

Android 4: databázový debilníček



Róbert Novotný
robert.novotny@upjs.sk
3. 3. 2014

Relačná databáza v Androide

- k dispozícii je SQLite
- embedded databáza
 - používaná i vo Firefoxu
- opensource, C kód, public domain
- rozoznáva základné dopyty v SQL

<http://www.sqlite.org/lang.html>

Základné triedy

SQLiteOpenHelper

- pomocná trieda pre správu životného cyklu databázy
 - vytváranie
 - upgrade

SQLiteDatabase

- metódy pre dopytovanie
- aktualizáciu
- transakcie

Cursor

- analógia ResultSet-u
- iteruje cez výsledok,
- ukazuje na aktuálny riadok
- umožňuje vyťahovať hodnoty

- továreň pre objekt databázy
- potrebné oddediť a prekryť:
 - **konštruktor**:
 - inicializujeme názov súboru databázy
 - a jej verziu
 - **onCreate**:
 - volaná pred úplne prvým vytvorením databázy
 - vložíme SQL direktívy CREATE pre tabuľky
 - **onUpgrade**:
 - aktualizácia štruktúry tabuliek

```
public class DatabaseOpenHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION = 2;
    private static final String DICTIONARY_TABLE_NAME = "dictionary";
    private static final String DICTIONARY_TABLE_CREATE =
        "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +
        KEY_WORD + " TEXT, " +
        KEY_DEFINITION + " TEXT);";
```

```
DatabaseOpenHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
```

```
public void onCreate(SQLiteDatabase db) {
    db.execSQL(DICTIONARY_TABLE_CREATE);
}
```

```
public void onUpgrade(SQLiteDatabase db) { /* nič */
}
```

- výkonná trieda pre dopyty
- `execSQL()`:
 - vykonávanie surových CREATE/DROP dopytov
- `rawQuery()`:
 - surové SQL dopyty
- `query()`, `insert()`, `update()`, `delete()`:
 - typovo bezpečené operácie

SQLiteOpenHelper

```
DatabaseOpenHelper dbHelper = new DatabaseOpenHelper(this);  
SQLiteDatabase db = getReadableDatabase();
```

- parametrom konštruktora je **kontext**
 - typicky inštancia aktivity, v rámci ktorej používame databázu
- Inštanciu databázy SQLiteDatabase získame
 - **getReadableDatabase()**: optimalizovaná pre čítanie
 - **getWritableDatabase()**: optimalizovaná pre čítanie i zápis

Kurzory a dopytovanie

```
DatabaseOpenHelper dbHelper = new DatabaseOpenHelper(this);  
Cursor cursor = getReadableDatabase().rawQuery(sql, params);
```

- kurzor sa správa ako **iterátor** cez tabuľku
 - filozofia `java.util.Scanner`
 - alebo `java.sql.ResultSet`
- metóda **`moveToNext()`** ho posúva dopredu
 - na ďalší riadok
- zároveň vracia **`true`**, ak je k dispozícii ďalší riadok

Kurzory a dopytovanie

- namiesto `rawQuery()` dopytujeme cez `query()`
- nemusíme riešiť lepenie reťazcov SQL dopytov
- korektne rieši úvodzovky
- cacheuje dopyty
- bohužiaľ úchylné API

```
Cursor cursor = database.query("dictionary",  
                                null, null, null, null, null, null)
```

Kurzory a adaptéry

- kurzory sa ľahko používajú s adaptérmí
- trieda `SimpleCursorAdapter`
- priamo mapuje stĺpce na položky zoznamu

<http://ics.upjs.sk/~novotnyr/blog/1046/android-zobrazovanie-udajov-z-sql-databazy>

Aktualizácia hodnôt

- klasické UPDATE/INSERT sú náchylné na chyby
 - absolútne neprehľadné pri 5+ stĺpcoch
 - escapovanie úvodzoviek
 - útoky SQL injection
- filozofia v Androide:
 - vytvoríme mapu z názvov riadkov do hodnôt
 - zavoláme príslušnú aktualizáciu metódu
- mapa: objekt typu **ContentValues**

<http://xkcd.com/327/>

Aktualizácia hodnôt: ContentValues

```
ContentValues values = new ContentValues();  
values.put("_ID", 1);  
values.put("NAME", "Android");  
  
database.insert("PHONES", null, values);
```

názov tabuľky

null column hack

Null Column Hack

- ak je mapa ContentValues **prázdna**, databáza nevie vložiť riadok do tabuľky
- v parametri **nullColumnHack** však vieme uviesť názov nullable stĺpca, do ktorého sa vloží NULL, ak je mapa prázdna

Aktualizácia hodnôt: update()

```
ContentValues values = new ContentValues();  
values.put("NAME", "Android");
```

```
String[] whereArgs = { String.valueOf(492) };  
database.update("PHONE", values, "_ID=?", whereArgs);
```

názov tabuľky

klauzula WHERE

zástupný znak

hodnoty, ktorými
sa nahradia
zástupné znaky

Konvencie :: tabuľky

- definujte konštantové triedy pre tabuľky
 - s názvami tabuliek a stĺpcov

```
public interface Database {  
    public static final String NAME = "tasks";  
    public static final int VERSION = 1;  
  
    public interface Task extends BaseColumns {  
        public static final String TABLE_NAME = "task";  
  
        public static final String DESCRIPTION = "description";  
        public static final String IS_DONE = "is_done";  
        public static final String DEADLINE = "deadline";  
    }  
}
```

Konvencie :: tabuľky

- primárny kľúč sa volá `_id`
 - malými písmenami!
- spolieha sa na to množstvo API
 - CursorAdaptery
 - zoznamy ListView
 - obsluhy vybraných tlačidiel
 - obsluhy kontextového menu

```
_id INTEGER PRIMARY KEY AUTOINCREMENT
```

- **null** má mnoho významov
 - defaultný / chýbajúci parameter
- konštanty pomôžu čitateľnosti
 - do triedy môžete staticky importovať

```
public interface CursorUtils {  
    public static final String[] NO_PROJECTION = null;  
    public static final String NO_SELECTION = null;  
    /*...*/  
}
```

```
return database.query(Database.Tasks.TABLE_NAME,  
    NO_PROJECTION,  
    NO_SELECTION, NO_SELECTION_ARGS,  
    NO_GROUP_BY, NO_HAVING, NO_SORT_ORDER);
```

Pozor na správu zdrojov

- **SimpleCursorAdapter** sa automaticky aktualizuje
 - pri `onPause()` / `onResume()`
 - ak sa do zoznamovej aktivity vrátíme z inej aktivity
- ak ho chceme ručne obnoviť:

```
adapter.setCursor().requery()
```

Pozor na správu zdrojov

- životný cyklus kurzora musíme spriahnuť so životným cyklom aktivity
- inak mrháme zdrojmi
- alebo môžu ostať otvorené po odstrelení aktivity

```
startManagingCursor(cursor);
```

Pozor na správu zdrojov

- ak databázu nepotrebujeme, zavríme ju
- obvykle zavretím vlastného `*OpenHelpera`
- zatváranie sa často deje v `onDestroy()` aktivity

```
@Override
protected void onDestroy() {
    databaseOpenHelper.close();
    super.onDestroy();
}
```

Pozor na správu zdrojov

- *OpenHelper postačí singleton
 - inak milión hlášok o nezatvorení databázy v aktivite

```
public class TaskDatabaseOpenHelper extends SQLiteOpenHelper {  
  
    private static TaskDatabaseOpenHelper INSTANCE;  
  
    public static TaskDatabaseOpenHelper getInstance(Context context) {  
        if(INSTANCE == null) {  
            INSTANCE = new TaskDatabaseOpenHelper(context.getApplicationContext());  
        }  
        return INSTANCE;  
  
        /.../  
    }  
}
```

Pozor na správu zdrojov

- alternatívne: všetky „DAO“ objekty do vlastnej podtriedy `android.app.Application`

<http://bit.ly/1olsLjU>

Problémy s API

- surová práca s kurzormi **je zastaralá** v Androide 4
- problém: práca s databázou beží v hlavnom vlákne
 - spomaľuje UI
 - podobne ako vo Swingu
- riešenie: **CursorLoadery** s **ContentProvidermi**

Megapríklad

<https://github.com/novotnyr/android-taskr-database>

Otázky?

