앱을 실행했다가 백버튼을 눌러서 다시 E03List 앱으로 돌아가고,

앱을 종료한 사례를 가정하자.

이 사례에서 앱 시작부터 앱 종료까지 순서대로 어떤 메소드가 호출되었는지 나열하고, 메소드 호출 직후 멤버 변수 cursor 값이 null 인지 아닌지 파악해서 적어라.

MainActivity.java 파일에서 정의된 메소드의 호출 순서만 고려하자.

|  |  |  |
| --- | --- | --- |
| 순서 | 호출되는 메소드 | 메소드 호출 후 멤버 변수 cursor 값이 null 인가? |
| 0 | 앱 시작 | ... |
| 1 | onCreate | ... |
| 2 | ... | ... |
| 3 | ... | ... |
| ... | ... | ... |

위 표의 나머지 부분을 채워서 제출하라.