**Заполнить таблицу в SQLite**

Иногда требуется сразу наполнить таблицу базы данных некоторыми данными. Если записей один-два и обчёлся, то добавить в коде несколько лишних переменных не сложно. Но если записей побольше, то проще автоматизировать процесс и заполнить таблицу через готовые ресурсы.

**Наполняем таблицы данными через массив строк**

Рассмотрим пример наполнения таблицы через строковый массив из ресурсов. В файле **res/values/strings.xml** определим массив строк, которые будут содержать имена котов:

<string-array name="catlist">

<item>Барсик</item>

<item>Мурзик</item>

<item>Васька</item>

<item>Рыжик</item>

</string-array>

Теперь создадим класс-обёртку для создания базы:

package ru.alexanderklimov.catbase;

import android.content.ContentValues;

import android.content.Context;

import android.content.res.Resources;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.util.Log;

public class DatabaseHelper extends SQLiteOpenHelper {

private final Context fContext;

private static final String DATABASE\_NAME = "cat\_database.db";

public static final String TABLE\_NAME = "cattable";

DatabaseHelper(Context context) {

super(context, DATABASE\_NAME, null, 1);

fContext = context;

}

@Override

public void onCreate(SQLiteDatabase db) {

// TODO Auto-generated method stub

db.execSQL("CREATE TABLE " + TABLE\_NAME + " ("

+ "\_id INTEGER PRIMARY KEY," + "title TEXT" + ");");

// Добавляем записи в таблицу

ContentValues values = new ContentValues();

// Получим массив строк из ресурсов

Resources res = fContext.getResources();

String[] cattable\_records = res.getStringArray(R.array.catlist\_v1);

// проходим через массив и вставляем записи в таблицу

int length = cattable\_records.length;

for (int i = 0; i &lt; length; i++) {

values.put("title", cattable\_records[i]);

db.insert(TABLE\_NAME, null, values);

}

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

// TODO Auto-generated method stub

Log.w("TestBase", "Upgrading database from version " + oldVersion

+ " to " + newVersion + ", which will destroy all old data");

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);

onCreate(db);

}

}

В методе **onCreate()** мы создаём нужную таблицу и заполняем её данными из ресурсов. Осталось только подключить класс в основной активности:

setContentView(R.layout.activity\_test);

// Инициализируем наш класс-обёртку

DatabaseHelper dbh = new DatabaseHelper(this);

// База нам нужна для записи и чтения

SQLiteDatabase sqdb = dbh.getWritableDatabase();

// закрываем соединения с базой данных

sqdb.close();

dbh.close();

Всё, теперь у нас в базе есть таблица с готовыми именами котов.

**Сложное наполнение таблицы данными через XML**

Предыдущий пример был слишком прост и годится для простой таблицы. Для более сложной структуры можно создать специальный XML-файл, а затем распарсить его.

Создадим в папке **res/xml/** новый файл **cats\_records.xml** следующего содержания:

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

<cats>

<record title="Барсик" color="чёрный" />

<record title="Мурзик" color="белый" />

<record title="Рыжик" color="рыжий" />

<record title="Васька" color="серый" />

</cats>

В данном примере у каждого кота есть имя и цвет его шубки. При желании можно добавить дополнительные атрибуты.

Модифицируем класс-обёртку из первого примера:

package ru.alexanderklimov.catbase;

import java.io.IOException;

import org.xmlpull.v1.XmlPullParser;

import org.xmlpull.v1.XmlPullParserException;

import android.content.ContentValues;

import android.content.Context;

import android.content.res.Resources;

import android.content.res.XmlResourceParser;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.util.Log;

public class DatabaseHelper extends SQLiteOpenHelper {

private final Context fContext;

private static final String DATABASE\_NAME = "colorcat\_database.db";

public static final String TABLE\_NAME = "cattable";

DatabaseHelper(Context context) {

super(context, DATABASE\_NAME, null, 1);

fContext = context;

}

@Override

public void onCreate(SQLiteDatabase db) {

// TODO Auto-generated method stub

db.execSQL("CREATE TABLE " + TABLE\_NAME + " ("

+ "\_id INTEGER PRIMARY KEY, " + "title TEXT, " + "color TEXT"

+ ");");

// Добавляем записи в таблицу

ContentValues values = new ContentValues();

// Получим файл из ресурсов

Resources res = fContext.getResources();

// Открываем xml-файл

XmlResourceParser \_xml = res.getXml(R.xml.cats\_records);

try {

// Ищем конец документа

int eventType = \_xml.getEventType();

while (eventType != XmlPullParser.END\_DOCUMENT) {

// Ищем теги record

if ((eventType == XmlPullParser.START\_TAG)

&& (\_xml.getName().equals("record"))) {

// Тег Record найден, теперь получим его атрибуты и

// вставляем в таблицу

String title = \_xml.getAttributeValue(0);

String color = \_xml.getAttributeValue(1);

values.put("title", title);

values.put("color", color);

db.insert(TABLE\_NAME, null, values);

}

eventType = \_xml.next();

}

}

// Catch errors

catch (XmlPullParserException e) {

Log.e("Test", e.getMessage(), e);

} catch (IOException e) {

Log.e("Test", e.getMessage(), e);

} finally {

// Close the xml file

\_xml.close();

}

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

// TODO Auto-generated method stub

Log.w("TestBase", "Upgrading database from version " + oldVersion

+ " to " + newVersion + ", which will destroy all old data");

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);

onCreate(db);

}

}

Подключаем класс и при запуске приложения в базе будет готовая таблица в следующем виде:

|  |  |  |
| --- | --- | --- |
| **\_id** | **title** | **color** |
| 1 | Барсик | чёрный |
| 2 | Мурзик | белый |
| 3 | Рыжик | рыжий |
| 4 | Васька | серый |