**Список дел с использованием контент-провайдера**

Переделаем приложение «Список дел» с использованием контент-провайдера.

Во-первых, перенесём код создания таблицы в отдельный класс. Хотя у нас используется одна таблица, лучше сразу привыкать к правильному коду. Создадим новый класс **TodoTable** и перенесём часть кода из класса **TodoDataBase**:

package ru.alexanderklimov.todo;

import android.database.sqlite.SQLiteDatabase;

import android.util.Log;

public class TodoTable {

// Таблица todos

public static final String TABLE\_TODO = "todos";

// поля таблицы

public static final String COLUMN\_ID = "\_id";

public static final String COLUMN\_CATEGORY = "category";

public static final String COLUMN\_SUMMARY = "summary";

public static final String COLUMN\_DESCRIPTION = "description";

// SQL-скрипт создания таблицы

private static final String DATABASE\_CREATE = "create table " + TABLE\_TODO

+ "(" + COLUMN\_ID + " integer primary key autoincrement, "

+ COLUMN\_CATEGORY + " text not null, " + COLUMN\_SUMMARY

+ " text not null," + COLUMN\_DESCRIPTION + " text not null" + ");";

public static void onCreate(SQLiteDatabase database) {

database.execSQL(DATABASE\_CREATE);

}

public static void onUpgrade(SQLiteDatabase database, int oldVersion,

int newVersion) {

Log.w(TodoTable.class.getName(), "Upgrading database from version "

+ oldVersion + " to " + newVersion

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

database.execSQL("DROP TABLE IF EXISTS " + TABLE\_TODO);

onCreate(database);

}

}

Код в классе **ToDoDatabase** уже "похудел". Однако мы ещё сократим его содержание, убрав методы CRUD, которые будут теперь находится в классе контент-провайдера. Останется совсем немного кода.

package ru.alexanderklimov.todo;

import android.content.Context;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

public class ToDoDatabase extends SQLiteOpenHelper {

private static final String DATABASE\_NAME = "todo\_app.db";

private static final int DATABASE\_VERSION = 1;

public ToDoDatabase(Context context) {

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

}

@Override

public void onCreate(SQLiteDatabase db) {

// TODO Auto-generated method stub

TodoTable.onCreate(db);

}

@Override

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

// TODO Auto-generated method stub

TodoTable.onUpgrade(db, oldVersion, newVersion);

}

}

Прям душа радуется. Всё такое компактное и аккуратное.

Но рано радоваться. Сейчас будем создавать контент-провайдер **TodoContentProvider**:

package ru.alexanderklimov.todoold;

import java.util.Arrays;

import java.util.HashSet;

import android.content.ContentProvider;

import android.content.ContentResolver;

import android.content.ContentValues;

import android.content.UriMatcher;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteQueryBuilder;

import android.net.Uri;

import android.text.TextUtils;

public class TodoContentProvider extends ContentProvider {

// database

private ToDoDatabase database;

// used for the UriMacher

private static final int TODOS = 10;

private static final int TODO\_ID = 20;

private static final String AUTHORITY = "ru.alexanderklimov.android.todos.contentprovider";

private static final String BASE\_PATH = "todos";

public static final Uri CONTENT\_URI = Uri.parse("content://" + AUTHORITY

+ "/" + BASE\_PATH);

public static final String CONTENT\_TYPE = ContentResolver.CURSOR\_DIR\_BASE\_TYPE

+ "/todos";

public static final String CONTENT\_ITEM\_TYPE = ContentResolver.CURSOR\_ITEM\_BASE\_TYPE

+ "/todo";

private static final UriMatcher sURIMatcher = new UriMatcher(

UriMatcher.NO\_MATCH);

static {

sURIMatcher.addURI(AUTHORITY, BASE\_PATH, TODOS);

sURIMatcher.addURI(AUTHORITY, BASE\_PATH + "/#", TODO\_ID);

}

@Override

public int delete(Uri uri, String selection, String[] selectionArgs) {

// TODO Auto-generated method stub

int uriType = sURIMatcher.match(uri);

SQLiteDatabase sqlDB = database.getWritableDatabase();

int rowsDeleted = 0;

switch (uriType) {

case TODOS:

rowsDeleted = sqlDB.delete(TodoTable.TABLE\_TODO, selection,

selectionArgs);

break;

case TODO\_ID:

String id = uri.getLastPathSegment();

if (TextUtils.isEmpty(selection)) {

rowsDeleted = sqlDB.delete(TodoTable.TABLE\_TODO,

TodoTable.COLUMN\_ID + "=" + id, null);

} else {

rowsDeleted = sqlDB.delete(TodoTable.TABLE\_TODO,

TodoTable.COLUMN\_ID + "=" + id + " and " + selection,

selectionArgs);

}

break;

default:

throw new IllegalArgumentException("Unknown URI: " + uri);

}

getContext().getContentResolver().notifyChange(uri, null);

return rowsDeleted;

}

@Override

public String getType(Uri arg0) {

// TODO Auto-generated method stub

return null;

}

@Override

public Uri insert(Uri uri, ContentValues values) {

// TODO Auto-generated method stub

int uriType = sURIMatcher.match(uri);

SQLiteDatabase sqlDB = database.getWritableDatabase();

//int rowsDeleted = 0;

long id = 0;

switch (uriType) {

case TODOS:

id = sqlDB.insert(TodoTable.TABLE\_TODO, null, values);

break;

default:

throw new IllegalArgumentException("Unknown URI: " + uri);

}

getContext().getContentResolver().notifyChange(uri, null);

return Uri.parse(BASE\_PATH + "/" + id);

}

@Override

public boolean onCreate() {

// TODO Auto-generated method stub

database = new ToDoDatabase(getContext());

return false;

}

@Override

public Cursor query(Uri uri, String[] projection, String selection,

String[] selectionArgs, String sortOrder) {

// TODO Auto-generated method stub

// Uisng SQLiteQueryBuilder instead of query() method

SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();

// check if the caller has requested a column which does not exists

checkColumns(projection);

// Set the table

queryBuilder.setTables(TodoTable.TABLE\_TODO);

int uriType = sURIMatcher.match(uri);

switch (uriType) {

case TODOS:

break;

case TODO\_ID:

// adding the ID to the original query

queryBuilder.appendWhere(TodoTable.COLUMN\_ID + "="

+ uri.getLastPathSegment());

break;

default:

throw new IllegalArgumentException("Unknown URI: " + uri);

}

SQLiteDatabase db = database.getWritableDatabase();

Cursor cursor = queryBuilder.query(db, projection, selection,

selectionArgs, null, null, sortOrder);

// make sure that potential listeners are getting notified

cursor.setNotificationUri(getContext().getContentResolver(), uri);

return cursor;

}

@Override

public int update(Uri uri, ContentValues values, String selection,

String[] selectionArgs) {

// TODO Auto-generated method stub

int uriType = sURIMatcher.match(uri);

SQLiteDatabase sqlDB = database.getWritableDatabase();

int rowsUpdated = 0;

switch (uriType) {

case TODOS:

rowsUpdated = sqlDB.update(TodoTable.TABLE\_TODO, values, selection,

selectionArgs);

break;

case TODO\_ID:

String id = uri.getLastPathSegment();

if (TextUtils.isEmpty(selection)) {

rowsUpdated = sqlDB.update(TodoTable.TABLE\_TODO, values,

TodoTable.COLUMN\_ID + "=" + id, null);

} else {

rowsUpdated = sqlDB.update(TodoTable.TABLE\_TODO, values,

TodoTable.COLUMN\_ID + "=" + id + " and " + selection,

selectionArgs);

}

break;

default:

throw new IllegalArgumentException("Unknown URI: " + uri);

}

getContext().getContentResolver().notifyChange(uri, null);

return rowsUpdated;

}

private void checkColumns(String[] projection) {

String[] available = { TodoTable.COLUMN\_CATEGORY,

TodoTable.COLUMN\_SUMMARY, TodoTable.COLUMN\_DESCRIPTION,

TodoTable.COLUMN\_ID };

if (projection != null) {

HashSet<String> requestedColumns = new HashSet<String>(

Arrays.asList(projection));

HashSet<String> availableColumns = new HashSet<String>(

Arrays.asList(available));

// check if all columns which are requested are available

if (!availableColumns.containsAll(requestedColumns)) {

throw new IllegalArgumentException(

"Unknown columns in projection");

}

}

}

}

Добавляем описание в манифест:

<provider

android:name="TodoContentProvider"

android:authorities="ru.alexanderklimov.android.todos.contentprovider"

android:exported="true" >

</provider>

Напомню, что у нас есть один пункт меню в ActionBar (res/menu/main.xml):

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item

android:id="@+id/insert"

android:showAsAction="always"

android:title="Добавить">

</item>

</menu>

В строковых ресурсах мы определили два вида задач - срочные и напоминание:

<string-array name="priorities">

<item>Срочные</item>

<item>Напомнить</item>

</string-array>

Разметка для элемента списка (res/layout/list\_row.xml):

<?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" >

<ImageView

android:id="@+id/icon"

android:layout\_width="30dp"

android:layout\_height="24dp"

android:layout\_marginLeft="4dp"

android:layout\_marginRight="8dp"

android:layout\_marginTop="8dp"

android:src="@drawable/ic\_launcher" >

</ImageView>

<TextView

android:id="@+id/label"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="6dp"

android:lines="1"

android:text=""

android:textColor="@color/black"

android:textSize="24sp" >

</TextView>

</LinearLayout>

Общая разметка первой активности (res/layout/activity\_main.xml):

<?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="match\_parent"

android:background="@color/listcolor"

android:orientation="vertical" >

<ListView

android:id="@android:id/list"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" >

</ListView>

<TextView

android:id="@android:id/empty"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/no\_todos" />

</LinearLayout>

Разметка для второй активности (res/layout/activity\_edit.xml):

<?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="match\_parent"

android:background="@color/listcolor"

android:orientation="vertical" >

<Spinner

android:id="@+id/category"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:entries="@array/priorities" >

</Spinner>

<LinearLayout

android:id="@+id/LinearLayout01"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" >

<EditText

android:id="@+id/todo\_edit\_summary"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:hint="Кратко"

android:imeOptions="actionNext" >

</EditText>

</LinearLayout>

<EditText

android:id="@+id/todo\_edit\_description"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_weight="1"

android:gravity="top"

android:hint="Описание" >

</EditText>

<Button

android:id="@+id/todo\_edit\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/todo\_confirm" >

</Button>

</LinearLayout>

Теперь код для основной активности. В отличие от прошлого урока, нам не нужно создавать собственный CursorLoader, а используем стандартный.

package ru.alexanderklimov.todoold;

import android.net.Uri;

import android.os.Bundle;

import android.app.ListActivity;

import android.app.LoaderManager.LoaderCallbacks;

import android.content.CursorLoader;

import android.content.Intent;

import android.content.Loader;

import android.database.Cursor;

import android.view.ContextMenu;

import android.view.ContextMenu.ContextMenuInfo;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.AdapterView.AdapterContextMenuInfo;

import android.widget.ListView;

import android.widget.SimpleCursorAdapter;

public class MainActivity extends ListActivity implements

LoaderCallbacks<Cursor> {

private static final int ACTIVITY\_CREATE = 0;

//private static final int ACTIVITY\_EDIT = 1;

private static final int DELETE\_ID = Menu.FIRST + 1;

private SimpleCursorAdapter adapter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

this.getListView().setDividerHeight(2);

registerForContextMenu(getListView());

fillData();

// dbHelper = new ToDoDatabase(this);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

// Реакция на выбор меню

@Override

public boolean onMenuItemSelected(int featureId, MenuItem item) {

switch (item.getItemId()) {

case R.id.insert:

createNewTask();

return true;

}

return super.onMenuItemSelected(featureId, item);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.insert:

createNewTask();

return true;

}

return super.onOptionsItemSelected(item);

}

@Override

public void onCreateContextMenu(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

super.onCreateContextMenu(menu, v, menuInfo);

menu.add(0, DELETE\_ID, 0, R.string.menu\_delete);

}

@Override

public boolean onContextItemSelected(MenuItem item) {

switch (item.getItemId()) {

case DELETE\_ID:

// получаем из пункта контекстного меню данные по пункту списка

AdapterContextMenuInfo info = (AdapterContextMenuInfo) item

.getMenuInfo();

Uri uri = Uri

.parse(TodoContentProvider.CONTENT\_URI + "/" + info.id);

getContentResolver().delete(uri, null, null);

fillData();

return true;

}

return super.onContextItemSelected(item);

}

private void createNewTask() {

Intent intent = new Intent(this, EditActivity.class);

startActivityForResult(intent, ACTIVITY\_CREATE);

}

private void fillData() {

// Fields from the database (projection)

// Must include the \_id column for the adapter to work

String[] from = new String[] { TodoTable.COLUMN\_SUMMARY };

// Fields on the UI to which we map

int[] to = new int[] { R.id.label };

getLoaderManager().initLoader(0, null, this);

adapter = new SimpleCursorAdapter(this, R.layout.list\_row, null, from,

to, 0);

setListAdapter(adapter);

}

@Override

protected void onListItemClick(ListView l, View v, int position, long id) {

super.onListItemClick(l, v, position, id);

Intent intent = new Intent(this, EditActivity.class);

Uri todoUri = Uri.parse(TodoContentProvider.CONTENT\_URI + "/" + id);

intent.putExtra(TodoContentProvider.CONTENT\_ITEM\_TYPE, todoUri);

startActivity(intent);

}

@Override

protected void onActivityResult(int requestCode, int resultCode,

Intent intent) {

super.onActivityResult(requestCode, resultCode, intent);

if (resultCode == RESULT\_OK) {

fillData();

}

}

@Override

public Loader<Cursor> onCreateLoader(int id, Bundle args) {

// TODO Auto-generated method stub

String[] projection = { TodoTable.COLUMN\_ID, TodoTable.COLUMN\_SUMMARY };

CursorLoader cursorLoader = new CursorLoader(this,

TodoContentProvider.CONTENT\_URI, projection, null, null, null);

return cursorLoader;

}

@Override

public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {

// TODO Auto-generated method stub

adapter.swapCursor(cursor);

}

@Override

public void onLoaderReset(Loader<Cursor> loader) {

// TODO Auto-generated method stub

adapter.swapCursor(null);

}

}

Код для второй активности.

package ru.alexanderklimov.todoold;

import android.net.Uri;

import android.os.Bundle;

import android.app.Activity;

import android.content.ContentValues;

import android.database.Cursor;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.Toast;

public class EditActivity extends Activity {

private Spinner mCategory;

private EditText mTitleText;

private EditText mBodyText;

private Uri todoUri;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit);

mCategory = (Spinner) findViewById(R.id.category);

mTitleText = (EditText) findViewById(R.id.todo\_edit\_summary);

mBodyText = (EditText) findViewById(R.id.todo\_edit\_description);

Button confirmButton = (Button) findViewById(R.id.todo\_edit\_button);

Bundle extras = getIntent().getExtras();

// check from the saved Instance

todoUri = (savedInstanceState == null) ? null

: (Uri) savedInstanceState

.getParcelable(TodoContentProvider.CONTENT\_ITEM\_TYPE);

// Or passed from the other activity

if (extras != null) {

todoUri = extras

.getParcelable(TodoContentProvider.CONTENT\_ITEM\_TYPE);

fillData(todoUri);

}

confirmButton.setOnClickListener(new View.OnClickListener() {

public void onClick(View view) {

if (TextUtils.isEmpty(mTitleText.getText().toString())) {

makeToast();

} else {

setResult(RESULT\_OK);

finish();

}

}

});

}

private void fillData(Uri uri) {

String[] projection = { TodoTable.COLUMN\_SUMMARY,

TodoTable.COLUMN\_DESCRIPTION, TodoTable.COLUMN\_CATEGORY };

Cursor cursor = getContentResolver().query(uri, projection, null, null,

null);

if (cursor != null) {

cursor.moveToFirst();

String category = cursor.getString(cursor

.getColumnIndexOrThrow(TodoTable.COLUMN\_CATEGORY));

for (int i = 0; i < mCategory.getCount(); i++) {

String s = (String) mCategory.getItemAtPosition(i);

if (s.equalsIgnoreCase(category)) {

mCategory.setSelection(i);

}

}

mTitleText.setText(cursor.getString(cursor

.getColumnIndexOrThrow(TodoTable.COLUMN\_SUMMARY)));

mBodyText.setText(cursor.getString(cursor

.getColumnIndexOrThrow(TodoTable.COLUMN\_DESCRIPTION)));

// always close the cursor

cursor.close();

}

}

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

saveState();

outState.putParcelable(TodoContentProvider.CONTENT\_ITEM\_TYPE, todoUri);

}

@Override

protected void onPause() {

super.onPause();

saveState();

}

private void saveState() {

String category = (String) mCategory.getSelectedItem();

String summary = mTitleText.getText().toString();

String description = mBodyText.getText().toString();

// only save if either summary or description

// is available

if (description.length() == 0 && summary.length() == 0) {

return;

}

ContentValues values = new ContentValues();

values.put(TodoTable.COLUMN\_CATEGORY, category);

values.put(TodoTable.COLUMN\_SUMMARY, summary);

values.put(TodoTable.COLUMN\_DESCRIPTION, description);

if (todoUri == null) {

// New todo

todoUri = getContentResolver().insert(

TodoContentProvider.CONTENT\_URI, values);

} else {

// Update todo

getContentResolver().update(todoUri, values, null, null);

}

}

private void makeToast() {

Toast.makeText(this, "Please maintain a summary", Toast.LENGTH\_LONG)

.show();

}

}

С одной стороны кода стало меньше, а с другой - разобраться стало сложнее. Тут надо собаку съесть, чтобы свободно писать такой код. Так как я не часто занимаюсь созданием приложений с базами данных с использованием собственных контент-провайдеров, то этот пример может пригодиться мне в качестве заготовки. А вам нравится создавать собственные контент-провайдеры?

Ещё раз напомню, что сам пример был написан на основе оригинальной статьи [Android SQLite database and content provider](http://www.vogella.com/articles/AndroidSQLite/article.html" \l "todo). Если у вас что-то не заработало, то обращайтесь к автору статьи. Мопед не мой.