PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET METROPOLITAN BEOGRAD

KI205 - JAVA 8: PROGRAMIRANJE U JAVI NA ANDROID PLATFORMI

Čuvanje podataka u Android aplikacijama

Lekcija 05

PRIRUČNIK ZA STUDENTE

KI205 - JAVA 8: PROGRAMIRANJE U JAVI NA ANDROID PLATFORMI

Lekcija 05

ČUVANJE PODATAKA U ANDROID APLIKACIJAMA

- Čuvanje podataka u Android aplikacijama
- Poglavlje 1: Snimanje i učitavanje korisničkih preferencija
- Poglavlje 2: Čuvanje podataka u datotekama
- Poglavlje 3: Primena baza podataka
- Poglavlje 4: Domaći zadatak 5
- Pregled Lekcije06

UVOD

Android operativni sistem podržava tri načina čuvanja podataka.

Perzistencija ili čuvanje podataka je nezaobilazni problem sa kojim svaka ozbiljna aplikacija mora da se izbori. Jednostavno, u savremenim aplikacijama, pa tako i mobilnim, u velikom broju slučajeva nije dovoljno da se neka informacija, prostog ili objektnog tipa, samo jednom iskoristi već je njena upotreba višestruka. Otuda će u ovoj lekciji će biti govora o načinima čuvanja podataka u Android aplikacijama. Ovo je veoma značajna tema budući da se u svakoj aplikaciji podaci mogu koristiti više puta. Android operativni sistem podržava sledeće načine čuvanja podataka:

- Deljenje preferencija;
- Čuvanje podataka u datotekama;
- Čuvanje podataka u bazama podataka.

Lekcija će staviti akcenat na sledeće teze:

- Snimanje jednostavnih podataka primenom SharedPreferences objekata;
- Modifikovanje preferencija upotrebom PreferenceActivity klase;
- Snimanje i učitavanje datoteka iz unutrašnje i ekstrerne memotije;
- Korišćenje i kreiranje SQLite baze podataka.

Nakon savladavanja ove lekcije, studenti će biti osposobljeni da koriste deljene preferencije u Android aplikacijama, da kreiraju mobilne aplikacije sa bazama podataka i da upravljaju datotekama čuvanim na unutrašnjoj ili eksternoj memoriji mobilnog uređaja.

Poglavlje 1

Snimanje i učitavanje korisničkih preferencija

PRISTUPANJE PREFERENCIJAMA U POKRENUTOJ AKTIVNOSTI

Android obezbeđuje SharedPreferences objekat za čuvanje jednostavnih podataka u aplikacijama.

Android operativni sistem obezbeđuje *SharedPreferences* objekat za čuvanje jednostavnih podataka u aplikacijama. Od posebnog značaja su podaci koje je definisao sam korisnik, tako da kada ponovo pokrene aplikaciju, ona bude podešena upravo na način kako to korisnik želi.

Korišćenjem objekta SharedPreferences dobija se mogućnost snimanja podataka od značaja u formi para (naziv, vrednost) koji se, zatim, automatski snimaju u odgovarajuću XML datoteku.

Sledećim primerom biće demonstrirano čuvanje podataka upotrebom objekta SharedPreferencesi kako korisnik može snimljene podatke da direktno modifikuje upotrebom posebnog tipa aktivnosti podržane Android operativnim sistemom.

U Android Studiorazvojnom okruženju biće kreiran projekat za demonstraciju rada sa preferencijama.Prvi zadatak je kreiranje podfoldera xml u okviru kojeg će biti kodirana XML datoteka sa nazivom myapppreferences.xml. Upravo u ovoj datoteci će biti čuvane korisničke preferencije koje je moguće posle koristi u Android aplikacijama kao podešavanja ili podrazumevane vrednosti.

Sledećom sliko je pokazana pozicija datoteke myapppreferences.xml u hijerarhiji projekta.

Slika 1.1 Položaj datoteke preferencija u hijerarhiji projekta

PRIMENA PREFERENCIJA – XML DATOTEKE

XML datotekom preferencija čuvaju se podešavanja aplikacije.

Kreirana datoteka sa preferencijama mora da bude snabdevena XML kodom koji će poslužiti za definisanje izvesnih podešavanja koja želimo da uvedemo u aplikaciju. Zbog toga je neophodno pristupiti lokaciji res/xml, otvoriti datoteku myapppreferences.xml, pomoću tekst editora ugrađenog u razvojno okruženje Android Studio IDE (ili nekog drugog tekst editora, na primer Notepad), i u nju ugraditi odgovarajući XML kod. Primer ovakvog koda je priložen sledećim listingom.

```
<?xml version="1.0" encoding="utf-8"?>


        <CheckBoxPreference
            android:title="Izaberite opciju"
            android:defaultValue="false"
            android:summary="Čekiran ili nečekiran"
            android:key="checkboxPref" />
        </PreferenceCategory>

        <EditTextPreference
            android:summary="Učitajte string"
            android:defaultValue="[Učitajte string ovde]"
            android:title="Izmenite tekst"
            android:key="editTextPref"
            />
        <RingtonePreference
            android:summary="Izaberite melodiju zvona"
            android:title="Melodije zvona"
            android:key="ringtonePref"
            />

            <EditTextPreference
                android:summary="Učitajte string"
                android:title="Učitajte string (drugi ekran)"
                android:key="secondEditTextPref"
                />
        </PreferenceScreen>
    </PreferenceCategory>
</PreferenceScreen>
```

U narednom koraku je neophodno definisati korisnički interfejs koji će biti učitan pozivom onCreate() metode glavne klase aktivnosti, koja u ovom projektu nosi opšti naziv MainActivity.java. Korisnički interfejs može da omogući brojne funkcionalnosti za rad sa preferencijama, na primer:

- pristup ekranu sa podešavanje preferencija;
- modifikovanje vrednosti preferencija;
- prikazivanje trenutnih vrednosti preferencija i tako dalje.

Ovakav grafički korisnički interfejs je moguće realizovati sledećim XML kodom datoteke activity_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/btnPreferences"
        android:text="Učitajte ekran preferencija"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickLoad"/>

    <Button
        android:id="@+id/btnDisplayValues"
        android:text="Prikažite vrednosti preferencija"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickDisplay"/>

    <EditText
        android:id="@+id/txtString"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btnModifyValues"
        android:text="Modifikujte vrednosti preferencija"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickModify"/>
</LinearLayout>
```

KLASA TIPA PREFERENCEACTIVITY

U nastavku projekta neophodno je implementirati naslednicu klase PreferenceActivity.

Da bi rad sa kreiranjem, čuvanjem i prikazivanjem preferencija, u Android aplikacijama bio moguć, neophodno je implementirati, pored glavne klase aktivnosti, i klasu koja nasleđuje osnovnu klasu pod nazivom PreferenceActivity. Ova klasa je prekoIntent objekat povezana sa klasom glavne aktivnosti i zadatak joj je da, upravo, učita XML dokument sa preferencijama. Klasa, u projektu, nosi naziv AppPreferenceActivity.java i sledećim listingom je priložen njen JAVA kod.

```
package com.metropolitan.preferencijedemo;

import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;

/**
 * Created by Vladimir Milicevic on 3.11.2016..
 */

public class AppPreferenceActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);

        PreferenceManager prefMgr = getPreferenceManager();
        prefMgr.setSharedPreferencesName("appPreferences");

        //---učitavanje preferencija iz XML datoteke---
        addPreferencesFromResource(R.xml.myapppreferences);
    }
}
```

Nakon kreiranog koda XML datoteka i naslednice klase PreferenceActivity, neophodno je uvesti dodatna pojašnjenja.

Kreiranjem XML datoteka preferencija i interfejsa, kreiran je korisnički interfejs koji sadrži sledeće elemente:

- Dve kategorije za obuhvatanje različitih tipova preferencija;
- Dva polja za potvrdu sa ključevima checkBoxPref i secondPrefScreenPref;
- Opciju za podešavanje melodije zvona sa ključem ringtonePref;
- Ekran sa opcijama za dodatne preferencije.

Specijalno, atributandroid:key definiše ključ koji je moguće programski referencirati u kodu sa ciljem definisanja ili preuzimanja vrednosti određene preferencije.

Da bi Android mogao da prikaže sve opcije koje su dostupne za izmenu, bilo je neophodno kreirati klasu naslednicu osnovne klase PreferenceActivity. Izvršavanjem njene metode addPreferenceFromResource() učitava se xml datoteka sa odgovarajućim podešavanjima.

UČITAVANJE I MODIFIKACIJA PREFERENCIJA

Za korišćenje podešavanja neophodno je koristiti klasu SharedPreferences.

Do sada je pokazano kako klasa *PreferenceActivity* omogućava jednostavno kreiranje preferencija i modifikovanje određenih opcija. Da bi u toku izvršavanja aplikacije bilo moguće koristiti navedena podešavanja neophodno je koristiti klasu SharedPreferences. U nastavku biće formirana i glavna klasa aktivnosti pod nazivom MainActivity koja nasleđuje baznu klasu Activity (ili AppComatActivity za najnovije Android API verzije) i koristi klasu SharedPreferences. Postavljanjem definisije ove klase omogućeno je testiranje kreiranih Android funkcionalnosti za upravljanje preferencijama.

Sledećim listingom je priložen JAVA kod glavne klase aktivnosti.

```
package com.metropolitan.preferencijedemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Intent;
import android.content.SharedPreferences;
```

```
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClickLoad(View view) {
        Intent i;
        i = new Intent("com.metropolitan.preferencijedemo.AppPreferenceActivity");
        startActivity(i);
    }
    public void onClickDisplay(View view) {

        SharedPreferences appPrefs =

getSharedPreferences("com.metropolitan.preferencijedemo_preferences",
                        MODE_PRIVATE);


        DisplayText(appPrefs.getString("editTextPref", ""));
    }
    public void onClickModify(View view) {

        SharedPreferences appPrefs =

getSharedPreferences("com.metropolitan.preferencijedemo_preferences",
                        MODE_PRIVATE);


        SharedPreferences.Editor prefsEditor = appPrefs.edit();
        prefsEditor.putString("editTextPref",
                ((EditText) findViewById(R.id.txtString)).getText().toString());
        prefsEditor.commit();
    }
    private void DisplayText(String str) {
        Toast.makeText(getBaseContext(), str, Toast.LENGTH_LONG).show();
    }

}
```

UČITAVANJE I MODIFIKACIJA PREFERENCIJA - FUNKCIONISANJE

Objekat klase SharedPreferences dobijen je pozivom metode getSharedPreferences() .

Prva značajna metoda (pogledati priloženi fragment koda koji sledi) koja je implementirana u klasi MainActivity je onClickDisplay(). U okviru ove metode, koja je vezana za klik na prvo dugme korisničkog interfejsa, dešava se poziv metode getSharedPreferences() pomoću koje je dobijen objekat klase SharedPreferences. Navedeno je omogućeno kroz specificiranje naziva XML datoteke (u konkretnom slučaju ("com.metropolitan.preferencijedemo_preferences") formatirano kao *nazivPaketa.NazivKlase_preferences.* Učitavanje podataka tipa string, realizovano je pozivom metode getString() kojoj je predat ključ preferencije koja se učitava. Konstanta MODE_PRIVATE ukazuje da datoteka sa preferencijama može da se koristi samo u aplikaciji projekta u kojem je kreirana.

Fragment koda kojim je realizovana navedena funkcionalnost priložen je sledećim listingom.

```
public void onClickDisplay(View view) {

        SharedPreferences appPrefs =

getSharedPreferences("com.metropolitan.preferencijedemo_preferences",
                        MODE_PRIVATE);


        DisplayText(appPrefs.getString("editTextPref", ""));
    }
```

U sledećoj metodi, onClickModify(), vezanoj za klik na treće dugme kreiranog korisničkog interfejsa, kreiran je objekat SharedPreferences.Editorupotrebom metode edit() pozvane objektom klase SharedPreferences. Za primenu vrednosti opcije, tipa string, iskorišćena je metoda putString(), a snimanje promene, u datoteku preferencija, obavljeno je metodom commit(). Fragment koda kojim je realizovana navedena funkcionalnost priložen je sledećim listingom.

```
public void onClickModify(View view) {

        SharedPreferences appPrefs =

getSharedPreferences("com.metropolitan.preferencijedemo_preferences",
                        MODE_PRIVATE);


        SharedPreferences.Editor prefsEditor = appPrefs.edit();
        prefsEditor.putString("editTextPref",
                ((EditText) findViewById(R.id.txtString)).getText().toString());
        prefsEditor.commit();
    }
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PROMENA PODRAZUMEVANOG NAZIVA DATOTEKE SA PREFERENCIJAMA

Ponekad je korisno dati specifičan naziv datoteci sa preferencijama.

U Android aplikacijama je moguće i upravljanje nazivima datoteka sa preferencijama. U konkretnom slučaju, datoteka sa prefernecijama je snimljena na mobilnom uređaju pod nazivom com.metropolitan.preferencijedemo_preferences. Međutim, programeri često daju izvesne specifične nazive datotekama sa preferencijama iz raznih razloga. To je moguće učiniti ako se u kreiranoj klasi AppPreferencesActivity.java naprave korekcije koje su istaknute sledećim kodom.

```
PreferenceManager prefMgr = getPreferenceManager();
        prefMgr.setSharedPreferencesName("appPreferences");

        //---učitavanje preferencija iz XML datoteke---
        addPreferencesFromResource(R.xml.myappppreferences);
```

Upravo na ovaj način je kreirana nova daoteka sa preferencijama koja se na mobilnom uređaju čuva pod anzivom appPreferences.

Klasa MainActivity.java definiše naziv datoteke za čuvanje preferencija, a to u daljem radu može biti nova datoteka appPreferences. Navedeno je moguće realizovati ako se u glavnoj klasi aktivnosti uvedu izmene koda koje se odnose na nazive datoteka preferencija. Upravo je to prikazano sledećim listingom.

```
package com.metropolitan.preferencijedemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Intent;
import android.content.SharedPreferences;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClickLoad(View view) {
        Intent i;
        i = new Intent("com.metropolitan.preferencijedemo.AppPreferenceActivity");
        startActivity(i);
    }
    public void onClickDisplay(View view) {
```

```
        SharedPreferences appPrefs =
                getSharedPreferences("appPreferencess",
                        MODE_PRIVATE);


        DisplayText(appPrefs.getString("editTextPref", ""));
    }
    public void onClickModify(View view) {

        SharedPreferences appPrefs =
                getSharedPreferences("appPreferences",
                        MODE_PRIVATE);


        SharedPreferences.Editor prefsEditor = appPrefs.edit();
        prefsEditor.putString("editTextPref",
                ((EditText) findViewById(R.id.txtString)).getText().toString());
        prefsEditor.commit();
    }
    private void DisplayText(String str) {
        Toast.makeText(getBaseContext(), str, Toast.LENGTH_LONG).show();
    }

}
```

PRIMER SA PREFERENCIJAMA - DEMONSTRACIJA

Nakon promene naziva podrazumevane datoteke sa preferencijama, obe datoteke su sačuvane u odgovarajućem folderu.

U poslednjem koraku neophodno je izvršiti testiranje kreirane aplikacije za rad sa preferencijama. U Android Studio IDE razvojnom okruženju, izborom opcije Run app (ili Shift + F10) pokreće se debagovanje i testiranje aplikacije u izabranom emulatoru ili realnom mobilnom uređaju.

Navigacija kroz program prikazana je sledećom slikom.

Slika 1.2 Kretanje kroz aplikaciju sa preferencijama

U primeru je prikazan način kako je moguće programski promeniti podrazumevani naziv datoteke sa preferencijama. Obe datoteke, izvorna i nova, čuvaju se u uređaju u folderu data/data/com.metropolitan.preferencijedemo/shared_prefs (sledeća slika).

Slika 1.3 Datoteke sa preferencijama

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 1 - ČUVANJE PODATAKA POMOĆU PREFERENCIJA

Vežbanje koncepta preferencije

Zadatak:

Kreirati Android aplikaciju za čuvanje podataka u korisničkim preferencijama. Omogućiti više ekrana za unos preferencija i kategorije preferencija.

Korisnički interfejs može biti definisan sledećim kodom.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="@drawable/metpozadina">

    <Button
        android:id="@+id/btnPreferences"
        android:text="Učitajte ekran preferencija"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickLoad"/>

    <Button
        android:id="@+id/btnDisplayValues"
        android:text="Prikažite vrednosti preferencija"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickDisplay"/>

    <EditText
        android:id="@+id/txtString"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btnModifyValues"
        android:text="Modifikujte vrednosti preferencija"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickModify"/>
</LinearLayout>
```

XML datoteka preferencija može biti data sledećim kodom:

```
<?xml version="1.0" encoding="utf-8"?>


```

```
        <CheckBoxPreference
            android:title="Izaberite opciju"
            android:defaultValue="false"
            android:summary="Čekiran ili nečekiran"
            android:key="checkboxPref" />
        </PreferenceCategory>

        <EditTextPreference
            android:summary="Učitajte string"
            android:defaultValue="[Učitajte string ovde]"
            android:title="Izmenite tekst"
            android:key="editTextPref"
            />
        <RingtonePreference
            android:summary="Izaberite melodiju zvona"
            android:title="Melodije zvona"
            android:key="ringtonePref"
            />

            <EditTextPreference
                android:summary="Učitajte string"
                android:title="Učitajte string (drugi ekran)"
                android:key="secondEditTextPref"
                />
        </PreferenceScreen>
    </PreferenceCategory>
</PreferenceScreen>
```

Klasa preferencija je data sledećim kodom:

```
package com.metropolitan.preferencijedemo;

import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;

/**
 * Created by Vladimir Milicevic on 3.11.2016..
 */

public class AppPreferenceActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        PreferenceManager prefMgr = getPreferenceManager();
        prefMgr.setSharedPreferencesName("appPreferences");

        //---učitavanje preferencija iz XML datoteke---
        addPreferencesFromResource(R.xml.myapppreferences);
    }
}
```

Glavna klasa aktivnosti data je sledećim kodom:

```
package com.metropolitan.preferencijedemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Intent;
import android.content.SharedPreferences;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClickLoad(View view) {
        Intent i;
        i = new Intent("com.metropolitan.preferencijedemo.AppPreferenceActivity");
        startActivity(i);
    }
    public void onClickDisplay(View view) {

        SharedPreferences appPrefs =

getSharedPreferences("com.metropolitan.preferencijedemo_preferences",
                        MODE_PRIVATE);


        DisplayText(appPrefs.getString("editTextPref", ""));
    }
    public void onClickModify(View view) {

        SharedPreferences appPrefs =

getSharedPreferences("com.metropolitan.preferencijedemo_preferences",
                        MODE_PRIVATE);


        SharedPreferences.Editor prefsEditor = appPrefs.edit();
        prefsEditor.putString("editTextPref",
                ((EditText) findViewById(R.id.txtString)).getText().toString());
        prefsEditor.commit();
    }
    private void DisplayText(String str) {
        Toast.makeText(getBaseContext(), str, Toast.LENGTH_LONG).show();
    }

}
```

ZADATAK 1 - KREIRAJTE VLASTITU APLIKACIJU ZA UPRAVLJANJE PREFERENCIJAMA

Pokušajte sami

- Samostalno kreirajte Android aplikaciju za čuvanje podataka u korisničkim preferencijama.
- Omogućiti više ekrana za unos preferencija i kategorije preferencija.
- Preferencije izabrati po vlastitom izboru.

Poglavlje 2

Čuvanje podataka u datotekama

ČUVANJE PODATAKA U INTERNOJ MEMORIJI

Klase koje omogućavaju tradicionalno čuvanje podataka nalaze se u paketu java.io.

U prethodnom slučaju je pokazano da objekat klase SharedPreferences omogućava snimanje podataka u formi para (naziv, vrednost). Međutim, ponekad je poželjno čuvati podatke na tradicionalan način, u datotekama, poput raznih tekstualnih dokumenata koji bi trebalo da budu prikazani u nekoj Android aplikaciji. Kompletna podrška, klasa za rad sa datotekama, nalazi je u JAVA paketu *java.io*.

Mobilni uređaji dozvoljavaju dva načina čuvanja podataka u datotekama:

- Snimanje datoteka sa podacima na internu memoriju uređaja;
- Snimanje datoteka sa podacima na eksternu memoriju uređaja.

Oba načina čuvanja podataka biće prikazana u ovoj lekciji, a sada će se krenuti od čuvanja podataka u internoj memoriji Android uređaja. U tu svrhu biće demonstrirano kako se string, kojeg unosi korisnik kroz interakciju sa korisničkim interfejsom, snima u datoteku na unutrašnjem memorijskom prostoru mobilnog uređaja.

Kao što je moguće primetiti, kod koji se odnosi na čitanje i pisanje u datoteke je u velikoj meri obrađen na predmetima prve godine u kojima se studenti upoznaju sa JAVA programskim jezikom. Iz navedenog razloga ovome se neće obraćati velika pažnja i akcenat će biti na specifičnim detaljima rada sa datotekama u Android aplikacijama.

Prvi korak će biti kreiranje projekta sa nazivom DatotekeDemo za koji se kreira datoteka korisničkog interfejsa, koja će zadržati podrazumevani naziv activity_main.xml datoteka. Kod ove datoteke je prikazan sledećim listingom.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Unesite neki tekstualni podatak!" />

    <EditText
```

```
        android:id="@+id/txtText1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btnSave"
        android:text="Snimite"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickSave" />

    <Button
        android:id="@+id/btnLoad"
        android:text="Učitajte"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickLoad" />
</LinearLayout>
```

ČUVANJE PODATAKA U INTERNOJ MEMORIJI – JAVA KLASA

Metode klase za rad sa ulazno/izlaznim podacima datoteka moraju da obrađuju izuzetke.

Sledećim kodom biće prikazana JAVA klasa aktivnosti projekta. Moguće je primetiti da su sve programske manipulacije sa učitavanjem, prikazivanjem i snimanjem podataka praćene upravljanjem izuzecima i realizacijom *try – catch* blokova koda. Kao što je napomenuto, upravljanje pisanjem u datoteku i čitanjem iz datoteke detaljno je obrađivano u prvoj godini na predmetima vezanim za programiranje u JAVA programskom jeziku. Isti je slučaj sa obradom izuzetaka.

Programski kod koji realizuje pomute zadatke integrisan je u glavnu klasu aktivnosti koja je prilikom definisanja projekta zadržala podrazumevani naziv MainActivity.java.

```
package com.metropolitan.datotekedemo;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
```

```
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {

    EditText textBox;
    static final int READ_BLOCK_SIZE = 100;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textBox = (EditText) findViewById(R.id.txtText1);

        InputStream is = this.getResources().openRawResource(R.raw.textfile);
        BufferedReader br = new BufferedReader(new InputStreamReader(is));

        String str = null;
        try {
            while ((str = br.readLine()) != null) {
                Toast.makeText(getBaseContext(),
                        str, Toast.LENGTH_SHORT).show();
            }
            is.close();
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public void onClickSave(View view) {
        String str = textBox.getText().toString();
        try
        {    //---Izbor skladišta datoteke---
             FileOutputStream fOut =
openFileOutput("textfile.txt",                  MODE_WORLD_READABLE);

            OutputStreamWriter osw = new
                    OutputStreamWriter(fOut);

            //---Upisivanje stringa u datoteku---
            osw.write(str);
            osw.flush();
            osw.close();

            //---Poruka o snimljenom podatku---
            Toast.makeText(getBaseContext(),
                    "Datoteka je uspešno snimljena!",
                    Toast.LENGTH_SHORT).show();

            //---Očisti EditText pogled---
            textBox.setText("");
        }
        catch (IOException ioe)
```

```
        {
            ioe.printStackTrace();
        }
    }

    public void onClickLoad(View view) {
        try
        {   //---Izbor skladišta datoteke---
          FileInputStream fIn = openFileInput("textfile.txt");
          InputStreamReader isr = new InputStreamReader(fIn);

            char[] inputBuffer = new char[READ_BLOCK_SIZE];
            String s = "";
            int charRead;
            while ((charRead = isr.read(inputBuffer))>0)
            {//---Konverzija niza karaktera u string---
                String readString =
                        String.copyValueOf(inputBuffer, 0,
                                charRead);
                s += readString;
                inputBuffer = new char[READ_BLOCK_SIZE];
            }
            //---Podešavanje EditText na učitani tekst
            textBox.setText(s);

            Toast.makeText(getBaseContext(),
                    "Datoteka je uspešno učitana!",
                    Toast.LENGTH_SHORT).show();
        }
        catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }

}
```

ČUVANJE PODATAKA U INTERNOJ MEMORIJI - FUNKCIONISANJE

Podaci se čuvaju kao niz karaktera.

Da bi neki sadržaj bio sačuvan u datoteci, neophodno je koristiti klasu FileOutputStream. Korišćenjem metode openFileOutput() otvara se datoteka za snimanje podataka u odgovarajućem režimu. Režimi mogu biti:

- U konkretnom slučaju, režim je MODE_WORD_READABLE, a to znači da datoteku mogu čitati i druge apliakcije;
- MODE_PRIVATE – datoteku može da čita jedino aplikacija koja ju je kreirala;
- MODE_APPEND – podaci se dodaju na kraj datoteke;

- MODE_WORD_WRITABLE – datoteci mogu da pristupaju i druge aplikacije sa mogućnošću snimanja podataka u datoteku.

U datoteci, podaci se čuvaju kao niz karaktera i, da bi bili učitani, neophodno ih je konvertovati u string. Takođe, pre upisivanja podataka u datoteku, podaci tipa string se prevode u niz karaktera.

Da bi niz karaktera bio konvertovan u string, koristi se objekat klase OutputStreamWriterkojem se prosleđuje objekat tipa FileOutputStream. Navedeno pokazuje priloženi kod klase aktivnosti.

Nakon konverzije, primenom metode write()vrši se snimanje stringa u datoteku. Da bi bilo osigurano da će svi bajtovi (karakteri) biti snimljeni u datoteci, koristi se metoda flush(). Na kraju, metodom close() zatvara se datoteka za upisivanje (videti kod klase ativnosti projekta).

Da bi sadržaj neke datoteke mogao biti učitan, koristi se klasa FileInputStream u kombinaciji sa klasom InputStreamReader. Nakon konverzije niza karaktera u string, vrši se njegovo učitavanje. Često nije poznata veličina datoteke koja se učitava. Zato taj sadržaj može da se prosleđuje u bafer koji predstavlja, u konkretnom slučaju, blok od 100 karaktera. Sledećim kodom je prikazano učitavanje stringa iz datoteke uz napomenu da metoda read() proverava broj karaktera koji se učitavaju i vraća vrednost -1 ukoliko je dotignut kraj datoteke.

```
char[] inputBuffer = new char[READ_BLOCK_SIZE];
            String s = "";
            int charRead;
            while ((charRead = isr.read(inputBuffer))>0)
            {//---Konverzija niza karaktera u string---
                String readString =
                        String.copyValueOf(inputBuffer, 0,
                                charRead);
                s += readString;
                inputBuffer = new char[READ_BLOCK_SIZE];
            }
```

ČUVANJE PODATAKA U INTERNOJ MEMORIJI - DEMONSTRACIJA

Moguće je u DDMS prikazu proveriti da li je aplikacija zaista snimila datoteku.

Sada je neophodno testirati kreirani programski kod za upravljanje tekstualnom datotekom koja se čuva na unutrašnjoj memoriji mobilnog uređaja (simulirano emulatorom u ovom slučaju).

Klikom na Shift + F10 (ili Run appl) u Android Studio IDE razvojnom okruženju, program se prevodi i pokreće se AVD uređaj, u demonstriranom slučaju AVD visokih performansi Genymotion. Korisnički interfejs nudi opcije čuvanja i čitanja podataka iz datoteka i polje za unos teksta koji može biti upisan u datoteku. Navedeno je prikazano sledećom slikom.

Slika 2.1 GUI za manipulaciju datotekama

Unošenjem stringa u polje za unos teksta i klikom na dugme SNIMITE, uneti string se čuva u kreiranoj tekstualnoj datoteci. Klikom na dugme UČITAJTE, tekst koji se čuva u navedenoj tekstualnoj datoteci se učitava u glavnu aktivnost i Toast klasom prikazuje na ekranu mobilnog uređaja ili emulatora.

Moguće je u DDMS prikazu, veoma lako korišćenjem *File Exprorer-a*, proveriti da li je aplikacija snimila pomenutu tekstualnu datoteku u folder res/raw. Provera je prikazana sledećom slikom.

Slika 2.2 Snimljena tekstualna datoteka sa odgovarajućim podacima

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ČUVANJE PODATAKA U EKSTERNOJ MEMORIJI

Mobilnost SD kartice omogućava lakšu razmenu podataka između korisnika.

U prethodnom izlaganju je prikazano kako se čuvaju podaci na unutrašnjem memorijskom skladištu mobilnog uređaja. Ponekad je korisno snimiti određene datoteke na SD karticu uređaja, zbog njenog većeg kapaciteta ili olakšane distribucije podataka - vađenjem kartice iz uređaja i prosleđivanjem drugom korisniku. Na prethodnom primeru biće napravljena korekcija u formi zamene određenih linija koda metoda onClickSave() i onClickLoad().

```
public void onClickSave(View view) {
        String str = textBox.getText().toString();
        try
        {    //---stari kod---
           /*  FileOutputStream fOut =
openFileOutput("textfile.txt",                  MODE_WORLD_READABLE);*/
            //---Novi kod---
            File sdCard = Environment.getExternalStorageDirectory();
            File directory = new File (sdCard.getAbsolutePath() +
                    "/MyFiles");
            directory.mkdirs();
            File file = new File(directory, "textfile.txt");
            FileOutputStream fOut = new FileOutputStream(file);


           ......
    }
-------------------------------------------------------------------------------
-----------------

public void onClickLoad(View view) {
        try
```

```
        {   //---stari kod---
          /*FileInputStream fIn = openFileInput("textfile.txt");
           InputStreamReader isr = new InputStreamReader(fIn);*/
            //---Novi kod--
            File sdCard = Environment.getExternalStorageDirectory();
            File directory = new File (sdCard.getAbsolutePath() +
                    "/MyFiles");
            File file = new File(directory, "textfile.txt");
            FileInputStream fIn = new FileInputStream(file);
            InputStreamReader isr = new InputStreamReader(fIn);


           ..........
    }
```

U prikazanom kodu, metodagetExternalStorageDirectory() vraća apsolutnu putanju do eksterne memorije uređaja. Putanja je najčešće predstavljena kao /sdcardna realnom uređaju ili /mnt/sdcardukoliko se aplikacija izvršava na emulatoru. Budući da naziv putanje može da bude različit, u zavisnosti od proizvođača kartice, ne preporučuje se ovaj način obeležavanja kartice u samom kodu. Upravo iz navedenog razloga, primena date metode je od izuzetnog značaja. Datoteka će biti snimljena u folder MyFilesna SD kartici (videti kod i sliku).

Slika 2.3 Datoteka na SD kartici

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ČUVANJE PODATAKA U EKSTERNOJ MEMORIJI – DODAVANJE PRIVILEGIJA

Bez dodavanja privilegije SD kartici, snimanje podataka na nju ne bi bilo moguće.

Eksterna kartica predstavlja dopunski element priključen na hardver mobilnog uređaja. Kao takvom, moraju mu se obezbediti mehanizmi pristupa i manipulacije njegovim sadržajem. U tu svrhu, u AndroidManifest.xml datoteci, neophodno je dodati WRITE_EXTERNAL_STORAGE privilegiju za snimanje podataka u eksternu memoriju. U nastavku je priložen kod AndroidManifest.xml datoteke.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.metropolitan.datotekedemo">

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
```

```
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Nakon definisanja privilegija za pristup eksternoj kartici mobilnog uređaja, neophodno je još prikazati i glavnu klasu aktivnosti koja je modifikovana na način da upravlja tekstualnom datotekom, i njenim sadržajem, sačuvanoj na eksternoj kartici mobilnog uređaja (u konkretnom slučaju simulirano emulatorom).

```
package com.metropolitan.datotekedemo;

import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {

    EditText textBox;
    static final int READ_BLOCK_SIZE = 100;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textBox = (EditText) findViewById(R.id.txtText1);

        InputStream is = this.getResources().openRawResource(R.raw.textfile);
        BufferedReader br = new BufferedReader(new InputStreamReader(is));
```

```
        String str = null;
        try {
            while ((str = br.readLine()) != null) {
                Toast.makeText(getBaseContext(),
                        str, Toast.LENGTH_SHORT).show();
            }
            is.close();
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public void onClickSave(View view) {
        String str = textBox.getText().toString();
        try
        {    //---stari kod---
           /*  FileOutputStream fOut = openFileOutput         ("textfile.txt",
MODE_WORLD_READABLE);*/
            //---Novi kod---
            File sdCard = Environment.getExternalStorageDirectory();
            File directory = new File (sdCard.getAbsolutePath() +
                    "/MyFiles");
            directory.mkdirs();
            File file = new File(directory, "textfile.txt");
            FileOutputStream fOut = new FileOutputStream(file);


            OutputStreamWriter osw = new
                    OutputStreamWriter(fOut);

            //---Upisivanje stringa u datoteku---
            osw.write(str);
            osw.flush();
            osw.close();

            //---Poruka o snimljenom podatku---
            Toast.makeText(getBaseContext(),
                    "Datoteka je uspešno snimljena!",
                    Toast.LENGTH_SHORT).show();

            //---Očisti EditText pogled---
            textBox.setText("");
        }
        catch (IOException ioe)
        {
            ioe.printStackTrace();
        }
    }

    public void onClickLoad(View view) {
        try
        {   //---stari kod---
          /*FileInputStream fIn = openFileInput("textfile.txt");
```

```
InputStreamReader isr = new InputStreamReader(fIn);*/
            //---Novi kod--
            File sdCard = Environment.getExternalStorageDirectory();
            File directory = new File (sdCard.getAbsolutePath() +
                    "/MyFiles");
            File file = new File(directory, "textfile.txt");
            FileInputStream fIn = new FileInputStream(file);
            InputStreamReader isr = new InputStreamReader(fIn);


            char[] inputBuffer = new char[READ_BLOCK_SIZE];
            String s = "";
            int charRead;
            while ((charRead = isr.read(inputBuffer))>0)
            {//---Konverzija niza karaktera u string---
                String readString =
                        String.copyValueOf(inputBuffer, 0,
                                charRead);
                s += readString;
                inputBuffer = new char[READ_BLOCK_SIZE];
            }
            //---Podešavanje EditText na učitani tekst
            textBox.setText(s);

            Toast.makeText(getBaseContext(),
                    "Datoteka je uspešno učitana!",
                    Toast.LENGTH_SHORT).show();
        }
        catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }

}
```

IZBOR OPTIMALNE OPCIJE ZA SNIMANJE

Za izbor načina čuvanja podataka u Android aplikacijama neophodno je pratiti neke prihvaćene preporuke.

U prethodnim izlaganjima obrađeno je čuvanje podataka u Android aplikacijama na tri načina. Korišćene su deljene preferencije, unutrašnja i spoljašnja memorijska skladišta Android uređaja. Da bi bio izabran pravi način čuvanja podataka, u Android aplikacijama, neophodno je poštovati određene preporuke izbora:

- Ukoliko se manipuliše podacima koji mogu da se prikažu u obliku para (naziv, vrednost) biće korišćene deljene preferencije. Na primer, deljenjim preferencijama moguće je sačuvati sledeće parove vrednosti: (korisnik, datum rođenja), (korisnik, vreme prijavljivanja na sistem), (pozadina, boja pozadine), (zvono, melodija zvona) itd.

- Ukoliko je neophodno brzo snimanje podataka poput preuzimanja slika sa web stranice, da bi ih neka ugrađena aplikacija naknadno prikazala, interna memorija mobilnog uređaja je dobar izbor za čuvanje podataka.
- U situacijama kada je neophodno olakšano razmenjivanje podataka sa drugim korisnicima, i kada je neophodno rasteretiti resurse unutrašnje memorije, trebalo bi koristiti skladište kao što je SD kartica za čuvanje podataka.

UPOTREBA STATIČKIH RESURSA

Moguće je koristiti i datoteke koje nisu kreirane dinamički, u aplikaciji, za čuvanje podataka.

Pored datoteka koje se dinamički generišu aplikacijom, podaci, u Android aplikacijama, mogu biti čuvani i preuzimani iz datoteka koje su dodate manuelno u paket aplikacije. U konkretnom primeru, u folderu res/raw projekta, ubačena je datoteka sa nazivomtextfile.txt u kojoj je sačuvan string koji odgovara nazivu našeg Univerizteta. Da bi ova datoteka bila iskorišćena, neophodno je uključiti getResources*()* metodu, klase *aktivnosti aplikacije,* koja vraća objekat tipa Resources(). Nakon toga, primenjuje se metoda openRawResource() sa ciljem otvaranja željene datoteke. Sledećim listingom, izdvojenim iz glavne klase aktivnosti, su prikazani neophodni paketi sa klasama i kod koji je neophodno implementirati u onCreate()metodu.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;

-----------------------------------------------------

InputStream is = this.getResources().openRawResource(R.raw.textfile);
BufferedReader br = new BufferedReader(new InputStreamReader(is));
String str = null;
        try {
            while ((str = br.readLine()) != null) {
                Toast.makeText(getBaseContext(),
                        str, Toast.LENGTH_SHORT).show();
            }
            is.close();
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Identifikator resursa, koji je smešten u res/raw folder, dobija naziv na osnovu naziva datoteke bez odgovarajuće ekstenzije. U konkretnom slučaju to je R.raw.textfile. Sledećom slikom je pokazano preuzimanje podataka aplikacijom iz statičke datoteke i njena lokacija u hijerarhiji projekta.

Slika 2.4 Lokacija i podatak iz datoteke

PRIMER 2 - ČUVANJE PODATAKA POMOĆU TEKSTUALNIH DATOTEKA

Vežbanje rada sa datotekama

Zadatak:

Kreirati Android aplikaciju koja omogućava čitanje i pisanje tekstualnih podataka u datotekama. Realizovati rad sa datotekama iz unutrašnje memorije, kao i sa spoljašnje. U slučaju rada sa spoljašnjim resursima, ugraditi odgovarajuću dozvolu u AndroidManifest.xml.

Sledećim kodom je data GUI datoteka:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Unesite neki tekstualni podatak!" />

    <EditText
        android:id="@+id/txtText1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btnSave"
        android:text="Snimite"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickSave" />

    <Button
        android:id="@+id/btnLoad"
        android:text="Učitajte"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickLoad" />
</LinearLayout>
```

Glavna datoteka, snabdevena kodom za rad sa datotekama sa unutrašnje i spoljašnje memorije, data je sledećim kodom:

```
package com.metropolitan.datotekedemo;

import android.os.Bundle;
import android.os.Environment;
```

```
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {

    EditText textBox;
    static final int READ_BLOCK_SIZE = 100;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textBox = (EditText) findViewById(R.id.txtText1);

        InputStream is = this.getResources().openRawResource(R.raw.textfile);
        BufferedReader br = new BufferedReader(new InputStreamReader(is));

        String str = null;
        try {
            while ((str = br.readLine()) != null) {
                Toast.makeText(getBaseContext(),
                        str, Toast.LENGTH_SHORT).show();
            }
            is.close();
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public void onClickSave(View view) {
        String str = textBox.getText().toString();
        try
        {    //---stari kod---
           /*  FileOutputStream fOut =
                    openFileOutput("textfile.txt",
                            MODE_WORLD_READABLE);*/
            //---Novi kod---
            File sdCard = Environment.getExternalStorageDirectory();
            File directory = new File (sdCard.getAbsolutePath() +
                    "/MyFiles");
            directory.mkdirs();
```

```
            File file = new File(directory, "textfile.txt");
            FileOutputStream fOut = new FileOutputStream(file);


            OutputStreamWriter osw = new
                    OutputStreamWriter(fOut);

            //---Upisivanje stringa u datoteku---
            osw.write(str);
            osw.flush();
            osw.close();

            //---Poruka o snimljenom podatku---
            Toast.makeText(getBaseContext(),
                    "Datoteka je uspešno snimljena!",
                    Toast.LENGTH_SHORT).show();

            //---Očisti EditText pogled---
            textBox.setText("");
        }
        catch (IOException ioe)
        {
            ioe.printStackTrace();
        }
    }

    public void onClickLoad(View view) {
        try
        {   //---stari kod---
          /*FileInputStream fIn =
                    openFileInput("textfile.txt");
            InputStreamReader isr = new
                    InputStreamReader(fIn);*/
            //---Novi kod--
            File sdCard = Environment.getExternalStorageDirectory();
            File directory = new File (sdCard.getAbsolutePath() +
                    "/MyFiles");
            File file = new File(directory, "textfile.txt");
            FileInputStream fIn = new FileInputStream(file);
            InputStreamReader isr = new InputStreamReader(fIn);


            char[] inputBuffer = new char[READ_BLOCK_SIZE];
            String s = "";
            int charRead;
            while ((charRead = isr.read(inputBuffer))>0)
            {//---Konverzija niza karaktera u string---
                String readString =
                        String.copyValueOf(inputBuffer, 0,
                                charRead);
                s += readString;
                inputBuffer = new char[READ_BLOCK_SIZE];
            }
```

```
            //---Podešavanje EditText na učitani tekst
            textBox.setText(s);

            Toast.makeText(getBaseContext(),
                    "Datoteka je uspešno učitana!",
                    Toast.LENGTH_SHORT).show();
        }
        catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }

}
```

AndroidManifest.xml sa odgovarajućom dozvolom, za korišćenje eksterne memorijske kartice, data je sledećim kodom:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.metropolitan.datotekedemo">

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

ZADATAK 2 - KREIRAJTE APLIKACIJU PODSETNIK TEKSTA PESME

Pokušajte sami!!!

Pokušajte da kreirate aplikaciju koja će biti vaš podsetnik za izbor pesme čijeg teksta ne možete da se setite. Nekoliko naziva pesama dodajete u listu odakle ih birate, a odgovarajući tekst se učitava i prikazuje iz tekstualne datoteke.

Poglavlje 3

Primena baza podataka

UVOD U ANDROID BAZE PODATAKA

Za snimanje relacionih podataka u Android aplikacijama, biće korišćene baze podataka.

Tehnike prikazane u prethodnim delovima lekcije bavile su se čuvanjem skupova podataka. Ukoliko se žele sačuvati podaci, koji su različitog tipa, mogu biti tabelarno prikazani i povezani odgovarajućim relacijama, koristiće se baze podataka. Na primer, potrebno je kreirati Android aplikaciju koja će obraditi i sačuvati rezultate nekog ispita. Mnogo je efikasnije koristiti bazu podataka za čuvanje i prikazivanje podataka iz razloga što je nad njom moguće kreirati brojne upite, a sa ciljem dobijanja konkretnih podataka u vezi sa studentima koji su polagali ispit. Takođe, baze podataka obezbeđuju integritet podataka kroz specificiranje veza između različitih tabela.

Android operativni sistem podržava *SQLite* sistem za upravljanje bazom podataka. Ovde je potrebno napomenuti da baza podataka koja je kreirana za određenu Android aplikaciju, može da se koristi isključivo u toj aplikaciji i druge Android aplikacije nemaju pristup navedenoj bazi podataka.

Pored poznatih tehnika upravljanja bazom podataka, u ovom delu lekcije biće prikazani i konkretni softverski alati za kreiranje i testiranje baze podatak izvan Android aplikacije. Alatom SQLite Browser je moguće kreirati bazu podataka, testirati je upitima i posle toga na pogodan način integrisati u kreiranu Android aplikaciju. O načinima integrisanja ovako kreirane baze podataka u Android aplikaciju biće posebno reči u ovom delu lekcije.

KREIRANJE KLASE DBADAPTER

Prilikom rada sa bazama podataka, praksa je kreiranje pomoćne klase koja enkapsulira postupak pristupa podacima.

U daljem izlaganju, cilj je pokazivanje načina kreiranja *SQLite* baze podataka u Android aplikaciji. Kreirana baza podataka, u Android operativnom sistemu, uvek se nalazi, za datu aplikaciju, u folderu *data/data/nazivPaketa/databases. Navigacija do ove lokacijeje moguća u DDMS prikazu, ako se koristi emulator za testiranje aplikacije, ili primenom nekog menadžera datoteka instaliranog na realnom mobilnom uređaju.*

Dobra praksa, u radu sa Android bazama podataka, jeste kreiranje pomoćne klase koja enkapsulira veoma složen postupak pristupa podacima. Iz navedenog razloga, biće kreirana

klasa *DBAdapter* koja će omogućiti kreiranje, otvaranje i zatvaranje baze podataka, kao i sve poznate načine manipulacije nad podacima pohranjenim u bazi.

Za potrebe demonstracije, biće uveden primer koji podrazumeva kreiranje baze podataka sa jednom tabelom pod nazivom *kontakti*. Tabela će biti izgrađena od tri kolone: *_id, ime* i *email* (sledeća slika). Bazu podataka će se zvati MyDB.

Slika 3.1 Tabela kreirane baze podataka

KREIRANJE KLASE DBADAPTER – JAVA KOD

U oviru JAVA koda, kao stringovi, biće implementirani i upiti za izvršavanje određenih akcija nad bazom podataka.

Sledeći zadatak predstavlja kreiranje JAVA koda pomoćne klase DBAdapter.java. Klasom će biti definisano: kreiranje, zatvaranje i otvaranje baze, učitavanje i upisivanje podataka, a biće implementirani i određeni upiti za izvršavanje određenih akcija nad bazom podataka. Lokacija klase biće standardi folder projekta *src*.

Kod klase DBAdapter.java biće prikazan sledećim listingom.

```
package com.metropolitan.dbdemo;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

/**
 * Created by Vladimir Milicevic on 4.11.2016..
 */

public class DBAdapter {
    static final String KEY_ROWID = "_id";
    static final String KEY_NAME = "ime";
    static final String KEY_EMAIL = "email";
    static final String TAG = "DBAdapter";

    static final String DATABASE_NAME = "MyDB";
    static final String DATABASE_TABLE = "kontakti";
    static final int DATABASE_VERSION = 2;

    static final String DATABASE_CREATE =
            "create table kontakti (_id integer primary key autoincrement, "
                    + "ime text not null, email text not null);";

    final Context context;
```

```
    DatabaseHelper DBHelper;
    SQLiteDatabase db;

    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper
    {
        DatabaseHelper(Context context)
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db)
        {
            try {
                db.execSQL(DATABASE_CREATE);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
        {
            Log.w(TAG, "Ažuriranje verzije baze podataka sa " + oldVersion + " na
verziju "
                    + newVersion + ", a to će uništiti postojeće podatke");
            db.execSQL("DROP TABLE IF EXISTS kontakti");
            onCreate(db);
        }
    }

    //---otvaranje baze podataka---
    public DBAdapter open() throws SQLException
    {
        db = DBHelper.getWritableDatabase();
        return this;
    }

    //---zatvaranje baze podataka---
    public void close()
    {
        DBHelper.close();
    }

    //---umetanje kontakta u bazu---
    public long insertContact(String ime, String email)
    {
```

```
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_NAME, ime);
        initialValues.put(KEY_EMAIL, email);
        return db.insert(DATABASE_TABLE, null, initialValues);
    }

    //---brisanje konkretnog kontakta---
    public boolean deleteContact(long rowId)
    {
        return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
    }

    //---preuzima sve kontakte---
    public Cursor getAllContacts()
    {
        return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
                KEY_EMAIL}, null, null, null, null, null);
    }

    //---preuzima konkretan kontakt---
    public Cursor getContact(long rowId) throws SQLException
    {
        Cursor mCursor =
                db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
                                KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
                        null, null, null, null);
        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;
    }

    //---ažurira kontakt---
    public boolean updateContact(long rowId, String ime, String email)
    {
        ContentValues args = new ContentValues();
        args.put(KEY_NAME, ime);
        args.put(KEY_EMAIL, email);
        return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId, null) > 0;
    }

}
```

KLASA DBADAPTER - FUNKCIONISANJE

Primenom klase Cursor omogućeno je efikasnije upravljanje vrstama i kolonama.

Na samom početku klase kreirano je nekoliko konstanti koje se odnose na polja tabele koja će biti kreirana u bazi podataka, npr. ***static final StringKEY_ROWID = "_id".*** Posebno je konstantom ***DATABASE_CREATE*** predstavljena *SQL* naredba za kreiranje tabele *kontakti*.

U nastavku, u klasu DBAdapter ugrađena je privatna klasa DataBaseHelper, naslednica klase SQLiteOpenHelper, čiji je zadatak asistiranje Android operativnom sistemu prilikom kreiranja baze podataka i upravljanja njenim verzijama.

Metoda onCreate() kreiraće novu bazu podataka, ukoliko ona nije prisutna u trenutku poziva metode. Ukoliko je neophodno poboljšati bazu podataka, metoda onUpgrade() biće pozvana. Za proveru verzije baze podataka, koju je neophodno poboljšati, metoda proverava konstantu ***DATABASE_VERSION.***

U daljem radu su kreirane metode za otvaranje i zatvaranje baze podataka, kao i za dodavanje, brisanje i ažuriranje vrsta u tabeli *kontakti* (videti priloženi kod klase *DBAdapter*).

Posebnu pažnju je neophodno obratiti na klasu Cursor koju Android operativni sistem koristi kao povratnu vrednost za upit.Kursor predstavlja tip pokazivača na skup dobijen izvršavanjem upita nad bazom podataka.

Primena ove klase, u Android operativnom sistemu, omogućava efikasnije upravljanje vrstama i kolonama tabele.

Za snimanje podataka u bazu, u formi para (naziv vrednost), korišćen je ContentValue objekat koji, primenom metodeput(), vrši snimanje kao na primer, initialValues.put (KEY_IME, ime).

Konačno, da bi kreiranje baze podatak bilo moguće, u aplikaciji je neophodno kreirati objekat klase *DBAdapter* primenom konstuktora sa sledeće slike. Ovaj konstruktor kreira objekat klase *DatabaseHelper* da bi kreirao novi bazu podataka.

```
public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper
    {
        DatabaseHelper(Context context)
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
```

PROGRAMSKO MANIPULISANJE BAZOM PODATAKA - CRUD OPERACIJE

Nad SQLite bazom podataka izvršavaju se standardne CRUD operacije.

Nakon kreiranja *DBAdapter* pomoćne klase sve je spremno za rad sa bazom podataka. U nastavku lekcije biće pokazano kako se nad SQLite bazom podataka izvršavaju se

standardneCRUD operacije (create, read, update i delete). U tu svrhu, neophodno je bilo kreirati klasu aktivnosti aplikacije koja će zadržati podrazumevani naziv MainActivity.java.

Za dodavanje novih kontakata u bazu podataka, neophodno je u onCreate()metodu, klase aktivnosti, dodati kod prikazan sledećim listingom.

```
/** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        DBAdapter db = new DBAdapter(this);


        //---dodavanje kontakata---
        db.open();
        long id = db.insertContact("vlada", "v@gmail.com");
        id = db.insertContact("pera", "p@gmail.com");
        id = db.insertContact("maja", "m@gmail.com");
        db.close();

}
```

Iz koda se vidi da je prvo kreiran objekat klase *DBAdapter:*

DBAdapter db = new DBAdapter(this).

Objekat poziva metodu insertContact() koja vraća identifikator unete vrste. Ukoliko se, prilikom izvršavanja metode, javi greška, vraća se vrednost -1. Pokretanjem aplikacije, uređajem ili emulatorom, moguće je uočiti, primenom DDMS i File Explorer-a, da je baza podataka *MyDB* kreirana u *databases* folderu (sledeća slika).

Slika 3.2 Folder databases

CRUD OPERACIJE – UČITAVANJE KONTAKATA

Implementacijom metode getAllContacts(), DBAdapter klase, prikazuju se svi kontakti.

Da bi svi kontakti ili pojedinačni, iz kreirane *MyDB* baze podataka, bili učitani, neophodno je pozvati metode getAllContacts()i getContacts() klase DBAdapter (izdvojeno u prvom listingu). Takođe, neophodno je implementirati metodu DisplayContact()(izdvojeno u drugom listingu), koja preuzima objekat tipa Cursor, za prikazivanje pojedinačnih kontakata. Prikazivanje kontakata funkcioniše na sledeći način:

- Metoda *getAllContacts()* klase *DBAdapter* učitava sve kontakte koji se nalaze u bazi podataka;
- Rezultat učitavanja se vraća kao *Cursor* objekat;
- *Cursor* objekat izvršava metodu *moveToFirst()*;

- Ukoliko je izvršavanje metode uspešno, sledeća, *DisplayContact()*, metoda prikazaće konkretan kontakt.
- Metoda *moveToNext()* pomera pokazivač na sledeći kontakt.

```
//--preuzimanje svih kontakata---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayContact(c);
            } while (c.moveToNext());
        }
        db.close();



        //---preuzimanje jednog kontakta---
        db.open();
        Cursor d = db.getContact(2);
        if (d.moveToFirst())
            DisplayContact(d);
        else
            Toast.makeText(this, "Nije pronađen kontakt",
                    Toast.LENGTH_LONG).show();
        db.close();
```

Sledećim listingom prikazan je kod pomenute metode DisplayContact(), glavne klase aktivnosti, čiji je zadatak prikazivanje kontakata iz baze podataka, primenom Toast klase, na ekranu mobilnog uređaja.

```
public void DisplayContact(Cursor c)
    {
        Toast.makeText(this,
                "id: " + c.getString(0) + "\n" +
                        "ime: " + c.getString(1) + "\n" +
                        "Email:  " + c.getString(2),
                Toast.LENGTH_LONG).show();
    }
```

Slika 3.3 Učitavanje kontakata iz baze podataka

CRUD OPERACIJE – AŽURIRANJE KONTAKATA

Ažuriranje kontakata se obavlja metodom updateContact() DBAdapter klase.

Da bi podatak, u konkretnom slučaju kontakt, bio ažuriran, neophodno je izvršiti metodu updateContact() klase DBAdapter. Ova metoda ažurira detalje koji se odnose na kontakt

upotrebom identifikacionog broja koji odgovara kontaktu koji se ažurira. Sledećim listingom prikazan je detalj metode onCreate() kojim je omogućeno ažuriranje baze kontakata.

```
//---ažuriranje kontakta---
       db.open();
        if (db.updateContact(2, "Milan", "mil@gmail.com"))
            Toast.makeText(this, "Ažuriranje je uspešno.",
                    Toast.LENGTH_LONG).show();
        else
            Toast.makeText(this, "Gređka u ažuriranju.",
                    Toast.LENGTH_LONG).show();
        db.close();
```

Pokretanjem programa, sa ugrađenim priloženim kodom, dobija se informacija o uspešnosti ažuriranja konkretnog kontakta, a potom se izlistava ažurirana lista kontakata. Navedeno je moguće sagledati uvidom u sledeću priloženu sliku.

Slika 3.4 Ažuriranje kontakata u bazi podataka

CRUD OPERACIJE – BRISANJE KONTAKATA IZ BAZE PODATAKA

Metoda deleteContact() koristi identifikacioni broj kontakta za njegovo brisanje iz baze podataka.

Za brisanje kontakata, iz kreirane baze podataka, neophodno je koristiti novu metodu klase DBAdapter. Metoda ima naziv deleteContact() i koristi identifikacioni broj kontakta za njegovo lociranje i brisanje iz baze podataka. Metoda je logičkog tipa i vraća vrednost *true*, ukoliko je brisanje bilo uspešno, ili *false* ukoliko je iz nekog razloga brisanje ciljanog kontakta izostalo. U metodi onCreate(), neophodno je obezbediti sledeći kod kojim se omogućava brisanje kontakata iz baze podataka.

```
//---brisanje kontakta---
        db.open();
        if (db.deleteContact(2))
            Toast.makeText(this, "Brisanje je bilo uspešno.",
                    Toast.LENGTH_LONG).show();
        else
            Toast.makeText(this, "Brisanje je bilo neuspešno.",
                    Toast.LENGTH_LONG).show();
        db.close();
```

Ako se pokrene program, sa navedenim modifikacijama u klasi aktivnosti, dobija se informacija o uspešnosti brisanja iz baze (sledeća slika) i prikazuje se nova, ažurirana, lista kontakata.

Slika 3.5 Informacija o uspešnom brisanju kontakta iz baze podataka

POBOLJŠANJE BAZE PODATAKA

U LogCat prozoru vide se rezultati promene verzije baze podataka.

Kako se razvija aplikacija, tako može doći i do potrebe za unapređenjem postojeće baze podataka kroz dodavanje novih tabela, promenu šeme, izmenu u definiciji tabela dodavanjem novih kolona itd. U tom slučaju je neophodno izvršiti migraciju podataka iz stare baze podataka u novu.

Da bi baza podataka bila unapređena, neophodno je promeniti vrednost konstante***DATABASE_VERSION*** dodeljujući joj novu, veću, vrednost u odnosu na prethodnu bazu podataka. Za aktuelnu bazu podataka vrednost ove konstante je iznosila 2. Ta vrednost biće promenjena i program će izvršiti unapređenje baze (pogledati kod priložen sledećim listingom).

```
/**
 * Created by Vladimir Milicevic on 4.11.2016..
 */

public class DBAdapter {
    static final String KEY_ROWID = "_id";
    static final String KEY_NAME = "ime";
    static final String KEY_EMAIL = "email";
    static final String TAG = "DBAdapter";

    static final String DATABASE_NAME = "MyDB";
    static final String DATABASE_TABLE = "kontakti";
    static final int DATABASE_VERSION = 3;
```

Kada je baza podataka unapređena, odnosno kada se pokrene aplikacija koja obavlja taj zadatak, u *LogCat* prozoru razvojnog okruženja Android Studio IDE ili Eclipse *IDE*, moguće je videti rezultate koji govore da li je došlo do promene verzije baze podataka (sledeća slika).

Zbog lakšeg razumevanja problematike, ovde je samo uklonjena postojeća i kreirana nova tabela. U realnim uslovima, obično se prvo radi *backup* postojeće tabele, a zatim njeno *prepisivanje* u novu.

Slika 3.6 Promena verzije baze podataka

SQLITE ALATI

Za rad sa SQLite bazom podataka moguće je koristiti i brojne alate.

Za kreiranje SQLite baze podataka i za sve poznate operacije nad bazom i njenim tabelama, moguće je koristiti neki od brojnih alata koji su dostupni za preuzimanje preko Interneta. Jedan od njih je *SQLite Database Browser* koji je potpuno besplatan i veoma jednostavan za korišćenje (sledeća slika). Ovaj alat je moguće preuzeti sa lokacije http://sqlitebrowser.org/.

SQLite Database Browser omogućava potpuno vizuelno kreiranje baze podataka, upita nad bazom i tabelama za dodavanje novih podataka, ažuriranje starih, brisanje polja, tabela i baze i tako dalje.

Slika 3.7 SQLite DataBase Browser

Na veoma jednostavan način moguće je kreirati i izvršiti željeni upit primenom ovog alata. U posebnom polju se otkuca željeni upit nad bazom podataka, a onda klikom na Execute SQL, kreirani upit se izvršava, pri čemu su rezultati izvršavanja ovog upita direktno prezentovani korisniku SQLite Database Browser aplikacije.

Navedeno je prikazano sledećom slikom.

Slika 3.8 Kreiranje i izvršavanje upita nad bazom podataka primenom alata SQLite Database Browser

OBJEDINJAVANJE BAZE PODATAKA

U assets folderu imena datoteka baza podataka sastoje se od malih slova.

U Android Studio IDE razvojnom oruženju, za tekući projekat, u folderu assets biće snimljena datoteka *SQLite* baze podataka pod nazivom mydb (pravilo je sa u ovom folderu nazivi se grade isključivo malim slovima). Ovu baza podataka je mogla da bude kreirana prethodno prikazanim alatom za upravljanje bazom podataka pod nazivom SQLite Database Browser.

Sledećom slikom prikazan je položaj snimljene datoteke baze podataka u hijerarhiji projekta koji je predmet demonstracije.

Slika 3.9 Baza podaka u hijerarhiji projekta

U klasi aktivnosti aplikacije, odnosno njenoj onCreate() metodi neophodno je uvesti modifikacije predstavljene sledećim kodom. Takođe, neophodno je kreirati metodu copyDB().

```
package com.metropolitan.dbdemo;

import android.database.Cursor;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class MainActivity extends AppCompatActivity {
```

```
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        DBAdapter db = new DBAdapter(this);

try {
            String destPath = "/data/data/" + getPackageName() +
                    "/databases";
            File f = new File(destPath);
            if (!f.exists()) {
                f.mkdirs();
                f.createNewFile();
                //---kopira db iz assets foldera u databases folder---
                CopyDB(getBaseContext().getAssets().open("mydb"),
                        new FileOutputStream(destPath + "/MyDB"));
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        //---preuzimanje svih kontakata---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayContact(c);
            } while (c.moveToNext());
        }
        db.close();
    }

    public void CopyDB(InputStream inputStream,
                       OutputStream outputStream) throws IOException {

        //---kopira 1KB u datom trenutku---
        byte[] buffer = new byte[1024];
        int length;
        while ((length = inputStream.read(buffer)) > 0) {
            outputStream.write(buffer, 0, length);
        }
        inputStream.close();
        outputStream.close();
    }

    public void DisplayContact(Cursor c)
    {
        Toast.makeText(this,
```

```
                "id: " + c.getString(0) + "\n" +
                        "ime: " + c.getString(1) + "\n" +
                        "Email:  " + c.getString(2),
                Toast.LENGTH_LONG).show();
    }

}
```

OBJEDINJAVANJE BAZE PODATAKA - FUNKCIONISANJE

Metodom copyDB() kopira se datoteka baze podataka sa jedne lokacije na drugu.

U prvom trenutku, kreirana je metoda copyDB()čiji je zadatak prebacivanje datoteke baze podataka na drugu lokaciju. Iz priloženog koda se vidi da su iskorišćeni InputStream objekat za učitavanje iz izvorne datoteke i OutputStreamobjekat za upisivanje učitanih podataka u odredišnu datoteku.

U nastavku, datoteka iz assets foldera kopirana je u odgovarajući folder Android uređaja ili emulatora. Navedeno je prikazano sledećim izolovanim listingom iz klase aktivnosti projekta.

```
try {
            String destPath = "/data/data/" + getPackageName() +
                    "/databases";
            File f = new File(destPath);
            if (!f.exists()) {
                f.mkdirs();
                f.createNewFile();
                //---kopira db iz assets foldera u databases folder---
                CopyDB(getBaseContext().getAssets().open("mydb"),
                        new FileOutputStream(destPath + "/MyDB"));
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

Važno je napomenuti da se ovaj način prebacivanja datoteke baze podataka obavlja isključivo u slučaju kada na odredištu ne postoji baza podataka. U suprotnom, datoteka na odredištu bi bila prebrisana izvornom datotekom, prilikom svakog iniciranja aktivnosti metodom onCreate(). To praktično znači da bi svi podaci, koji su sačuvani u bazi podataka tokom izvršavanja aplikacije, bili uništeni prilikom ponovnog iniciranja aktivnosti. Za proveravanje da li je datoteka baze podataka iskopirana, neophodno je koristiti već pokazani DDMS prikaz i *File Explorer*.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 3 - ČUVANJE PODATAKA POMOĆU BAZA PODATAKA

Vežbanje rada sa bazama podataka.

Zadatak:

Kreirati Android aplikaciju koja omogućava izvođenje CRUD operacija nad bazom podataka. Kreirati klasu DBAdapter za implementaciju CRUD funkcionalnosti.

GUI aplikacije može biti dat sledećim listingom:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="SQLite baza podataka - primer!" />

</LinearLayout>
```

Tražena klasa DBAdapter.java može biti definisana sledećim kodom:

```
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

/**
 * Created by Vladimir Milicevic on 4.11.2016..
 */

public class DBAdapter {
    static final String KEY_ROWID = "_id";
    static final String KEY_NAME = "ime";
    static final String KEY_EMAIL = "email";
    static final String TAG = "DBAdapter";

    static final String DATABASE_NAME = "MyDB";
    static final String DATABASE_TABLE = "kontakti";
    static final int DATABASE_VERSION = 3;

    static final String DATABASE_CREATE =
            "create table kontakti (_id integer primary key autoincrement, "
                    + "ime text not null, email text not null);";
```

```
    final Context context;

    DatabaseHelper DBHelper;
    SQLiteDatabase db;

    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper
    {
        DatabaseHelper(Context context)
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db)
        {
            try {
                db.execSQL(DATABASE_CREATE);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
        {
            Log.w(TAG, "Ažuriranje verzije baze podataka sa " + oldVersion + " na
verziju "
                    + newVersion + ", a to će uništiti postojeće podatke");
            db.execSQL("DROP TABLE IF EXISTS kontakti");
            onCreate(db);
        }
    }

    //---otvaranje baze podataka---
    public DBAdapter open() throws SQLException
    {
        db = DBHelper.getWritableDatabase();
        return this;
    }

    //---zatvaranje baze podataka---
    public void close()
    {
        DBHelper.close();
    }
```

```
    //---umetanje kontakta u bazu---
    public long insertContact(String ime, String email)
    {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_NAME, ime);
        initialValues.put(KEY_EMAIL, email);
        return db.insert(DATABASE_TABLE, null, initialValues);
    }

    //---brisanje konkretnog kontakta---
    public boolean deleteContact(long rowId)
    {
        return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
    }

    //---preuzima sve kontakte---
    public Cursor getAllContacts()
    {
        return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
                KEY_EMAIL}, null, null, null, null, null);
    }

    //---preuzima konkretan kontakt---
    public Cursor getContact(long rowId) throws SQLException
    {
        Cursor mCursor =
                db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
                                KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
                        null, null, null, null);
        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;
    }

    //---ažurira kontakt---
    public boolean updateContact(long rowId, String ime, String email)
    {
        ContentValues args = new ContentValues();
        args.put(KEY_NAME, ime);
        args.put(KEY_EMAIL, email);
        return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId, null) > 0;
    }

}
```

Glavna klasa aktivnosti je data sledećim kodom:

```
package com.metropolitan.dbdemo;

import android.database.Cursor;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
```

```
import android.widget.Toast;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        DBAdapter db = new DBAdapter(this);


      /*  //---dodavanje kontakata---
        db.open();
        long id = db.insertContact("vlada", "v@gmail.com");
        id = db.insertContact("pera", "p@gmail.com");
        id = db.insertContact("maja", "m@gmail.com");
        db.close();*/



       /* //--preuzimanje svih kontakata---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayContact(c);
            } while (c.moveToNext());
        }
        db.close();*/



     /*   //---preuzimanje jednog kontakta---
        db.open();
        Cursor d = db.getContact(2);
        if (d.moveToFirst())
            DisplayContact(d);
        else
            Toast.makeText(this, "Nije pronađen kontakt",
                    Toast.LENGTH_LONG).show();
        db.close();*/
```

```
    /* //---ažuriranje kontakta---
    db.open();
     if (db.updateContact(2, "Milan", "mil@gmail.com"))
         Toast.makeText(this, "Ažuriranje je uspešno.",
                 Toast.LENGTH_LONG).show();
     else
         Toast.makeText(this, "Greška u ažuriranju.",
                 Toast.LENGTH_LONG).show();
     db.close();*/



    /* //---brisanje kontakta---
     db.open();
     if (db.deleteContact(2))
         Toast.makeText(this, "Brisanje je bilo uspešno.",
                 Toast.LENGTH_LONG).show();
     else
         Toast.makeText(this, "Brisanje je bilo neuspešno.",
                 Toast.LENGTH_LONG).show();
     db.close();
 */

     try {
         String destPath = "/data/data/" + getPackageName() +
                 "/databases";
         File f = new File(destPath);
         if (!f.exists()) {
             f.mkdirs();
             f.createNewFile();
             //---kopira db iz assets foldera u databases folder---
             CopyDB(getBaseContext().getAssets().open("mydb"),
                     new FileOutputStream(destPath + "/MyDB"));
         }
     } catch (FileNotFoundException e) {
         e.printStackTrace();
     } catch (IOException e) {
         e.printStackTrace();
     }

     //---preuzimanje svih kontakata---
     db.open();
     Cursor c = db.getAllContacts();
     if (c.moveToFirst())
     {
         do {
             DisplayContact(c);
         } while (c.moveToNext());
     }
     db.close();
 }
```

```
    public void CopyDB(InputStream inputStream,
                       OutputStream outputStream) throws IOException {

        //---kopira 1KB u datom trenutku---
        byte[] buffer = new byte[1024];
        int length;
        while ((length = inputStream.read(buffer)) > 0) {
            outputStream.write(buffer, 0, length);
        }
        inputStream.close();
        outputStream.close();
    }

    public void DisplayContact(Cursor c)
    {
        Toast.makeText(this,
                "id: " + c.getString(0) + "\n" +
                        "ime: " + c.getString(1) + "\n" +
                        "Email:  " + c.getString(2),
                Toast.LENGTH_LONG).show();
    }


}
```

ZADATAK 3 - KREIRAJTE SAMOSTALNO ANDROID APLIKACIJU NAD BAZOM PODATAKA

Pokušajte sami

Koristeći pokazne primere iz ovog dela lekcije, kreirajte Android CRUD aplikaciju nad bazom podataka koja ima jednu tabelu prikazanu sledećom slikom.

Slika

11 - Tabela iz baze podataka

Poglavlje 4

Domaći zadatak 5

ZADATAK 1

Zadatak ima za cilj vežbanje kreiranja Android aplikacija sa bazama podataka.

Izvršiti sledeće zadatke:

1. Instalirati alat *SQLite Database Browser;*
2. Pokrenuti alat i izvršiti upite za kreiranje nove baze podataka *(ispit)* i njene tabele *(studenti);*
3. Tabela student sadrži kolone: broj_indeksa (PK), ime, broj_bodova;
4. Kreirati novi Android projekat IspitIzProgramiranja;
5. Kopirati datoteku kreirane baze podataka u folder *assets* projekta;
6. Kreirati main.xml datotekom UI koji predviđa dugmad za CRUD operacije nad kreiranom bazom podataka;
7. Čuvati u bazi podataka ažurne podatke o rezultatu ispita iz programiranja, nakon primenjenih CRUD operacija;
8. Omogućiti prikazivanje ažuriranih rezultata ispita.

Pregled Lekcije06

PREGLED LEKCIJE 05

Lekcija je pokazala tri načina čuvanja podataka u Android aplikacijama.

U lekciji su naučeni različiti načini kako se čuvaju podaci u Android aplikacijama. Jednostavne, nestruktuirani podaci, primenom *SharedPreferences* objekta veoma se jednostavno čuvaju u formi para (naziv, vrednost). Ukoliko se čuvaju tekstualni podaci, bez konkretne organizacije, idealno je koristiti tradicionalno čuvanje podataka u tekstualnim datotekama. Na kraju, kada se govori o struktuiranim podacima, njihovo čuvanje zahteva primenu baza podataka. Android operativni sistem podržava *SQLite* sistem za upravljanje relacionim bazama podataka. Sistem je veoma jednostavan za korišćenje i kreira i upravlja datotekama baza podataka koje su privatnog tipa. To znači da baza podataka kreirana za jednu aplikaciju, ne može da se koristi u drugoj aplikaciji.

Dakle, podaci čuvani pomoću baza podataka i *SharedPreferences* objekata ne mogu da se koriste izvan matične aplikacije i neophodno je tražiti druge načine za deljenje podataka između aplikacija. To se postiže provajderima sadržaja koji su tema naredne lekcije.

LITERATURA

Za pripremanje lekcije korišćena je sledeća literatura

1. Lee W. M. 2012. *Android 4 – razvoj aplikacija*, Wiley Publishing, INC
2. http://developer.android.com/training/index.html
3. http://www.tutorialspoint.com/android/
4. http://www.vogella.com/tutorials/android.html
5. http://sqlitebrowser.org/.