**SimpleCursorAdapter и получение данных**

В Android имеется встроенная поддержка одной из распространенных систем управления базами данных - SQLite. Для этого в пакете **android.database.sqlite** определен набор классов, которые позволяют работать с базами данных SQLite. И каждое приложение может создать свою базу данных.

Чтобы использовать SQLite в Android, надо создать базу данных с помощью выражение на языке SQL. После этого база данных будет храниться в каталоге приложения по пути:

DATA/data/[Название\_приложения]/databases/[Название\_файла\_базы\_данных]

ОС Android по умолчанию уже содержит ряд встроенных бад SQLite, которые используются стандартными программами - для списка контактов, для хранения фотографий с камеры, музыкальных альбомов и т.д.

Итак, создадим новый проект.

Для работы с базами данных SQLite в Android используется класс **SQLiteOpenHelper**. И в приложении нам надо будет создать класса-наследника 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 новый класс DatabaseHelper:

|  |  |
| --- | --- |
| 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 | package com.example.eugene.sqlitedbapp;    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, которое затем выполняется. Логично в этом методе выполнить 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 layout для activity следующую разметку:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?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 MainActivity следующим образом:

|  |  |
| --- | --- |
| 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 | package com.example.eugene.sqlitedbapp;    import android.content.Intent;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;    import android.view.View;  import android.widget.AdapterView;  import android.widget.TextView;  import android.widget.ListView;  import android.database.sqlite.SQLiteDatabase;  import android.database.Cursor;  import android.widget.SimpleCursorAdapter;    public class MainActivity extends ActionBarActivity {        ListView mList;      TextView header;      DatabaseHelper sqlHelper;      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);          mList = (ListView)findViewById(R.id.list);          mList.setOnItemClickListener(new AdapterView.OnItemClickListener() {              @Override              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {                }          });            sqlHelper = new DatabaseHelper(getApplicationContext());      }      @Override      public void onResume() {          super.onResume();          // открываем подключение           db = sqlHelper.getReadableDatabase();            //получаем данные из бд          userCursor =  db.rawQuery("select \* from "+ DatabaseHelper.TABLE, null);          String[] headers = new String[] {DatabaseHelper.COLUMN\_NAME, DatabaseHelper.COLUMN\_YEAR};          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()));          mList.setAdapter(userAdapter);      }        @Override      public void onDestroy(){          super.onDestroy();          // Закрываем подключения           db.close();           userCursor.close();      }        @Override      public boolean onCreateOptionsMenu(Menu menu) {          getMenuInflater().inflate(R.menu.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);      }  } |

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) или getWritableDatabase(). 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);  mList.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 cursorболее предпочтительно вызывать в методе onDestroy() фрагмента или activity.

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

