**SimpleCursorAdapter и получение данных**

**Последнее обновление: 05.03.2017**

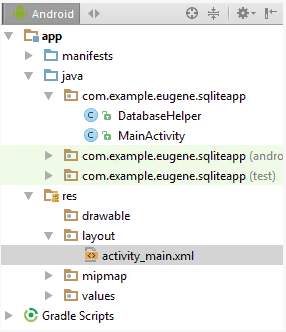
В прошлой теме было рассмотрено, как подключаться к базе данных SQLite и выполнять запросы. Теперь пойдем дальше и создадим полностью интерфейс для работы с базой данных.

Итак, создадим новый проект.

Для упрощения работы с базами данных SQLite в Android нередко применяется класс **SQLiteOpenHelper**. Для использования необходимо создать класса-наследник от SQLiteOpenHelper, переопределив как минимум два его метода:

* onCreate(): вызывается при попытке доступа к базе данных, но когда еще эта база данных не создана
* onUpgrade(): вызывается, когда необходимо обновление схемы базы данных. Здесь можно пересоздать ранее созданную базу данных в onCreate(), установив соответствующие правила преобразования от старой бд к новой

Поэтому добавим в проект, в ту же папку, где находится класс MainActivity, новый класс **DatabaseHelper**:



|  |
| --- |
| package com.example.eugene.sqliteapp;    import android.database.sqlite.SQLiteOpenHelper;  import android.database.sqlite.SQLiteDatabase;  import android.content.Context;  import android.content.ContentValues;    public class DatabaseHelper extends SQLiteOpenHelper {      private static final String DATABASE\_NAME = "userstore.db"; // название бд      private static final int SCHEMA = 1; // версия базы данных      static final String TABLE = "users"; // название таблицы в бд      // названия столбцов      public static final String COLUMN\_ID = "\_id";      public static final String COLUMN\_NAME = "name";      public static final String COLUMN\_YEAR = "year";        public DatabaseHelper(Context context) {          super(context, DATABASE\_NAME, null, SCHEMA);      }        @Override      public void onCreate(SQLiteDatabase db) {            db.execSQL("CREATE TABLE users (" + COLUMN\_ID                  + " INTEGER PRIMARY KEY AUTOINCREMENT," + COLUMN\_NAME                  + " TEXT, " + COLUMN\_YEAR + " INTEGER);");          // добавление начальных данных          db.execSQL("INSERT INTO "+ TABLE +" (" + COLUMN\_NAME                  + ", " + COLUMN\_YEAR  + ") VALUES ('Том Смит', 1981);");      }      @Override      public void onUpgrade(SQLiteDatabase db, int oldVersion,  int newVersion) {          db.execSQL("DROP TABLE IF EXISTS "+TABLE);          onCreate(db);      }  } |

Если база данных отсутствует или ее версия (которая задается в переменной SCHEMA) выше текущей, то срабатывает метод onCreate().

Для выполнения запросов к базе данных нам потребуется объект **SQLiteDatabase**, который представляет базу данных. Метод onCreate()получает в качестве параметра базу данных приложения.

Для выполнения запросов к SQLite используется метод **execSQL()**. Он принимает sql-выражение CREATE TABLE, которое создает таблицу. Здесь также при необходимости мы можем выполнить и другие запросы, например, добавить какие-либо начальные данные. Так, в данном случае с помощью того же метода и выражения sql INSERT добавляется один объект в таблицу.

В методе onUpgrade() происходит обновление схемы БД. В данном случае для примера использован примитивный поход с удалением предыдущей базы данных с помощью sql-выражения DROP и последующим ее созданием. Но в реальности если вам будет необходимо сохранить данные, этот метод может включать более сложную логику - добавления новых столбцов, удаление ненужных, добавление дополнительных данных и т.д.

Далее определим в файле **activity\_main.xml** следующую разметку:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"      android:orientation="vertical"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent">      <TextView          android:id="@+id/header"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:textSize="18dp"/>      <ListView          android:id="@+id/list"          android:layout\_width="match\_parent"          android:layout\_height="match\_parent" />  </LinearLayout> |

Здесь определен список ListView, для отображения полученных данных, с заголовком, который будет выводить число полученных объектов.

И изменим код класса **MainActivity** следующим образом:

|  |
| --- |
| package com.example.eugene.sqliteapp;    import android.database.Cursor;  import android.database.sqlite.SQLiteDatabase;  import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.view.View;  import android.widget.AdapterView;  import android.widget.ListView;  import android.widget.SimpleCursorAdapter;  import android.widget.TextView;    public class MainActivity extends AppCompatActivity {        ListView userList;      TextView header;      DatabaseHelper databaseHelper;      SQLiteDatabase db;      Cursor userCursor;      SimpleCursorAdapter userAdapter;        @Override      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);            header = (TextView)findViewById(R.id.header);          userList = (ListView)findViewById(R.id.list);            databaseHelper = new DatabaseHelper(getApplicationContext());      }      @Override      public void onResume() {          super.onResume();          // открываем подключение          db = databaseHelper.getReadableDatabase();            //получаем данные из бд в виде курсора          userCursor =  db.rawQuery("select \* from "+ DatabaseHelper.TABLE, null);          // определяем, какие столбцы из курсора будут выводиться в ListView          String[] headers = new String[] {DatabaseHelper.COLUMN\_NAME, DatabaseHelper.COLUMN\_YEAR};          // создаем адаптер, передаем в него курсор          userAdapter = new SimpleCursorAdapter(this, android.R.layout.two\_line\_list\_item,                  userCursor, headers, new int[]{android.R.id.text1, android.R.id.text2}, 0);          header.setText("Найдено элементов: " + String.valueOf(userCursor.getCount()));          userList.setAdapter(userAdapter);      }        @Override      public void onDestroy(){          super.onDestroy();          // Закрываем подключение и курсор          db.close();          userCursor.close();      }  } |

В методе onCreate() происходит создание объекта SQLiteOpenHelper. Сама инициализация объектов для работы с базой данных происходит в методе onResume(), который срабатывает после метода onCreate().

Чтобы получить объект базы данных, надо использовать метод getReadableDatabase() (получение базы данных для чтения) илиgetWritableDatabase(). Так как в данном случае мы будет только считывать данные из бд, то воспользуемся первым методом:

|  |  |
| --- | --- |
| 1 | db = sqlHelper.getReadableDatabase(); |

**Получение данных и Cursor**

Android предоставляет различные способы для осуществления запросов к объекту SQLiteDatabase. В большинстве случаев мы можем применять метод **rawQuery()**, который принимает два параметра: SQL-выражение SELECT и дополнительный параметр, задающий параметры запроса.

После выполнения запроса rawQuery() возвращает объект **Cursor**, который хранит результат выполнения SQL-запроса:

|  |  |
| --- | --- |
| 1 | userCursor =  db.rawQuery("select \* from "+ DatabaseHelper.TABLE, null); |

Класс Cursor предлагает ряд методов для управления выборкой, в частности:

* getCount(): получает количество извлеченных из базы данных объектов
* Методы moveToFirst() и moveToNext() позволяют переходить к первому и к следующему элементам выборки. Метод isAfterLast()позволяет проверить, достигнут ли конец выборки.
* Методы get\*(columnIndex) (например, getLong(), getString()) позволяют по индексу столбца обратиться к данному столбцу текущей строки

**CursorAdapter**

Дополнительно для управления курсором в Android имеется класс CursorAdapter. Он позволяет адаптировать полученный с помощью курсора набор к отображению в списковых элементах наподобие ListView. Как правило, при работе с курсором используется подкласс CursorAdapter - **SimpleCursorAdapter**. Хотя можно использовать и другие адаптеры, типа ArrayAdapter.

|  |  |
| --- | --- |
| 1  2  3 | userAdapter = new SimpleCursorAdapter(this, android.R.layout.two\_line\_list\_item,                  userCursor, headers, new int[]{android.R.id.text1, android.R.id.text2}, 0);  userList.setAdapter(userAdapter); |

Конструктор класса SimpleCursorAdapter принимает шесть параметров:

1. Первым параметром выступает контекст, с которым ассоциируется адаптер, например, текущая activity
2. Второй параметр - ресурс разметки интерфейса, который будет использоваться для отображения результатов выборки
3. Третий параметр - курсор
4. Четвертый параметр - список столбцов из выборки, которые будут отображаться в разметке интерфейса
5. Пятый параметр - элементы внутри ресурса разметки, которые будут отображать значения столбцов из четвертого параметра
6. Шестой параметр - флаги, задающие поведения адаптера

При использовании CursorAdapter и его подклассов следует учитывать, что выборка курсора должна включать целочисленный столбец с названием **\_id**, который должен быть уникальным для каждого элемента выборки. Значение этого столбца при нажатии на элемент списка затем передается в метод обработки onListItemClick(), благодаря чему мы можем по id идентифицировать нажатый элемент.

В данном случае у нас первый столбец как раз называется "\_id".

После завершения работу курсор должен быть закрыт методом close()

И также надо учитывать, что если мы используем курсор в SimpleCursorAdapter, то мы не можем использовать метод close(), пока не завершим использование SimpleCursorAdapter. Поэтому метод cursor более предпочтительно вызывать в методе onDestroy() фрагмента или activity.

И если мы запустим приложение, то увидим список из одного добавленного элемента:

