[CursorLoader](http://startandroid.ru/ru/uroki/vse-uroki-spiskom/278-urok-136-cursorloader.html" \o "Урок 136. CursorLoader)

В Уроке 52 использованы устаревшие на данный момент методы запроса данных от БД и связки Activity с Cursor.  Вместо них рекомендуется использовать [CursorLoader](http://developer.android.com/reference/android/content/CursorLoader.html" \t "_blank), который будет асинхронно читать данные и возвращать Cursor. Этот урок будет являться копией Урока 52 только с использованием CursorLoader.

CursorLoader представляет собой наследника класса AsyncTaskLoader<Cursor> и по умолчанию заточен на работу с ContentProvider, т.к. при работе требует Uri. Мы же в этом примере используем его для работы со своей БД. Для этого нам придется его расширить и вставить свою реализацию в его основной метод.

Приложение урока - это список, который отображает содержимое БД. Кнопкой можно записи добавлять, а контекстным меню - удалять.

Создадим проект:

**Project name**: P1361\_CursorLoader  
**Build Target**: Android 2.3.3   
**Application name**: CursorLoader  
**Package name**: ru.startandroid.develop.p1361cursorloader  
**Create Activity**: MainActivity

В **strings.xml** добавим строки:

|  |  |
| --- | --- |
| 1  2 | <string name="add\_record">Добавить запись</string>  <string name="delete\_record">Удалить запись</string> |

Экран **main.xml**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:orientation="vertical">      <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:onClick="onButtonClick"          android:text="@string/add\_record">      </Button>      <ListView          android:id="@+id/lvData"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content">      </ListView>  </LinearLayout> |

Кнопка добавления записи и список

Layout пункт-а списка **item.xml**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="match\_parent"      android:layout\_height="wrap\_content"      android:orientation="horizontal">      <ImageView          android:id="@+id/ivImg"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:src="@drawable/ic\_launcher">      </ImageView>      <TextView          android:id="@+id/tvText"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:layout\_gravity="center\_vertical"          android:layout\_marginLeft="10dp"          android:text=""          android:textSize="18sp">      </TextView>  </LinearLayout> |

Статичная картинка и текст.

Работу с БД вынесем в отдельный класс **DB.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 | package ru.startandroid.develop.p1361cursorloader;    import android.content.ContentValues;  import android.content.Context;  import android.database.Cursor;  import android.database.sqlite.SQLiteDatabase;  import android.database.sqlite.SQLiteDatabase.CursorFactory;  import android.database.sqlite.SQLiteOpenHelper;    public class DB {      private static final String DB\_NAME = "mydb";    private static final int DB\_VERSION = 1;    private static final String DB\_TABLE = "mytab";      public static final String COLUMN\_ID = "\_id";    public static final String COLUMN\_IMG = "img";    public static final String COLUMN\_TXT = "txt";      private static final String DB\_CREATE =      "create table " + DB\_TABLE + "(" +        COLUMN\_ID + " integer primary key autoincrement, " +        COLUMN\_IMG + " integer, " +        COLUMN\_TXT + " text" +      ");";      private final Context mCtx;        private DBHelper mDBHelper;    private SQLiteDatabase mDB;      public DB(Context ctx) {      mCtx = ctx;    }      // открыть подключение    public void open() {      mDBHelper = new DBHelper(mCtx, DB\_NAME, null, DB\_VERSION);      mDB = mDBHelper.getWritableDatabase();    }      // закрыть подключение    public void close() {      if (mDBHelper!=null) mDBHelper.close();    }      // получить все данные из таблицы DB\_TABLE    public Cursor getAllData() {      return mDB.query(DB\_TABLE, null, null, null, null, null, null);    }      // добавить запись в DB\_TABLE    public void addRec(String txt, int img) {      ContentValues cv = new ContentValues();      cv.put(COLUMN\_TXT, txt);      cv.put(COLUMN\_IMG, img);      mDB.insert(DB\_TABLE, null, cv);    }      // удалить запись из DB\_TABLE    public void delRec(long id) {      mDB.delete(DB\_TABLE, COLUMN\_ID + " = " + id, null);    }      // класс по созданию и управлению БД    private class DBHelper extends SQLiteOpenHelper {        public DBHelper(Context context, String name, CursorFactory factory,          int version) {        super(context, name, factory, version);      }        // создаем и заполняем БД      @Override      public void onCreate(SQLiteDatabase db) {        db.execSQL(DB\_CREATE);          ContentValues cv = new ContentValues();        for (int i = 1; i < 5; i++) {          cv.put(COLUMN\_TXT, "sometext " + i);          cv.put(COLUMN\_IMG, R.drawable.ic\_launcher);          db.insert(DB\_TABLE, null, cv);        }      }        @Override      public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {      }    }  } |

Здесь создание БД, управление подключением и методы по чтению/добавлению/удалению записей.

**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  120  121 | package ru.startandroid.develop.p1361cursorloader;    import java.util.concurrent.TimeUnit;    import android.content.Context;  import android.database.Cursor;  import android.os.Bundle;  import android.support.v4.app.FragmentActivity;  import android.support.v4.app.LoaderManager.LoaderCallbacks;  import android.support.v4.content.CursorLoader;  import android.support.v4.content.Loader;  import android.support.v4.widget.SimpleCursorAdapter;  import android.view.ContextMenu;  import android.view.ContextMenu.ContextMenuInfo;  import android.view.MenuItem;  import android.view.View;  import android.widget.AdapterView.AdapterContextMenuInfo;  import android.widget.ListView;    public class MainActivity extends FragmentActivity implements LoaderCallbacks<Cursor> {      private static final int CM\_DELETE\_ID = 1;    ListView lvData;    DB db;    SimpleCursorAdapter scAdapter;      /\*\* Called when the activity is first created. \*/    public void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.main);        // открываем подключение к БД      db = new DB(this);      db.open();        // формируем столбцы сопоставления      String[] from = new String[] { DB.COLUMN\_IMG, DB.COLUMN\_TXT };      int[] to = new int[] { R.id.ivImg, R.id.tvText };        // создаем адаптер и настраиваем список      scAdapter = new SimpleCursorAdapter(this, R.layout.item, null, from, to, 0);      lvData = (ListView) findViewById(R.id.lvData);      lvData.setAdapter(scAdapter);        // добавляем контекстное меню к списку      registerForContextMenu(lvData);        // создаем лоадер для чтения данных      getSupportLoaderManager().initLoader(0, null, this);    }      // обработка нажатия кнопки    public void onButtonClick(View view) {      // добавляем запись      db.addRec("sometext " + (scAdapter.getCount() + 1), R.drawable.ic\_launcher);      // получаем новый курсор с данными      getSupportLoaderManager().getLoader(0).forceLoad();    }      public void onCreateContextMenu(ContextMenu menu, View v,        ContextMenuInfo menuInfo) {      super.onCreateContextMenu(menu, v, menuInfo);      menu.add(0, CM\_DELETE\_ID, 0, R.string.delete\_record);    }      public boolean onContextItemSelected(MenuItem item) {      if (item.getItemId() == CM\_DELETE\_ID) {        // получаем из пункта контекстного меню данные по пункту списка        AdapterContextMenuInfo acmi = (AdapterContextMenuInfo) item            .getMenuInfo();        // извлекаем id записи и удаляем соответствующую запись в БД        db.delRec(acmi.id);        // получаем новый курсор с данными        getSupportLoaderManager().getLoader(0).forceLoad();        return true;      }      return super.onContextItemSelected(item);    }      protected void onDestroy() {      super.onDestroy();      // закрываем подключение при выходе      db.close();    }      @Override    public Loader<Cursor> onCreateLoader(int id, Bundle bndl) {      return new MyCursorLoader(this, db);    }      @Override    public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {      scAdapter.swapCursor(cursor);    }      @Override    public void onLoaderReset(Loader<Cursor> loader) {    }      static class MyCursorLoader extends CursorLoader {        DB db;        public MyCursorLoader(Context context, DB db) {        super(context);        this.db = db;      }        @Override      public Cursor loadInBackground() {        Cursor cursor = db.getAllData();        try {          TimeUnit.SECONDS.sleep(3);        } catch (InterruptedException e) {          e.printStackTrace();        }        return cursor;      }      }  } |

В **onCreate** подключаемся к БД, создаем SimpleCursorAdapter, добавляем контекстное меню к списку и создаем CursorLoader. Я везде для лоадера буду использовать ID = 0.

В **onButtonClick** добавляем запись в БД, получаем лоадер и просим его получить для нас новый курсор с данными.

**onCreateContextMenu** – создание контекстного меню.

В **onContextItemSelected** мы реализуем удаление записи из БД. И после удаления снова просим лоадер дать нам новый курсор с данными.

В **onDestroy** отключаемся от БД.

Далее идут колбэк-методы интерфейса LoaderCallbacks.

В **onCreateLoader** создаем Loader и даем ему на вход объект для работы с БД.

В **onLoadFinished** мы получаем результат работы лоадера – новый курсор с данными. Этот курсор мы отдаем адаптеру методом swapCursor.

**MyCursorLoader** – наш лоадер, наследник класса CursorLoader. У него мы переопределяем метод **loadInBackground**, в котором просто получаем курсор с данными БД. Ну и я 3-х секундной паузой сэмулировал долгое чтение БД для наглядности асинхронной работы.

Все сохраняем, запускаем пример. Работает и добавление по нажатию кнопки и удаление через контекстное меню. Работает с задержкой в 3 секунды, но при этом не тормозит интерфейс, т.к. работа выполняется асинхронно.

Кроме асинхронной загрузки, CursorLoader:

- закрывает старый курсор при успешном получении нового  
- закрывает курсор при уничтожении лоадера (т.е. и при выходе из приложения)  
- при переходе в состояние «стартован» проверяет метку, которую ставит Observer и запускает работу, если данные изменились  
- при переходе в состояние «стартован» стартует работу, если еще не было получено никаких результатов (например при первом запуске).