Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Elektronsko in mobilno poslovanje

VAJE 6

Jan Meznarič
Simon Vrhovec

Ljubljana
30. 11. 2016

Delo z bazo - primer

- Delali bomo z bazo *UserContacts*
- Delali bomo s tabelo *contacts,* ki vsebuje:
  - (name, email, phone, street, city)

```
public class DatabaseConnector {
        private static final String DATABASE_NAME = "UserContacts";


        private static final int DATABASE_VERSION = 1;

        private SQLiteDatabase database; // database object
        private DatabaseOpenHelper databaseOpenHelper; // database helper

        // public constructor for DatabaseConnector
        public DatabaseConnector(Context context) {
                databaseOpenHelper = new DatabaseOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        public void open() throws SQLException {
                // create or open a database for reading/writing
                database = databaseOpenHelper.getWritableDatabase();
        }

        public void close() {
                if (database != null) database.close(); // close the database connection
        }
```

```
public void insertContact(String name, String email, String phone, String state, String city) {
        ContentValues newContact = new ContentValues();
        newContact.put("name", name);
        newContact.put("email", email);
        newContact.put("phone", phone);
        newContact.put("street", state);
        newContact.put("city", city);

        open(); // open the database
        database.insert("contacts", null, newContact);
        close(); // close the database
}

public void updateContact(long id, String name, String email, String phone, String state, String city) {
        ContentValues editContact = new ContentValues();
        editContact.put("name", name);
        editContact.put("email", email);
        editContact.put("phone", phone);
        editContact.put("street", state);
        editContact.put("city", city);

        open(); // open the database
        database.update("contacts", editContact, "_id=" + id, null);
        close(); // close the database
}

public void deleteContact(long id) {
        open(); // open the database
        database.delete("contacts", "_id=" + id, null);
        close(); // close the database
}

public Cursor getAllContacts() {
        return database.query("contacts", new String[] { "_id", "name" }, null,
        null, null, null, "name");
}

// get a Cursor containing all information about the contact specified by the given id
public Cursor getOneContact(long id) {
        return database.query("contacts", null, "_id"= + id, null, null, null, null);
}
```

```
        private class DatabaseOpenHelper extends SQLiteOpenHelper {
                // public constructor
                public DatabaseOpenHelper(Context context, String name, CursorFactory factory, int version) {
                        super(context, name, factory, version);
                }

                // creates the contacts table when the database is created
                @Override
                public void onCreate(SQLiteDatabase db) {
                        String createQuery = "CREATE TABLE contacts"
                                + "(_id integer primary key autoincrement,"
                                + "name TEXT, email TEXT, phone TEXT,"
                                + "street TEXT, city TEXT);";

                        // initializing the database
                        String insertValues = "INSERT INTO contacts (_ID, name, email, phone, street, city) values (NULL, 'MOJCA',
                        'mojca@gmail.com', '041-444-555', 'Trzaska cesta 25', 'Ljubljana');";

                        db.execSQL(createQuery); // execute the query
                        db.execSQL(insertValues);
                }

                @Override
                public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                        // code for updates
                }

                @Override
                public void onDowngrade (SQLiteDatabase db, int oldVersion, int newVersion) {
                        // from Android 3.0 when the database needs to be downgraded
                }
        }
} // end class DatabaseConnector
```

Identifikator vsakega zapisa, ki je obvezen za delovanje razreda CursorAdapter

Prikaz z uporabo razreda `ListView`

```
<android.support.v7.widget.ListViewCompat
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior" />

<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/contactTextView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:textSize="20sp" />
```

```
contactListView = (ListView) findViewById(R.id.listView);
contactListView.setOnItemClickListener(viewContactListener);

String[] from = new String[]{"name"};
int[] to = new int[]{R.id.contactTextView};

contactAdapter = new SimpleCursorAdapter(this, R.layout.contact_list_item,
                            null, from, to, 0);

contactListView.setAdapter(contactAdapter);
```

Pregledovanje podatkov

- Sqlite3
  - Baza je v pomnilniku naprave na lokaciji `/data/data/<ime.paketa>/databases/<ImeBaze>`
  - ADB je na lokaciji `<Android>\sdk\platform-tools`
  - Poženemo:
    1. `adb shell`
    2. `sqlite3 /data/data/emp.addressbook/databases/UserContacts`
  - Pregledovanje in interakcije z bazo
    - `.tables` – izpis vseh tabel
    - Uporaba poljubnih SQL stavkov

Vaja 11

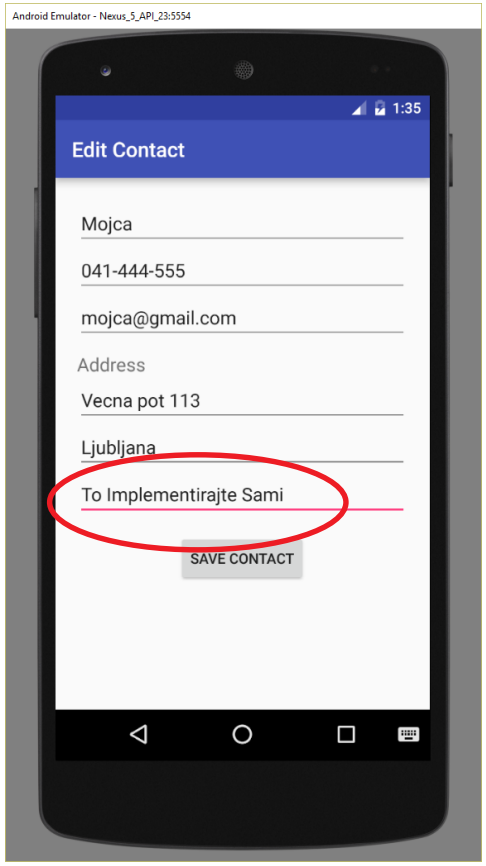
- Uporabite ADB in v bazo vaše aplikacije dodajte še nekaj različnih vrednosti

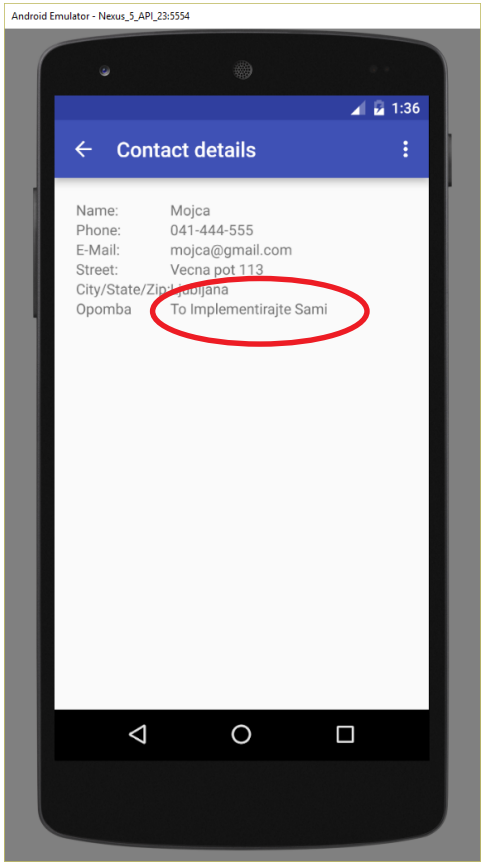
- Primer INSERT stavka

  INSERT INTO contacts (name, email, phone, street, city) values ('MARKO', 'marko@gmail.com', '041-124-123', 'Vecna pot 113', 'Ljubljana');

Vaja 12

- V program AddressBook dodajte polje “opomba“ ter poskrbite, da bo program deloval v celoti.





AsyncTask

- Razred AsyncTask omogoča enostavno uporabo niti. Primeren je zlasti za izvajanje dolgotrajnejših procesov oz. procesov, za katere ne moremo z gotovostjo napovedati časa trajanja (npr. povezava z internetom, prenos datoteke, odpiranje baze itd.).

- Na ta način zagotovimo, da je uporabniški vmesnik še vedno odziven, hkrati pa lahko nanj pošiljamo tudi sporočila o napredku.

Primer – AsyncTask

```
private class MyAsyncTask extends AsyncTask<InParams, ProgressParams, OutParam> {
    protected OutParam doInBackground(InParams... myparams) {
        int count = myparams.length;
        OutParam totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(myparams [i]);
            publishProgress((ProgressParams) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(ProgressParams... myparams) {
        setProgressPercent(myparams [0]);
    }

    protected void onPostExecute(OutParam result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

```
myTask1=(MyAsyncTask)new MyAsyncTask().execute(<InParams>);
myTask1.cancel(true);
```