Adattárolási lehetőségek, beállítások, objektum relációs leképezés

Ekler Péter

BME VIK AUT, AutSoft

peter.ekler@aut.bme.hu

Tematika

1. Android platform bemutatása, Kotlin alapok
2. Alkalmazás komponensek, Kotlin konvenciók
3. Felhasználói felület
4. Fragmentek, haladó UI
5. Listák kezelése hatékonyan
6. Perzisztens adattárolás, adatbázisok, haladó Kotlin
7. Hálózati kommunikáció
8. Felhő szolgáltatások
9. Helymeghatározás, térkép kezelés
10. Architektúra komponensek, JetPack

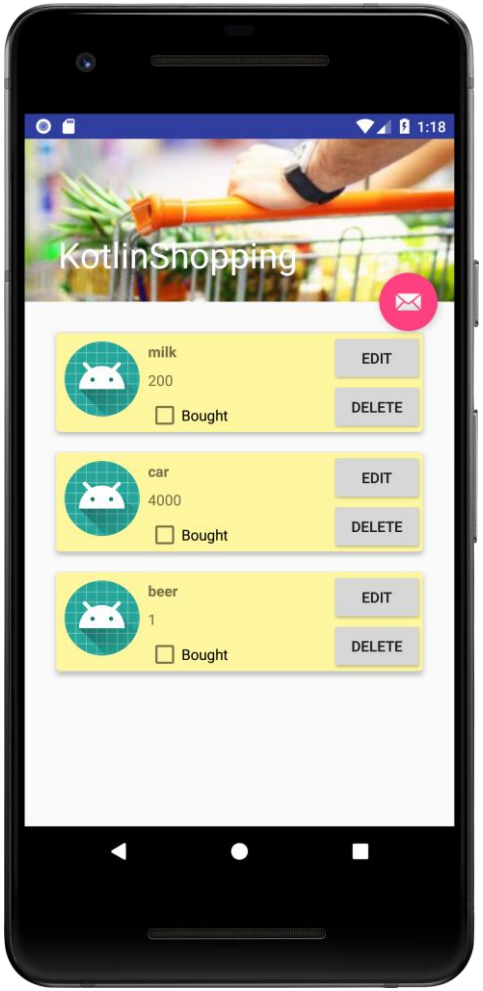
Tartalom

- RecyclerView
- CardView
- RecyclerView - gesztus támogatás
- Perzisztens adattárolás
  - SQLite
  - *SharedPreferences*
  - *File kezelés*
  - *ORM - Room*

LISTÁK KEZELÉSE

Bevásárló lista - RecyclerView

- Termékek listázása
  - > *Név, ár, vásárlás állapota*
- Új termék felvitele
  - > DialogFragment
- Törlés
- Szerkesztés
- Touch gesztusok
- Változások mentése adatbázisba



Swipe és drag&drop gesztusok

- Távolítsuk el az elemeket swipe hatására
- Tegyük lehetővé az elemek átrendezését drag&drop-pal
- *RecyclerView* támogatás:
  - `ItemTouchHelper.Callback`

ItemTouchHelper.Callback 1/2

- *isLongPressDragEnabled():*
  - True visszatérés ha a drag&drop támogatott
- *isItemViewSwipeEnabled():*
  - True visszatérésé ha a swipe támogatott
- *onMove(...):*
  - Elem mozgatás esetén hívódik meg
- *onSwipe(...):*
  - Swipe esetén hívódik meg

ItemTouchHelper.Callback 2/2

- Drag és swipe irányok beállítása:

```
override fun getMovementFlags(recyclerView: RecyclerView,
  viewHolder: RecyclerView.ViewHolder): Int {
    val dragFlags = ItemTouchHelper.UP or ItemTouchHelper.DOWN
    val swipeFlags = ItemTouchHelper.START or ItemTouchHelper.END
    return ItemTouchHelper.Callback.makeMovementFlags(dragFlags, swipeFlags)
}
```

Perzisztens adattárolás

Bevezetés

- Gyakorlatilag minden Android alkalmazásnak kell perzisztensen tárolnia bizonyos adatokat
  - Beállítások szinte mindig vannak
  - Kamera alkalmazások: új fénykép fájl mentése
  - Online erőforrásokat használó appok: lokális cache
  - Email alkalmazások: levelek indexelt adatbázisa
  - Bejelentkezést tartalmazó appok: be van-e jelentkezve a felhasználó
  - Első indításkor tutorial megjelenítése: első vagy későbbi indítás?
  - Picasa, Dropbox: elsődleges tárhely a felhőben

Bevezetés

- Androidon minden igényre van beépített megoldás:
  - **SQLite adatbázis**: strukturált adatok tárolására
  - ***SharedPreferences***: alaptípusok tárolása kulcs-érték párokban
  - **Privát lemezterület**: nem publikus adatok tárolása a fájlrendszerben
  - **SD kártya**: nagy méretű adatok tárolása, nyilvánosan hozzáférhető
  - **Hálózat**: saját webszerveren vagy felhőben tárolt adatok

SQLite

SQLite

- Az Android alapból tartalmaz egy teljes értékű relációs adatbáziskezelőt
  - SQLite – majdnem MySQL
- Strukturált adatok tárolására ez a legjobb választás
- Alapból nincs objektum-relációs réteg (ORM) fölötte, nekünk kell a sémát meghatározni és megírni a query-ket
- Külső ORM osztálykönyvtár:
  - http://ormlite.com/sqlite_java_android_orm.shtml
- Mivel SQL, érdemes minden táblában elsődleges kulcsot definiálni
  - autoincrement támogatás
  - Ahhoz, hogy *ContentProvider*-rel ki tudjuk ajánlani (később), illetve UI elemeket Adapterrel feltölteni (pl. list, grid), **kötelező egy ilyen oszlop**, melynek neve: „_id”

Android SQLite jellemzői 1/2

- Standard relációs adatbázis szolgáltatások:
  - SQL szintaxis
  - Tranzakciók
  - Prepared statement
- Támogatott oszlop típusok (a többit ilyenekre kell konvertálni):
  - TEXT (Java String)
  - INTEGER (Java long)
  - REAL (Java double)
- Az SQLite nem ellenőrzi a típust adatbeíráskor, tehát pl Integer érték automatikusan bekerül Text oszlopba szövegként

Android SQLite jellemzői 2/2

- Az SQLite adatbázis elérés file rendszer elérést jelent, ami miatt lassú lehet!
- Adatbázis műveleteket érdemes aszinkron módon végrehajtani (pl *AsyncTask* használata v. *Loader*)

SQLite debug

- Az Android SDK „platform-tools” mappájában található egy konzolos adatbázis kezelő: sqlite3
- Ennek segítségével futás közben láthatjuk az adatbázist, akár emulátoron, akár telefonon
- Hasznos eszköz, de sajnos nincs grafikus felülete
- Használata (emulátoron, vagy root-olt eszközön):
  - Konzolban megnyitjuk a **platform-tools** könyvtárat
  - „adb shell” futtatása, egy eszköz legyen csatlakoztatva
  - **„sqlite3 data/data/[Package név]/databases/[DB neve]”** futtatása
  - Megkapjuk az SQLite konzolt, itt már az adatbázison futtathatunk közvetlen parancsokat (Pl. „.dump orak;”)

OBJECT RELATION MAPPING (ORM)

Mi az ORM?

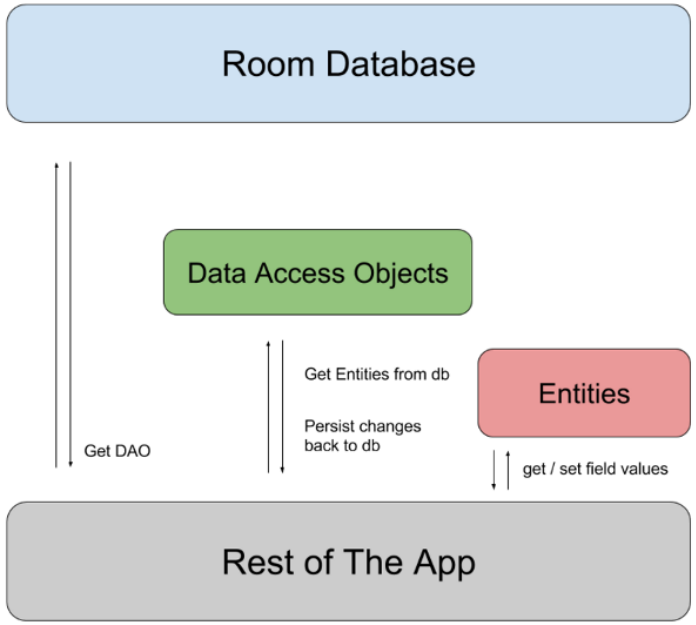
- Java objektumok tárolása relációs adatbázisban
- Alapelvek:
  - Osztálynév -> Tábla név
  - Objektum -> Tábla egy sora
  - Mező -> Tábla oszlopa
  - Stb.

ORM könyvtárak Androidon

- Sugar-ORM
  - http://satyan.github.io/sugar/index.html
- Realm.io (NoSQL), nem SQLite-ot használ
  - http://realm.io
- Objectbox
  - https://objectbox.io/
- ORMLite
  - http://ormlite.com/
- GreenDAO
  - http://greendao-orm.com/

Room Persistence Library

- Absztrakciós réteg az SQLite felett
- SQLite teljes képességeinek használata
- Room architektúra:



Szálkezelés

- *Thread*
  - https://developer.android.com/guide/components/processes-and-threads.html
- A felhasználói felület csak a fő szálról módosítható:
  - *runOnUiThread(runnable: Runnable)*
- Szálakat le kell állítani
  - Biztosítani kell, hogy a run() függvény befejeződjön, ne maradjon végtelen ciklusban

Szál példa

```
private inner class MyThread : Thread() {
    override fun run() {
        while (threadEnabled) {
            runOnUiThread {
                Toast.makeText(this@MainActivity,
                        "Message", Toast.LENGTH_LONG).show()

            }
            Thread.sleep(6000)
        }
    }
}



MyThread().start()
```

Room példa - Entity

```
@Entity(tableName = "grade")
data class Grade(
        @PrimaryKey(autoGenerate = true) var gradeId: Long?,
        @ColumnInfo(name = "studentid") var studentId: String,
        @ColumnInfo(name = "grade") var grade: String
)
```

Room példa - DAO

```
@Dao
interface GradeDAO {
    @Query("""SELECT * FROM grade WHERE grade="B"""")
    fun getBGrades(): List<Grade>

    @Query("SELECT * FROM grade")
    fun getAllGrades(): List<Grade>

    @Query("SELECT * FROM grade WHERE grade = :grade")
    fun getSpecificGrades(grade: String): List<Grade>

    @Insert
    fun insertGrades(vararg grades: Grade)

    @Delete
    fun deleteGrade(grade: Grade)
}
```

RoomDatabase

```
@Database(entities = arrayOf(Grade::class), version = 1)
abstract class AppDatabase : RoomDatabase() {

    abstract fun gradeDao(): GradeDAO

    companion object {
        private var INSTANCE: AppDatabase? = null

        fun getInstance(context: Context): AppDatabase {
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(context.applicationContext,
                        AppDatabase::class.java, "grade.db").build()
            }
            return INSTANCE!!
        }

        fun destroyInstance() {
            INSTANCE = null
        }
    }
}
```

Room használat

- Insert

```
val grade = Grade(null, etStudentId.text.toString(),
          etGrade.text.toString())

val dbThread = Thread {
    AppDatabase.getInstance(this@MainActivity).gradeDao().insertGrades(grade)
}
dbThread.start()
```

- Query

```
val dbThread = Thread {
        val grades = AppDatabase.getInstance(this@MainActivity).gradeDao()
                .getSpecificGrades("A+")
        runOnUiThread {
            tvResult.text = ""
            grades.forEach {
                tvResult.append("${it.studentId} ${it.grade}\n")
            }
        }
}
dbThread.start()
```

Összefoglalás

- Perzisztens adattárolási lehetőségek
- Adatbázistámogatás, SQLite
- ORM megoldások
- Room használata a gyakorlatban
  - Komplex alkalmazás megvalósítása listakezeléssel és adatbázis-támogatással
- Haladó Kotlin nyelvi elemek

A következő alkalommal...

- Egyszerű kulcs-érték tár: SharedPreferences
- File-kezelés, belső és külső tárterület, cache könyvtár
- Backend as a Service szolgáltatások áttekintése
- Firebase képességek
- Adattárolási lehetőségek: real time adatbázis és cloud firestore
- Kamera használati alapok
- Állományok, képek tárolása felhőben
- Push értesítések
- Crash reporting lehetőségek

Köszönöm a figyelmet!

peter.ekler@aut.bme.hu

AutSoft