MyMemo 개선

2014-11-25

이승진

**목차**

[1. MyMemo 개선 2](#_Toc405526077)

[2. MainActivity에 애니메이션 효과 추가 2](#_Toc405526078)

[1) res/anim 폴더 생성 2](#_Toc405526079)

[2) res/anim/animation1.xml 파일 생성 2](#_Toc405526080)

[3) res/anim/layout\_animation1.xml 파일 생성 2](#_Toc405526081)

[4) res/layout/activity\_main.xml 수정 3](#_Toc405526082)

[5) 실행 3](#_Toc405526083)

[3. 예외처리 구현 4](#_Toc405526084)

[1) 강제로 예외 발생 4](#_Toc405526085)

[2) 강제로 예외 발생 테스트 4](#_Toc405526086)

[3) 에러 메시지 문자열 리소스 추가 5](#_Toc405526087)

[4) ExceptionHandler.java 생성 5](#_Toc405526088)

[5) MainActivity.java 수정 6](#_Toc405526089)

[4. 삭제 확인 대화상자 8](#_Toc405526090)

[1) 삭제 확인 문자열 리소스 추가 8](#_Toc405526091)

[2) EditActivity.java 수정 8](#_Toc405526092)

[3) 실행 10](#_Toc405526093)

[5. 한글 문자열 리소스 XML 파일 생성 11](#_Toc405526094)

[1) res/values-ko 폴더 생성 11](#_Toc405526095)

[2) res/values-ko/strings.xml 파일 11](#_Toc405526096)

[3) 실행 11](#_Toc405526097)

# MyMemo 개선

지금까지 개발한 MyMemo 앱을 개선하자.

강의노트 웹폴더의 소스코드 폴더에서 MyMemo#3.zip 파일을 다운로드 받자.

# MainActivity에 애니메이션 효과 추가

## res/anim 폴더 생성

애니메이션 효과를 정의하는 XML 파일은 res/anim 폴더에 저장해야 한다.

res 폴더 아래에 anum 폴더를 생성하자.

이클립스의 Package Explorer 창에서 res 폴더를 우클릭하고 메뉴에서 New – Folder를 클릭하자.

Folder name: anim

## res/anim/animation1.xml 파일 생성

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?xml version="1.0" encoding="utf-8"?>  <set xmlns:android="http://schemas.android.com/apk/res/android"  android:interpolator="@android:anim/accelerate\_interpolator" >  <translate  android:duration="500"  android:fromXDelta="100%"  android:toXDelta="0" />  </set> |

뷰 객체 하나에 대한 애니메이션을 정의한 XML 파일이다.

(줄5) translate 애니메이션을 정의한다. translate 애니메이션은 좌표 이동이다.

(줄6) 애니메이션은 500 밀리초 동안 실행된다

(줄7) 이동 애니메이션이 시작할 시작 x 좌표는 화면의 오른쪽 끝(100%)이다.

(줄8) 이동 애니메이션이 끝나는 마지막 x 좌표는 화면의 왼쪽 끝이다.

## res/anim/layout\_animation1.xml 파일 생성

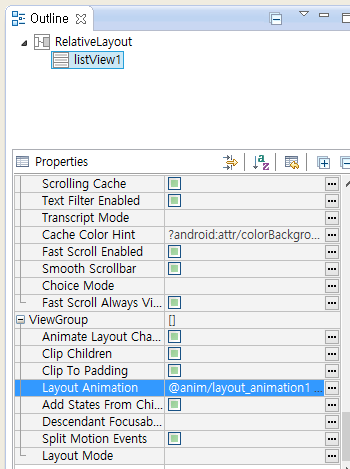
|  |  |
| --- | --- |
| 1  2  3  4 | <?xml version="1.0" encoding="utf-8"?>  <layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"  android:animation="@anim/animation1"  android:delay="10%" /> |

ListView와 같은 뷰그룹(ViewGroup) 내부의 뷰(View) 항목들 각각에 대한 애니메이션을 정의한 XML 파일이다.

(줄3) 뷰 항목 각각에 대해 적용할 애니메이션을 지정한다.

(줄4) 뷰 항목 각각의 애니메이션 사이 대기 시간을 지정한다. 뷰 항목 하나의 애니메이션 길이(duration)의 10% 라는 의미.

## res/layout/activity\_main.xml 수정



listView1의 Layout Animation 속성을 @anim/layout\_animation1 으로 설정한다.

## 실행

실행하여 레이아웃 애니메이션의 효과를 확인하라.

# 예외처리 구현

안드로이드 앱을 실행하는 도중에 여러가지 이유로 에러가 발생할 수 있다.

안드로이드 앱에서 에러가 발생하면, 기본적으로 앱이 강제로 종료된다.

Java 언어에서 에러를 예외(exception)이라고 부른다.

## 강제로 예외 발생

강제로 예외를 발생시켜서 앱이 죽는 것을 확인해보자.

MainActivity의 메뉴에서 Settings를 누르면 강제로 예외를 발생하도록 구현하자.

src/skhu.sw.mymemo/MainActivity.java 수정

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

if (id == R.id.action\_settings) {

throw new RuntimeException("Exception Test");

// return true;

} else if (id == R.id.action\_new) {

Intent intent = new Intent(this, EditActivity.class);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

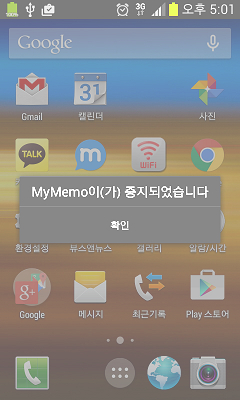
}

MainActivity 클래스의 onOptionsItemSelected 메소드에서

return true; 문장을 주석으로 막고, 대신에 throw new RuntimeException("Exception Text"); 문장을 추가하자.

## 강제로 예외 발생 테스트

MyMemo 앱을 실행하여 MainActivity의 메뉴에서 Settings 메뉴를 눌러보자.



예외가 발생하면 앱이 강제로 종료되는 이 기본 동작을 수정하자.

예외가 발생하면 에러 메시지를 대화상자로 보여주고, 앱이 종료되지 않도록 하자.

## 에러 메시지 문자열 리소스 추가

문자열 리소스 XML 파일에 에러 메시지 문자열을 추가하자.

문자열 리소스 XML 파일 편집

res/values/strings.xml 편집

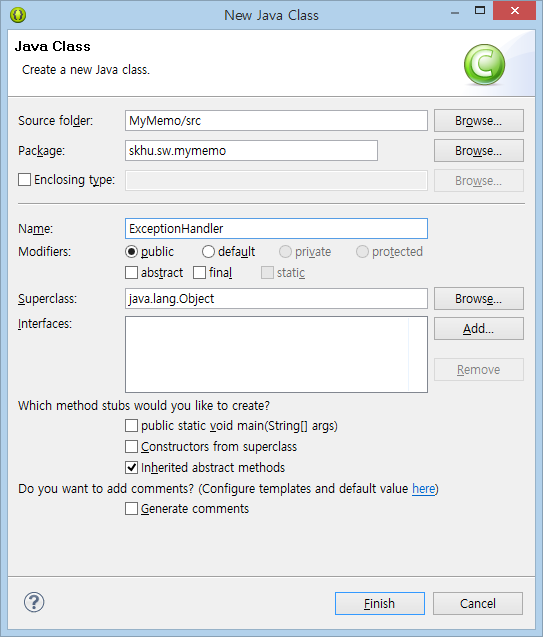
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?xml version="1.0" encoding="utf-8"?>  <resources>  <string name="app\_name">MyMemo</string>  <string name="hello\_world">Hello world!</string>  <string name="action\_settings">Settings</string>  <string name="title\_activity\_edit">Edit Memo</string>  <string name="action\_edit">Edit</string>  <string name="action\_save">Save</string>  <string name="action\_new">New</string>  <string name="action\_remove">Remove</string>  <string name="action\_close">Close</string>  <string name="error\_title">Runtime Error</string>    </resources> |

줄12,13의 문자열 리소스가 추가되었다.

## ExceptionHandler.java 생성

skhu.sw.mymemo 패키지 아래에 ExceptionHandler 클래스를 생성하자.

이클립스 메뉴에서 File - New - Class



Package: skhu.sw.mymemo

Name: ExceptionHandler

### ExceptionHandler.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package skhu.sw.mymemo;  import android.app.AlertDialog;  import android.content.Context;  public class ExceptionHandler {    public static void showDialog(Context context, Exception exception) {  AlertDialog.Builder builder = new AlertDialog.Builder(context);  builder.setTitle(R.string.error\_title);  builder.setMessage(exception.getMessage());  builder.setNeutralButton(R.string.action\_close, null);  builder.show();  }  } |

(줄8) 예외(exception)가 발생하면, 예외 메시지를 대화상자로 보여주기 위한 메소드를 구현하였다.

첫째 파라미터는 Context 이다. Context는 Activity의 부모 클래스이다.

둘째 파라미터는 Exception 이다.

(줄9) 대화상자를 보여주기위해 필요한 클래스는 AlertDialog.Builder 이다.

Java 언어에서는 클래스 내부에 멤버 변수나 메소드를 정의할 수 있을 뿐만 아니라, 클래스 내부에 클래스를 정의할 수도 있다.

다른 클래스 내부에 정의된 클래스를 내부 클래스(inner class)라고 부른다.

AlertDialog.Builder는 AlertDialog 클래스 내부에 정의된 Builder 클래스의 이름이다.

내부 클래스 이름을 사용할 때는 바깥 클래스 이름을 앞에 붙여주어야 한다.

(줄10) 대화상자의 제목줄을 지정한다.

R.string.error\_title는 문자열 리소스 아이디이다.

(줄11) 대화상자의 내용 문자열을 지정한다.

exception.getMessage()는 예외에 대한 설명 문자열을 리턴한다.

(줄12) 대화상자에 표시할 버튼을 하나 추가한다.

이 메소드의 첫째 파라미터는 버튼에 표시할 문자열이다.

R.string.action\_close는 문자열 리소스 아이디이다.

이 버튼이 눌러지면 대화상자는 저절로 닫힌다.

(줄13) 대화상자를 화면에 표시한다.

## MainActivity.java 수정

public boolean onOptionsItemSelected(MenuItem item) {

try {

int id = item.getItemId();

if (id == R.id.action\_settings) {

throw new RuntimeException("Exception Test");

// return true;

} else if (id == R.id.action\_new) {

Intent intent = new Intent(this, EditActivity.class);

startActivity(intent);

return true;

}

} catch (Exception exception) {

ExceptionHandler.showDialog(this, exception);

}

return super.onOptionsItemSelected(item);

}

예외가 발생할 수 있는 문장들을 try 블럭으로 묶어주면,

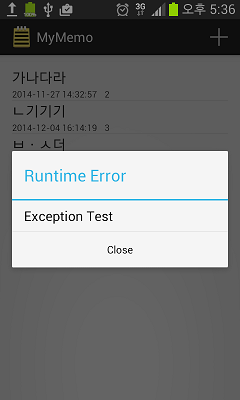
try 블럭 내부의 문장을 실행하다가 예외가 발생한 경우 catch 블럭으로 점프하게 된다.

이때 그 예외가 catch 블럭의 파라미터로 전달된다.

catch 블럭에서는 전달 받은 예외에 대한 정보를 대화상자로 보여주고 있다.

예외에 대한 정보를 대화상자로 보여주기 위해, 방금 구현한 ExceptionHandler 클래스의 showDialog 메소드를 호출하고 있다.

이 메소드 호출의 첫째 파라미터 this는 MainActivity 객체이다. 둘째 파라미터는 catch 블럭에 전달된 예외 객체이다.



# 삭제 확인 대화상자

메모 삭제 메뉴 핸들러에서, 바로 메뉴를 삭제하지 않고, 삭제 여부를 묻는 대화상자를 화면에 표시하자.

대화상자에서 삭제 버튼이 눌려지면 메모를 삭제하고, 취소 버튼이 눌려지면 삭제하지 않는다.

## 삭제 확인 문자열 리소스 추가

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="app\_name">MyMemo</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>

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

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

<string name="action\_close">Close</string>

<string name="action\_cancel">Cancel</string>

<string name="error\_title">Runtime Error</string>

<string name="confirm\_delete">Do you want to delete?</string>

</resources>

R.string.action\_cancel, R.string.confirm\_delete 문자열 리소스가 추가되었다.

## 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  84  85  86  87  88  89  90  91  92  93  94  95  96 | package skhu.sw.mymemo;  import android.support.v4.app.NavUtils;  import android.support.v7.app.ActionBarActivity;  import android.app.AlertDialog;  import android.content.ContentValues;  import android.content.DialogInterface;  import android.content.DialogInterface.OnClickListener;  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) {  AlertDialog.Builder builder = new AlertDialog.Builder(this);  builder.setTitle(R.string.action\_remove);  builder.setMessage(R.string.confirm\_delete);  builder.setNegativeButton(R.string.action\_cancel, null);  builder.setPositiveButton(R.string.action\_remove, new OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int button) {  LectureDBOpenHelper helper = new LectureDBOpenHelper(EditActivity.this);  SQLiteDatabase db = helper.getWritableDatabase();  db.execSQL("delete from MyMemo where \_id = " + \_id);  db.close();  helper.close();  NavUtils.navigateUpFromSameTask(EditActivity.this);  }  });  builder.show();  }  return true;  }  return super.onOptionsItemSelected(item);  }  } |

EditActivity에서 Remove 메뉴가 눌러지면 줄54의 이벤트 핸들러 메소드가 저절로 호출된다.

onOptionsItemSelected는 메뉴 이벤트 핸들러이다.

Java에서 on으로 시작하는 이름의 메소드는 이벤트 핸들러 메소드이다.

이벤트 핸들러 메소드는 어떤 이벤트가 발생하면 저절로 호출된다.

안드로이드에서 앱의 주메뉴를 옵션 메뉴(option menu)라고 부른다.

onOptionsItemSelected 메소드는 옵션 메뉴 항목(option menu item)이 선택되면 저절로 즉시 호출되는 메소드이다.

(줄73) 멤버 변수 \_id는 현재 표시되는 메모의 기본키(primary key)값이다.

SQLiteDataBase에서 모든 테이블의 기본키는 int 타입의 \_id 필드이어야 한다.

이 필드값은 1부터 시작하는 일련번호이다. \_id 필드값 0은 가능하지 않다.

(줄74) 대화상자를 보여주기위해 필요한 클래스는 AlertDialog.Builder 이다.

(줄75) 대화상자의 제목줄을 지정한다.

R.string.action\_remove는 문자열 리소스 아이디이다.

(줄76) 대화상자의 내용 문자열을 지정한다.

R.string.confirm\_delete는 문자열 리소스 아이디이다.

(줄77) setNegativeButton 메소드는 대화상자에 표시할 버튼을 하나 추가한다.

이 메소드의 첫째 파라미터는 버튼에 표시할 문자열이다.

R.string.action\_cancel은 문자열 리소스 아이디이다.

이 버튼이 눌러지면 대화상자는 저절로 닫힌다.

(줄78~88) setPositiveButton 메소드는 대화상자에 표시할 버튼을 하나 추가한다.

positive button은 일종의 yes 버튼이다.

negative button은 일종의 no 버튼이다.

neutral button은 평범한 버튼이다.

setPositiveButton 메소드의 첫째 파라미터는 버튼에 표시할 문자열 리소스 ID이다.

setPositiveButton 메소드의 둘째 파라미터는 버튼이 눌러졌을 때 저절로 호출될 이벤트 핸들러 객체(리스너 객체)이다.

대화상자의 positive 버튼이 눌러지면 OnClickListener 객체의 onClick 메소드가 호출된다.

setPositiveButton 메소드의 둘째 파라미터는 다음과 같다.

new OnClickListener() {

@Override

public void onClick(...) {

...

}

}

anonymous inner class 문법이다.

삭제 메뉴가 눌러졌을 때 실행되는 코드는 다음과 같다.

1) 줄74 객체 생성

2) 줄75 setTitle 메소드

3) 줄76 setMessage 메소드

4) 줄77 setPositiveButton 메소드

5) 줄78 setNegativeButton 메소드

6) 줄89 show 메소드. 대화상자 표시됨

대화상자에서 positive 버튼이 눌러졌을 때 실행되는 코드는 다음과 같다

1) 줄80 onClick 메소드

2) 줄81 객체 생성

3) 줄82 getWritableDatabase 메소드

4) 줄83 execSQL 메소드. DB에서 삭제됨

5) 줄84 close 메소드.

6) 줄85 close 메소드.

7) 줄86 navigateUpFromSameTask 메소드. MainActivity로 돌아감.

## 실행

삭제 버튼을 눌러서 삭제 확인 대화상자를 실행해 보자.

# 한글 문자열 리소스 XML 파일 생성

기본 문자열 리소스 XML 파일인 res/values/strings.xml 파일의 문자열들은 가급적 기본 언어인 영어로 만드는 것이 좋다.

앱 개발이 끝난 후 한글 문자열 리소스 XML 파일만 만들어주면 앱이 한글화된다.

소스코드에서 수정할 부분은 없다.

스마트폰의 언어 설정이 한글이면, 한글 문자열 리소스 XML 파일이 사용되고, 그렇지 않으면, 기본 문자열 리소스 XML 파일이 사용된다.

## res/values-ko 폴더 생성

res 폴더 아래에 values-ko 폴더를 생성한다.

이클립스에서 res 폴더를 우클릭하고 메뉴에서 New – Folder 클릭

폴더명 values-ko

## res/values-ko/strings.xml 파일

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?xml version="1.0" encoding="utf-8"?>  <resources>  <string name="app\_name">MyMemo</string>  <string name="hello\_world">안녕하세요!</string>  <string name="action\_settings">설정</string>  <string name="title\_activity\_edit">메모 편집</string>  <string name="action\_edit">편집</string>  <string name="action\_save">저장</string>  <string name="action\_new">작성</string>  <string name="action\_remove">삭제</string>  <string name="action\_close">닫기</string>  <string name="action\_cancel">취소</string>  <string name="error\_title">에러 발생</string>  <string name="confirm\_delete">삭제하시겠습니까?</string>    </resources> |

res/values/strings.xml 파일은 영어 문자열 리소스 XML 파일이다.

res/values-ko/strings.xml 파일은 한글 문자열 리소스 XML 파일이다.

## 실행

앱을 실행하여 메뉴와 대화상자의 한글 출력을 확인하자.

## 결과

MyMemo#4.zip