LUKÁŠ NOVÁK

KMI / TMA

Tvorba mobilních aplikací

6. seminář | 2.11.2017

ZS 2017/2018 | čtvrtek 13:15-15:45

OBSAH SEMINÁŘE

Ukládání dat

UKLÁDÁNÍ DAT

Jaké jsou možnosti?

- **SharedPreferences**
  - jednoduchá data
- **databáze SQLite**
  - relační data
- **assets/raw resources**
  - soubory jen pro čtení přibalené v APK
- **systém souborů**
  - klasické Java File I/O

SHAREDPREFERENCES

Persistence jednoduchých dat

- uložené dvojice **klíč:hodnota**
  - **klíč**: String
  - **hodnota**:
    - primitivní datový typ int, boolean, float, long
    - String, Set<String>
- uložené v souboru na disku
  - blackbox, systém poskytuje rozhraní

SHAREDPREFERENCES

ZÍSKÁNÍ INSTANCE

- rozhraní poskytuje **SharedPreferences**
- instanci **SharedPreferences** získáme pomocí metody:
  - **getSharedPreferences(String, int)**
    - můžeme mít více úložišť, většinou to však není potřeba
    - získáním instance neexistujícího úložiště, dojde k jeho vytvoření

SHAREDPREFERENCES

Získání instance

- **getSharedPreferences(String name, int mode)**
  - name: název úložiště dle našeho uvážení (doporučuje se **applicationId** + naše jméno
  - mode: **MODE_PRIVATE** (=0), ostatní konstanty jsou deprecated

SHAREDPREFERENCES

Ukládání dat

- probíhá pomocí **SharedPreferences.Editor**
- instanci editoru získáme zavoláním **edit()** na instanci SharedPreferences
- editor poskytuje metody:
  - **putBoolean**(key, value), **putInt**(key, value) a další pro vložení hodnot
  - **commit/apply** pro uložení změn do souboru
  - další pomocné metody (např. clear)

SHAREDPREFERENCES

ZÍSKÁNÍ DAT

- data z úložiště získáme přímo na instanci **SharedPreferences** pomocí metod:
  - int getInt(key, default),
  - String getString(key, default)
  - ...
- pokud je hodnota v úložišti, je vrácena, v opačném případě je vrácena hodnota **default**

DATABÁZE SQLITE

Co je to?

- relační databázový systém implementovaný v C
- téměř celý standard **SQL-92**
- 1 databáze = 1 soubor; rychlý přístup
- syntaxe základních CRUD operací velmi podobná MySQL/MariaDB, PostgreSQL

DATABÁZE SQLITE

SQLite v Androidu

- podpora v systému od počátku věků
- systém poskytuje vlastní rozhraní (žádné JDBC a podobně)
- pro práci s databází se používá třída **SQLiteDatabase**
  - metody pro DML operace **query**, **insert**, **update**, **delete**, ...
  - **execSQL** pro jakýkoli SQL příkaz a další

DATABÁZE SQLITE

Inicializace databáze

- pro jednodušší inicializaci se používá třída **SQLiteOpenHelper**
- pro vytvoření databáze vyvoříme potomka **SQLiteOpenHelper**
- potřeba zavolat konstruktor rodiče a přepsat nejméně dvě metody **onCreate()** a **onUpgrade()**

DATABÁZE SQLITE

Inicializace databáze

- **super**(Context context, String name, CursorFactory factory, int version)
  - **context**: předáme kontext z aktivity
  - **name**: název databáze
  - **factory**: null, nepotřebujeme
  - **version**: verze databáze pro následné aktualizace

DATABÁZE SQLITE

Inicializace databáze

```
public class MyDatabase extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "my_db";

    private static final int VERSION = 1;

    public MyDatabase(Context context) {

        super(context, DATABASE_NAME, null, VERSION);

    }
```

DATABÁZE SQLITE

Inicializace databáze

- **onCreate(SQLiteDatabase db)**
  - zde proběhne inicializace databáze při prvním přístupu, tj. Zejména vytvoření tabulek
  - například:

```
@Override
public void onCreate(SQLiteDatabase db) {

    db.execSQL("CREATE TABLE todo (id INTEGER, title TEXT, content TEXT)");

}
```

DATABÁZE SQLITE

Inicializace databáze

- **onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)**
  - pokud změníme strukturu databáze a zvýšíme verzi db v konstruktoru SQLiteOpenHelper, systém detekuje změnu a umožní nám provést operace pro upgrade, např. „ALTER TABLE ... ADD COLUMN ...“
  - pokud nemáme více verzí, nemusí dělat nic

DATABÁZE SQLITE

PŘÍSTUP K DATABÁZI

- získáme instanci našeho potomka **SQLiteOpenHelper**
- na něm zavoláme **getReadableDatabase** nebo **getWritableDatabase**
- tyto metody inicializují databázi a vrátí nám instanci **SQLiteDatabase**, se kterou můžeme pracovat
  - **getReadableDatabase**: číst
  - **getWritableDatabase**: číst/zapisovat

DATABÁZE SQLITE

ContentValues

- pro vkládání dat do databáze se používá pomocná třída **ContentValues**
- instance uchovává dvojice klíč:hodnota
  - **klíč**: String, sloupec tabulky
  - **hodnota**: dle datového typu sloupce
- např.

```
ContentValues cv = new ContentValues();
cv.put("column1", value);
cv.put("column2", value);
```

DATABÁZE SQLITE

Vložení dat

- můžeme použít metodu **execSQL** s SQL příkazem („INSERT INTO …“)
- nebo metodu insert(String table, String nullColumnHack, ContentValues cv)
  - **insert(“my_table”, null, cv);**

DATABÁZE SQLITE

Změna dat

- můžeme použít metodu execSQL s SQL příkazem („UPDATE ...“)
- nebo metodu update(String table, ContentValues cv, String whereClause, String[] whereArgs)
  - **table**: tabulka, **cv**: hodnoty pro změnu
  - **whereClause**: klauzule WHERE
  - **whereArgs**: pole řetězců, které jsou doplněny za výskyty „?“ v řetězci whereClause

DATABÁZE SQLITE

Změna dat

```
ContentValues contentValues = new ContentValues();
contentValues.put("column1", value1);
contentValues.put("column2", value2);

mDb.update("table_name", contentValues, "id = ?",
    new String[] {Integer.toString(id)});
```

DATABÁZE SQLITE

ZÍSKÁNÍ DAT

- **Cursor**
  - objekt uchovávájící výsledek dotazu z databáze, zpravidla více záznamů tabulky
  - metody pro změnu ukazatele na řádek výsledku
    - **boolean moveToFirst()**
      - posune ukazatel na první záznam
      - true, pokud byl posun úspěšný
    - **boolean moveToNext()**
      - posune ukazatel na další záznam
      - true, pokud byl posun úspěšný

DATABÁZE SQLITE

ZÍSKÁNÍ DAT

- **Cursor**
  - metody pro získání záznamů z řádku, na který aktuálně ukazuje
    - **getInt|String|Float|...(int columnIndex)**
      - jako argument očekává index sloupce, který získáme pomocí metody **getColumnIndex(String column)**, případně můžeme vložit číslo ručně

DATABÁZE SQLITE

ZÍSKÁNÍ DAT

- získání výsledku dotazu, tj. instance třídy **Cursor**
  - přetížené metody **query** podle částí klauzule SELECT

DATABÁZE SQLITE

ZÍSKÁNÍ DAT

```
int value3 = 0;

// SELECT column1, column2 FROM table_name WHERE column3 = 0 ORDER BY column1

Cursor cursor = mDb.query("table_name",
    new String[] {"column1", "column2"},
    "column3 = ?", new String[] {Integer.toString(value3)},
    null, null,
    "column1");
if (cursor.moveToFirst()) {
    do {
        int value1 = cursor.getInt(cursor.getColumnIndex("column1"));
        String value2 = cursor.getString(cursor.getColumnIndex("column2"));
        // do something with values
    } while (cursor.moveToNext());
}
cursor.close();
```

FILESYSTEM

UKLÁDÁNÍ PŘÍMO DO PAMĚTI ZAŘÍZENÍ

- použití klasického Java I/O
- třídy File, InputStream, InputReader, ...
- získání cest pomocí metod
  - getFilesDir
  - getExternalFilesDir
  - getExternalStoragePublicDirectory,
  - ...

ASSETS

Soubory přibalené k APK

- pro různé vlastní konfigurační soubory, zvuky, jpg a další formáty
- pouze ke čtení
- uložené v **src/main/assets**
- v Javě přístup pomocí metody **getAssets().open(...)**

ÚKOL 6. SEMINÁŘE

1) Do třídy TODO položky přidat datovou složku indikující provedení úkolu. Pro změnu příznaku přidat do editovací aktivity **Switch** „provedeno“.
2) Přidat aktivitu pro nastavení přístupnou z toolbar menu, která bude obsahovat **TextView** a **Switch** pro možnost „Zobrazení i provedených úkolu“. Na základě této možnosti upravit načítání TODO položek (true: zobrazení provedených i neprovedených, false: zobrazení pouze neprovedených). Hodnotu ukládat do **SharedPreferences**.
3) Implementovat ukládání/načítání TODO položek pomocí **SQLite databáze**. Po restartu si aplikace musí seznam položek pamatovat.

Tipy pro řešení:

1) K TODO položce se bude hodit přidat i datovou složku id, která bude uchovávat id záznamu v databázi.
2) Hodí se označit ID v databázi jako primární klíč s příznakem autoincrement pro automatické vytváření ID.
3) Při onCreate/po přidání/po editaci zavolat metodu, která naplní ArrayList aktuálními záznamy z databáze a aktualizuje adaptér.

OTÁZKY

Ptejte se!