Работа с базой данных SQLite

ОС Андроид также позволяет хранить данные приложения в базе данных **SQLite**. Мы рассмотрим простейший пример создания базы данных, а также работу с ее данными (чтение, вставка, обновление и удаление записей).

Нам понадобиться создать:  
1. **MainActivity.java** (здесь будет происходить основная логика)  
2. **DbHelper.java** (вспомогательный класс, отвечающий за создание базы данных и ее обновление при изменении версии)  
3. **Db.java** (класс, в котором мы поместим все основные методы работы с базой данных. Из Activity мы будем лишь вызывать эти методы)  
4. **Friend.java** (класс модели данных)  
5. **activity\_main.xml** (разметка с пустым [LinearLayout](http://androiddocs.ru/linearlayout/) — все данные будем выводить только в лог)

Итак, создадим класс модели Friend. Здесь будет всего три свойства — **id**, **имя (name)** и **email**. Также добавим «геттеры» и «сеттеры».

**Friend.java**

|  |
| --- |
| **package** ru.androiddocs.sqlite.model;    **public** **class** Friend {    **private** Integer id;  **private** String name;  **private** String email;    Friend(){}    **public** Friend(Integer id, String name, String email) {  **this**.id = id;  **this**.name = name;  **this**.email = email;  }    **public** Integer getId() {  **return** id;  }    **public** **void** setId(Integer id) {  **this**.id = id;  }    **public** String getName() {  **return** name;  }    **public** **void** setName(String name) {  **this**.name = name;  }    **public** String getEmail() {  **return** email;  }  **public** **void** setEmail(String email) {  **this**.email = email;  }  } |

Класс-помощник **DbHelper** отвечает за создание и обновление базы данных. Здесь мы помещаем два ключевых метода **onCreate()** и **onUpgrade()**. Первый срабатывает при первом создании базы данных. Если вдруг версия базы данных меняется (константа **DATABASE\_VERSION**), то будет запущен метод onUpgrade(). В нашем случае при обновлении базы мы просто удаляем текущую базу данных и вызываем метод **onCreate()** для создания новой.

Обратите внимание, что также мы в константы поместили имя таблицы и наименования колонок. В методе onCreate() мы запускаем выполнение запроса на создание таблицы **execSQL()**, а затем делаем вставку трех строк с именем и email. Для передачи вставляемых данных используется коллекция **ContentValues**, где в качестве ключа выступает имя колонки, а значение — это значение для данного поля.

**DbHelper.java**

|  |
| --- |
| **package** ru.androiddocs.sqlite;    **import** android.content.ContentValues;  **import** android.content.Context;  **import** android.database.sqlite.SQLiteDatabase;  **import** android.database.sqlite.SQLiteOpenHelper;  **import** android.util.Log;    **public** **class** DbHelper **extends** SQLiteOpenHelper {    **private** **static** **final** String LOG\_TAG = "my\_tag";    **public** **static** **final** String TABLE\_NAME = "friends";    **public** **static** **final** String KEY\_ID = "\_id";  **public** **static** **final** String KEY\_NAME = "name";  **public** **static** **final** String KEY\_EMAIL = "email";    **private** **static** **final** String DATABASE\_NAME = "friendsDB";  **private** **static** **final** **int** DATABASE\_VERSION = 1;    **public** DbHelper(Context context) {  **super**(context, DATABASE\_NAME, **null**, DATABASE\_VERSION);  }    @Override  **public** **void** onCreate(SQLiteDatabase db) {    db.execSQL("create table " + TABLE\_NAME +" ("  + KEY\_ID + " integer primary key autoincrement,"  + KEY\_NAME + " text,"  + KEY\_EMAIL + " text" + ");");    ContentValues cv = **new** ContentValues();    cv.put(KEY\_NAME, "Igor");  cv.put(KEY\_EMAIL, "email1@email.com");  db.insert(TABLE\_NAME, **null**, cv);    cv.put(KEY\_NAME, "Dima");  cv.put(KEY\_EMAIL, "email2@email.com");  db.insert(TABLE\_NAME, **null**, cv);    cv.put(KEY\_NAME, "Alex");  cv.put(KEY\_EMAIL, "email3@email.com");  db.insert(TABLE\_NAME, **null**, cv);  }    @Override  **public** **void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);  **this**.onCreate(db);  }    } |

В сети встречаются примеры, где в класс DbHelper помещаются также другие методы для работы с базой данных. Однако, это некорректно, поскольку согласно документации есть такое определение данного класса:

**A helper class to manage database creation and version management.**

Нужные методы для работы с данными мы поместим в новый класс, который назовем **Db**.

**Db.java**

|  |
| --- |
| **package** ru.androiddocs.sqlite;    **import** java.util.ArrayList;  **import** java.util.List;    **import** ru.androiddocs.sqlite.model.Friend;    **import** android.content.ContentValues;  **import** android.content.Context;  **import** android.database.Cursor;  **import** android.database.sqlite.SQLiteDatabase;  **import** android.util.Log;    **public** **class** Db {    **private** **static** **final** String LOG\_TAG = "my\_tag";  DbHelper dbHelper;  Context context;  Cursor cursor;  SQLiteDatabase db;  List<Friend> mFriendsList;    **public** Db(Context context) {  **this**.context = context;  dbHelper = **new** DbHelper(context);  }  *// возвращает количество записей в таблице*  **public** **int** getItemCount() {    db = dbHelper.getReadableDatabase();    cursor = db.query(DbHelper.TABLE\_NAME, **null**, **null**, **null**, **null**, **null**, **null**);  **int** cnt = cursor.getCount();  cursor.close();    **return** cnt;  }  *// метод для обновления email*  **public** **void** updateEmail(String name, String newEmail){  db = dbHelper.getWritableDatabase();  ContentValues cv = **new** ContentValues();  cv.put(DbHelper.KEY\_EMAIL, "newemail@newemail.com");  String[] args = **new** String[]{name};  db.update(DbHelper.TABLE\_NAME, cv, "name = ?", args);  }  *// метод для удаления строки по id*  **public** **void** deleteItem(**int** id) {  db = dbHelper.getWritableDatabase();  db.delete(DbHelper.TABLE\_NAME, DbHelper.KEY\_ID + "=" + id, **null**);  }  *// метод возвращающий коллекцию всех данных*  **public** List<Friend> getFriends() {  cursor = db.query(DbHelper.TABLE\_NAME, **null**, **null**, **null**, **null**, **null**, **null**);  mFriendsList = **new** ArrayList<Friend>();    **if** (cursor.moveToFirst()) {    **int** idColInd = cursor.getColumnIndex(DbHelper.KEY\_ID);  **int** nameColInd = cursor.getColumnIndex(DbHelper.KEY\_NAME);  **int** emailColInd = cursor.getColumnIndex(DbHelper.KEY\_EMAIL);    **do** {  Friend friend = **new** Friend(cursor.getInt(idColInd),  cursor.getString(nameColInd), cursor.getString(emailColInd));  mFriendsList.add(friend);  } **while** (cursor.moveToNext());    } **else** {  Log.d(LOG\_TAG, "В базе нет данных!");  }    cursor.close();    **return** mFriendsList;    }  *// здесь закрываем все соединения с базой и класс-помощник*  **public** **void** close() {  dbHelper.close();  db.close();  }    } |

Пустой layout.

**activity\_main.xml**

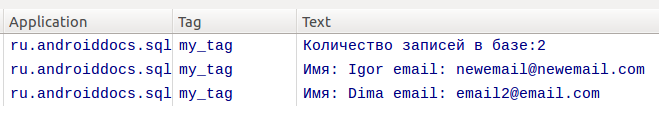
|  |
| --- |
| <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:id="@+id/container"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  tools:context="ru.androiddocs.sqlite.MainActivity">    </LinearLayout> |

Класс MainActivity, в котором помещаем основную логику.

**MainActivity.java**

|  |
| --- |
| **package** ru.androiddocs.sqlite;    **import** java.util.List;    **import** ru.androiddocs.sqlite.model.Friend;  **import** android.support.v7.app.ActionBarActivity;    **import** android.os.Bundle;  **import** android.util.Log;    **public** **class** MainActivity **extends** ActionBarActivity {    **private** **static** **final** String LOG\_TAG = "my\_tag";  Db db;    @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);    db = **new** Db(**this**);  *// получаем количество записей в базе перед изменениями*  **int** mCount = db.getItemCount();  Log.d(LOG\_TAG, "Количество записей в базе:" + mCount);    *// обновляем запись с id = 1 (меняем email)*  db.updateEmail("Igor", "newemail@newemail.com");    *// удаляем запись с id = 3*  db.deleteItem(3);    *// выводим все имеющиеся записи в лог*  List<Friend> friends = db.getFriends();  **for** (Friend friend : friends) {  Log.d(LOG\_TAG, "Имя: " + friend.getName() + " email: " + friend.getEmail());  }    db.close();  }  } |

Как видно из кода, мы вызываем различные методы класса Db для изменения данных в таблице Friends. В самом конце мы достаем все текущие данные и выводим их в лог. Думаю, будет понятно, почему получился именно такой результат.



**Использование интерфейса BaseColumns**

Описание структуры таблиц базы данных также удобно делать с помощью внутреннего класса, реализующего интерфейс **BaseColumns**. В этом случае, колонку **\_ID** описывать не нужно, т.к. она уже есть в «родителе». Это может быть описано примерно так:

|  |
| --- |
| **package** ru.androiddocs.sqlite;    **import** android.content.ContentValues;  **import** android.content.Context;  **import** android.database.sqlite.SQLiteDatabase;  **import** android.database.sqlite.SQLiteOpenHelper;  **import** android.provider.BaseColumns;      **public** **class** DbHelper **extends** SQLiteOpenHelper {    **public** **static** **final** String TABLE\_NAME = "friends";    **private** **static** **final** String DATABASE\_NAME = "friendsDB";  **private** **static** **final** **int** DATABASE\_VERSION = 1;    **public** DbHelper(Context context) {  **super**(context, DATABASE\_NAME, **null**, DATABASE\_VERSION);  }    **public** **class** FriendsContract **implements** BaseColumns {  **public** **static** **final** String KEY\_NAME = "name";  **public** **static** **final** String KEY\_EMAIL = "email";  }    **private** **static** **final** String CREATE\_FRIENDS\_TABLE\_SQL = "create table " + TABLE\_NAME +" ("  + FriendsContract.\_ID + " integer primary key autoincrement,"  + FriendsContract.KEY\_NAME + " text,"  + FriendsContract.KEY\_EMAIL + " text" + ");";    @Override  **public** **void** onCreate(SQLiteDatabase db) {    db.execSQL(CREATE\_FRIENDS\_TABLE\_SQL);    ContentValues cv = **new** ContentValues();    cv.put(FriendsContract.KEY\_NAME, "Igor");  cv.put(FriendsContract.KEY\_EMAIL, "email1@email.com");  db.insert(TABLE\_NAME, **null**, cv);  }    @Override  **public** **void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);  **this**.onCreate(db);  }    } |