

Provajderi sadržaja

Doc. dr Vladimir Milićević



UVOD U ANDROID DELJENJE PODATAKA

U Android operativni sistem je ugrađeno nekoliko veoma korisnih provajdera sadržaja.

Android operativni sistem predlaže **provajdere sadržaja** za deljenje podataka između različitih Android paketa. Provajder sadržaja može biti shvaćen kao izvesno skladište podataka kojem paketi pristupaju primenom odgovarajućeg interfejsa. U mnogim situacijama, provajder sadržaja se ponaša slično bazi podataka. Moguće je postavljati upite, menjati podatke, dodavati nove i uklanjati stare podatke itd. Međutim, za razliku od baze podataka, provajder sadržaja može da koristi različite načine skladištenja podataka. Podaci mogu da budu smešteni u bazu podataka, datoteku ili dostupni preko mreže.

U Android operativni sistem je ugrađeno nekoliko veoma korisnih provajdera sadržaja:

- *Browser* – čuva podatke kao što se zabeležene web stranice, istorija pregleda stranica itd;
- *CallLog* – čuva podatke kao što su propušteni pozivi, detalji o pozivima itd;
- *Contacts* – čuva podatke o kontaktima;
- *MediaStore* – čuva multimedijalne datoteke;
- *Settings* – čuva podešavanja uređaja i preferencije korisnika.

Pored ugrađenih, Android podržava rad sa provajderima sadržaja koje su kreirali programeri tokom razvoja izvesnih Android aplikacija.

UPITI NAD PROVAJDERIMA SADRŽAJA

U Androidu, upit nad provajderom sadržaja koristi formu URI identifikatora.

U Android operativnom sistemu, upit nad provajderom sadržaja koristi formu *URI* (Uniform Resource Identifier) identifikatora sa opcionim specifikatorom koji se odnosi na konkretnu vrstu. Opšti oblik upita nad provajderom sadržaja izgleda ovako:

```
<standardni_prefiks>://<vlasnik>/<putanja_podataka>/<id>
```

Upit je izgrađen iz sledećih komponentata:

- *standardni_prefiks* - za provajdere sadržaja je uvek *content://*.
- *vlasnik* – predstavlja naziv provajdera sadržaja.
- *putanja_podataka* – specificira vrstu traženih podataka. Na primer, ukoliko su u aplikaciji neophodni kontakti iz *Contacts* provajdera sadržaja, putanja može da bude označena kao *people*, a URI identifikator da glasi: *content://contacts/people*.
- *id* – specificira zahtevani zapis. Na primer, ukoliko se zahteva peti kontakt u *Contacts* provajderu sadržaja, *URI* identifikator može da ima sledeći oblik: *content://contacts/people/5*.

Sledećom slikom predstavljeni su često korišćeni stringovi upita nad provajderima sadržaja.

String upita	Opis
<code>content://media/internal/images</code>	Vraća listu svih slika iz interne memorije
<code>content://media/external/images</code>	Vraća listu svih slika iz eksterne memorije
<code>content://call_log/calls</code>	Vraća listu svih poziva koje je registrovao Call Log
<code>content://browser/bookmarks</code>	Vraća listu zapamćenih stranica web čitačem

Slika-1 Primeri stringova upita

KORIŠĆENJE UGRAĐENIH PROVAJDERA SADRŽAJA

U AndroidManifest.xml datoteci neophodno je ugraditi odgovarajuće privilegije.

Za razumevanje koncepta provajdera sadržaja biće uveden odgovarajući primer. U *Eclipse IDE* kreira se projekat pod nazivom *Provider*, a njegova main.xml datoteka ima sledeći oblik.

```
<?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" >

<ListView
    android:id="@+id/android:list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:stackFromBottom="false"
    android:transcriptMode="normal" />

<TextView
    android:id="@+id/contactName"
    android:textStyle="bold"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/contactID"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

Slika-2 main.xml datoteka projekta Provider

AndroidManifest.xml datoteka sa ugrađenom dozvolom pristupa odgovarajućem sadržaju, data je sledećim kodom.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.Provider"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.READ_CONTACTS"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".ProviderActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Slika-3 Davanje privilegija pristupa sadržaju

KORIŠĆENJE UGRAĐENIH PROVAJDERA SADRŽAJA – JAVA KLASA

URI objekat je ugrađen u klasu aktivnosti aplikacije.

JAVA klasom aktivnosti projekta obavljaju se sve aktivnosti vezane za pristup i manipulaciju sadržajem kojeg obezbeđuje određeni provajder sadržaja. URI objekat, koji je odgovoran za izvršavanje stringa upita, ugrađen je u klasu aktivnosti aplikacije. Sledećim kodom je predstavljena klasa aktivnosti projekta *Provider*.

```
package net.learn2develop.Provider;
import android.app.ListActivity;
public class ProviderActivity extends ListActivity {
    /** Poziiva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Uri allContacts = Uri.parse("content://contacts/people");

        Cursor c;
        if (android.os.Build.VERSION.SDK_INT < 11) {
            /*---pre verzije Honeycomb---
            c = managedQuery(allContacts, null, null, null, null);
            /*---Dozvoljava da aktivnost upravlja kursorom
            startManagingCursor(c);
            } else {
            /*---Honeycomb i novije verzije---
            CursorLoader cursorLoader = new CursorLoader(
            this, allContacts, null, null, null, null);
            c = cursorLoader.loadInBackground();
            }

import android.app.ListActivity;
import android.content.CursorLoader;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.widget.CursorAdapter;
import android.widget.SimpleCursorAdapter;
import android.util.Log;

String[] columns = new String[] {
    ContactsContract.Contacts.DISPLAY_NAME,
    ContactsContract.Contacts._ID};

int[] views = new int[] {R.id.contactName, R.id.contactID};

SimpleCursorAdapter adapter;

if (android.os.Build.VERSION.SDK_INT < 11) {
    /*---pre Honeycomb---
    adapter = new SimpleCursorAdapter(
        this, R.layout.main, c, columns, views);
    } else {
    /*---Honeycomb i novije verzije---
    adapter = new SimpleCursorAdapter(
        this, R.layout.main, c, columns, views,
        CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }
    this.setListAdapter(adapter);
}
```

Slika-4 JAVA klasa projekta Provider

KORIŠĆENJE UGRAĐENIH PROVAJDERA SADRŽAJA – FUNKCIONISANJE

Pristup provajderu sadržaja promenio se sa pojavom Android verzije Honeycomb.

Postojanje podataka o kontaktima, u mobilnom telefonu ili emulatoru, uslov je da bi ti podaci mogli da budu prikazani. Iz tog razloga, neophodno je, ukoliko je lista kontakata prazna, dodati nekoliko kontakata u provajder sadržaja *Contacts*. Aplikacija učitava sve kontakte iz aplikacije *Contacts*, a zatim ih prikazuje primenom *ListView* pogleda.

Za pristup aplikaciji *Contacts*, kreiran je URI upit:

```
Uri allContacts = Uri.parse("content://contacts/people");
```

U nastavku, proveravana je verzija Android operativnog sistema, na kojem se aplikacija izvršava. Ukoliko je verzija OS starija od verzije Honeycomb (Android API nivo manji od 11) moguće je koristiti *menageQuery()* metodu za manipulisanje kursorom koji rukuje svim događajima koji se odnose na pauziranje i restartovanje aplikacija.

Novije verzije Android operativnog sistema napuštaju ovu metodu i koriste *CursorLoader* klasu (videti priloženi kod).

```
CursorLoader cursorLoader = new CursorLoader(  
this, allContacts, null, null, null, null);  
c = cursorLoader.loadInBackground();
```

Slika-5 CursorLoader klasa

Ova klasa izvršava upit uz korišćenje kursora u pozadinskoj niti i na taj način ne blokira korisnički interfejs aplikacije. Objekat klase *SimpleCursorAdapter* povezuje *TextView* (ili *ImageView*) poglede definisane u *main.xml* datoteci. Takođe, kodom je prikazan prevaziđeni konstruktor ovog objekat i novi (sledeća slika) koji se koristi u svim novim verzijama Androida (API nivo 11 i veći). Novi konstruktor koristi *Fleg* za registrovanje adaptera i na taj način dobija informacije o promenama na strani provajdera sadržaja.

Takođe, aplikacija zahteva *READ_CONTACTS* privilegiju, u *AndroidManifest.xml* datoteci, da bi mogla da pristupi sadržaju provajdera *Contacts*.

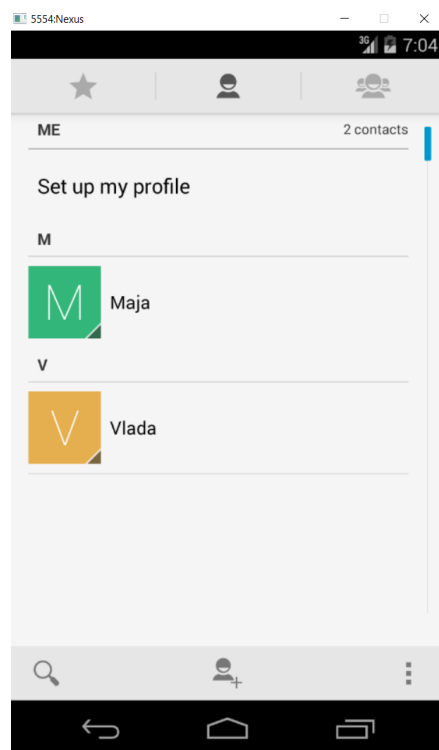
```
//---Honeycomb i novije verzije---  
adapter = new SimpleCursorAdapter(  
this, R.layout.main, c, columns, views,  
CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
```

Slika-6 Novi SimpleAdapter konstruktor

KORIŠĆENJE UGRAĐENIH PROVAJDERA SADRŽAJA – DEMONSTRACIJA

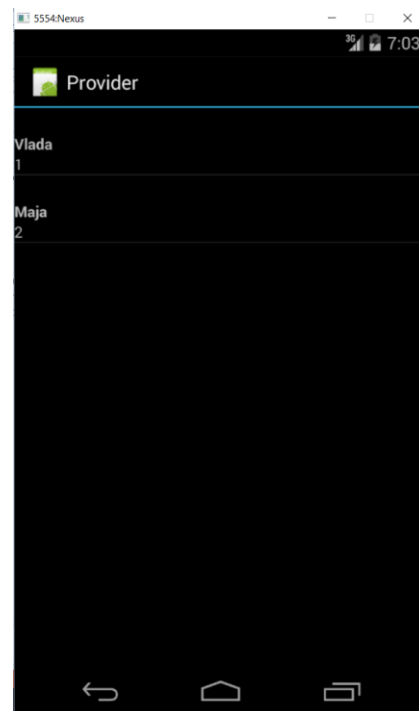
Pokretanjem aplikacije emulatorom prikazuje se lista svih kontakata iz provajdera Contacts.

Klikom na F11, aplikacija se prevodi i pokreće emulatorom. Ukoliko je lista kontakata prazna, neophodno je uneti nekoliko kontakata primenom aplikacije *Contacts* (sledeća slika).



Slika-7 Dodavanje kontakata

Pokretanjem aplikacije emulatorom, ili Android telefonom, prikazuje se lista svih kontakata iz provajdera Contacts (sledeća slika).



Slika-8 Prikazivanje liste kontakata

UGRAĐENE KONSTANTE STRINGA UPITA

Pored URI identifikatora upita, moguće je koristiti i listu ugrađenih konstanti stringa upita za specificiranje URI identifikatora.

Pored URI identifikatora upita, moguće je koristiti i listu ugrađenih konstanti stringa upita za specificiranje URI identifikatora, za različite tipove podataka, u Android aplikacijama. Na primer, sledeće dve naredbe su ekvivalentne:

```
Uri allContacts = Uri.parse("content://contacts/people");
```

```
Uri allContacts = ContactsContract.Contacts.CONTENT_URI;
```

Slede primeri najčešće korišćenih konstanti za obraćanje provajderima sadržaja:

- `Browser.BOOKMARKS_URI`;
- `Browser.SEARCHES_URI`;
- `CallLog.CONTENT_URI`;
- `MediaStore.Images.Media.INTERNAL_CONTENT_URI`;
- `MediaStore.Images.Media.EXTERNAL_CONTENT_URI`;
- `Settings_CONTENT_URI`.

Za očitavanje prvog kontakta, identifikacioni broj se specificira na sledeći način:

```
Uri allContacts = Uri.parse("content://contacts/people/1");
```

Kao alternativu, moguće je koristiti predefinisanu konstantu sa metodom `withAppendedID()` klase `ContentUris`:

```
Uri allContacts = ContentUris.
```

```
withAppendedID(ContactsContract.Contacts.CONTENT_URI,1)
```

Podatke je moguće, umesto ListView pogledom, prikazivati i kursorom (sledeća slika).

```
package net.learn2develop.Provider;
import android.app.ListActivity; //ovde je uključeno i android.util.Log
public class ProviderActivity extends ListActivity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Uri allContacts = ContactsContract.Contacts.CONTENT_URI;

        Cursor c;
        *****
    }
    this.setAdapter(adapter);
    PrintContacts(c);
}

private void PrintContacts(Cursor c)
{
    if (c.moveToFirst()) {
        do{
            String contactID = c.getString(c.getColumnIndex(
                ContactsContract.Contacts._ID));
            String contactDisplayName =
                c.getString(c.getColumnIndex(
                    ContactsContract.Contacts.DISPLAY_NAME));
            Log.v("Content Providers", contactID + ", " +
                contactDisplayName);
        } while (c.moveToNext());
    }
}
```

Slika-9 Korišćenje kursora za prikaz

UGRAĐENE KONSTANTE STRINGA UPITA – PRISTUP DODATNIM INFORMACIJAMA

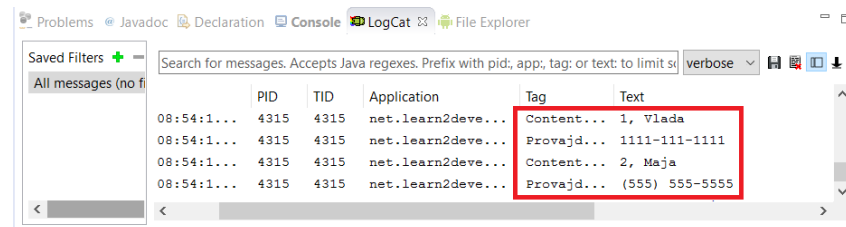
Za preuzimanje dopunskih informacija neophodno je ponovo izvršiti upit nad provajderom sadržaja.

Prethodnim primerom, preuzete su informacije koje se odnose na identifikacioni broj i naziv svakog kontakta iz aplikacije *Contacts*. Ukoliko se želi preuzimanje još neke informacije, na primer telefonskog broja, neophodno je još jednom izvršiti upit nad provajderom sadržaja (sledeća slika).

```
private void PrintContacts(Cursor c) {
    if (c.moveToFirst()) {
        do {
            String contactID = c.getString(c.getColumnIndex(
                ContactsContract.Contacts._ID));
            String contactDisplayName =
                c.getString(c.getColumnIndex(
                    ContactsContract.Contacts.DISPLAY_NAME));
            Log.v("Content Providers", contactID + ", " +
                contactDisplayName);
            //učitavanje broja telefona
            int hasPhone=
                c.getInt(c.getColumnIndex(
                    ContactsContract.Contacts.HAS_PHONE_NUMBER));
            if (hasPhone==1) {
                Cursor phoneCursor =
                    getContentResolver().query(
                        ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                        ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " +
                            contactID, null, null);
                while (phoneCursor.moveToNext()) {
                    Log.v("Provajderi sadržaja ", phoneCursor.getString(
                        phoneCursor.getColumnIndex(
                            ContactsContract.CommonDataKinds.Phone.NUMBER));
                }
                phoneCursor.close();
            }
        } while (c.moveToNext());
    }
}
```

Slika-10 Učitavanje broja telefona

Prethodni kod, sadržan u proširenoj metodi *PrintContacts()*, prvo proverava da li kontakt sadrži telefonski broj primenom polja *ContactsContract.Contacts.HAS_PHONE_NUMBER*. Ukoliko kontakt sadrži bar jedan telefonski broj, upit nad provajderom sadržaja *Contacts* biće ponovo izvršen i preuzeti brojevi mogu da se pročitaju u *LogCat* prozoru *Eclipse IDE* razvojnog okruženja.



Slika-11 CatLog prozor

PROJEKCIJE

Projekcija je parametar kojim se određuje koliko kolona se vraća prilikom izvršavanja upita.

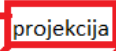
U oba načina, prevaziđena primena metode `manageQuery()` i aktuelna primena klase `CursorLoader`, koriste parametre kojim se određuje koliko kolona se vraća prilikom izvršavanja upita. Ovaj parametar se naziva projekcija. U Aktuelnom primeru, njegova vrednost iznosi `null` i to je prikazano sledećom slikom.

```
Cursor c;
if (android.os.Build.VERSION.SDK_INT < 11) {
    //---pre verzije Honeycomb---
    c = managedQuery(allContacts, null, null, null, null);
    //---Dozvoljava da aktivnost upravlja kursorom
    startManagingCursor(c);
} else {
    //---Honeycomb i novije verzije---
    CursorLoader cursorLoader = new CursorLoader(
        this, allContacts, null, null, null, null);
    c = cursorLoader.loadInBackground();
}
```

Slika-12 Projekcija

Ovaj način, manipulacije podacima, omogućava da se tačno specificira broj kolona koje se vraćaju kada se kreiraju polja koja sadrže nazive kolona. Navedeno je prikazano sledećim kodom.

```
Cursor c;
String[] columns = new String[] {
    ContactsContract.Contacts.DISPLAY_NAME,
    ContactsContract.Contacts.ID,
    ContactsContract.Contacts.HAS_PHONE_NUMBER,
};
if (android.os.Build.VERSION.SDK_INT < 11) {
    //---pre verzije Honeycomb---
    c = managedQuery(allContacts, columns, null, null, null);
    //---Dozvoljava da aktivnost upravlja kursorom
    startManagingCursor(c);
} else {
    //---Honeycomb i novije verzije---
    CursorLoader cursorLoader = new CursorLoader(
        this, allContacts, columns, null, null, null);
    c = cursorLoader.loadInBackground();
}
```



Slika-12 Projekcija vraća tri kolone

FILTRIRANJE I SORTIRANJE

Filtriranje i sortiranje su omogućeni kroz izvršavanje SQL klauzula WHERE i ORDER BY.

Filtriranje je omogućeno kroz izvršavanje SQL klauzule *WHERE*, a to je određeno trećim i četvrtim parametrom prevaziđene metode *manageQuery* i četvrtim i petim parametrom aktuelnog pristupa koji podrazumeva korišćenje klase *CursorLoader*. Na primer, sledeća naredba omogućava učitavanje samo onih kontakata koji počinu slovom *v*.

```
Cursor c;
String[] columns = new String[] {
    ContactsContract.Contacts.DISPLAY_NAME,
    ContactsContract.Contacts._ID,
    ContactsContract.Contacts.HAS_PHONE_NUMBER,
};
if (android.os.Build.VERSION.SDK_INT < 11) {
    //---pre verzije Honeycomb---
    c = managedQuery(allContacts, columns,
        ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
        new String[] {"V%"}, null);
    //---Dozvoljava da aktivnost upravlja kursorom
    startManagingCursor(c);
} else {
    //---Honeycomb i novije verzije---
    CursorLoader cursorLoader = new CursorLoader(
        this, allContacts, columns, ContactsContract.Contacts.DISPLAY_NAME +
        " LIKE ?",
        new String[] {"V%"}, null);
    c = cursorLoader.loadInBackground();
}
```

Slika-14 Filtriranje sadržaja

Poslednji parametar, oba pristupa, omogućava specificiranje SQL klauzule *ORDER BY* kojom se realizuje sortiranje rezultata izvršavanja upita. Primena sortiranja je prikazana kodom sa sledeće slike.

```
Cursor c;
String[] columns = new String[] {
    ContactsContract.Contacts.DISPLAY_NAME,
    ContactsContract.Contacts._ID,
    ContactsContract.Contacts.HAS_PHONE_NUMBER,
};
if (android.os.Build.VERSION.SDK_INT < 11) {
    //---pre verzije Honeycomb---
    c = managedQuery(allContacts, columns,
        ContactsContract.Contacts.DISPLAY_NAME + " LIKE ?",
        new String[] {"V%"},
        ContactsContract.Contacts.DISPLAY_NAME + " ASC");
    //---Dozvoljava da aktivnost upravlja kursorom
    startManagingCursor(c);
} else {
    //---Honeycomb i novije verzije---
    CursorLoader cursorLoader = new CursorLoader(
        this, allContacts, columns, ContactsContract.Contacts.DISPLAY_NAME +
        " LIKE ?",
        new String[] {"V%"},
        ContactsContract.Contacts.DISPLAY_NAME + " ASC");
    c = cursorLoader.loadInBackground();
}
```

Slika-15 Sortiranje sadržaja

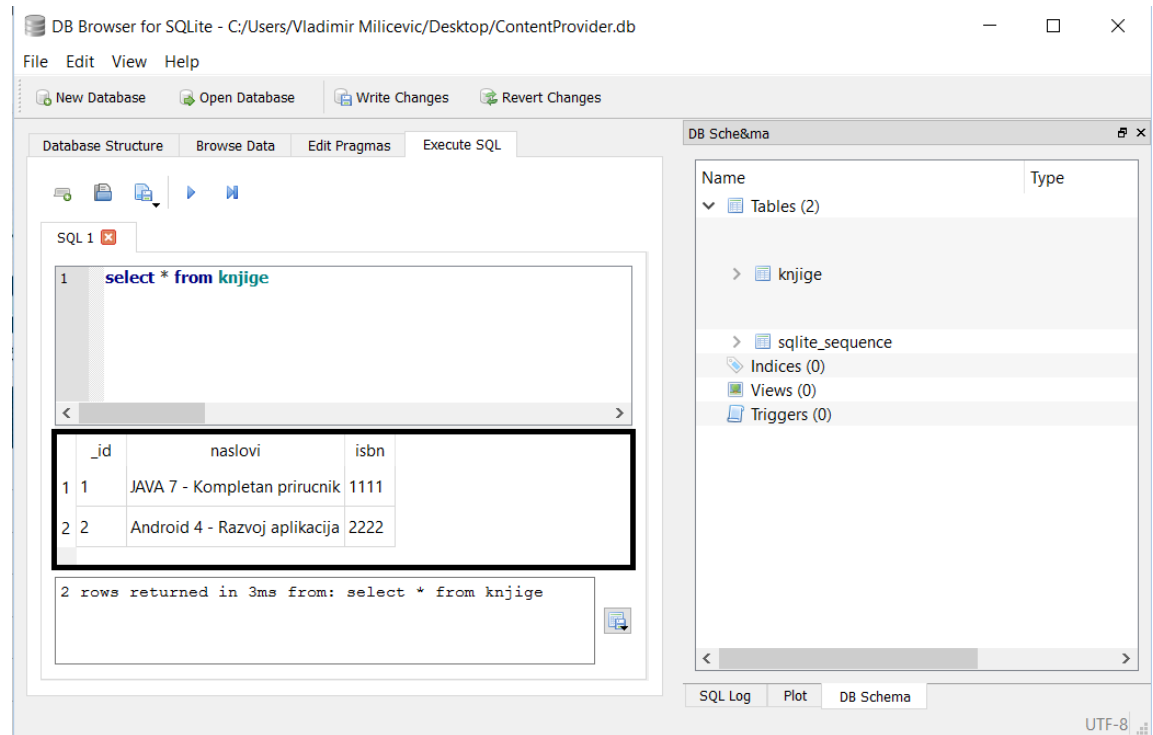
KLASA VLASTITOG PROVAJDERA SADRŽAJA

Klasa provajdera sadržaja je naslednica klase `ContentProvider`.

Kreiranje vlastitog provajdera sadržaja je, u osnovi, veoma jednostavno. Neophodno je implementirati klasu, koja nasleđuje iz apstraktne klase `ContentProvider` i potom definisati različite metode te klase. Kao primer, biće kreiran provajder sadržaja koji skladišti knjige u tabeli baze podataka. Tabela sadrži tri polja:

- `_id`;
- `naslov`;
- `isbn`.

Sledećom slikom prikazana je tabela baze podataka u kojoj će kreirani provajder sadržaja skladištiti knjige.



Slika-1 Tabela baze podataka

KREIRANJE KLASA PROVAJDERA SADRŽAJA

Prvi zadatak je kreiranje prazne klase provajdera sadržaja.

U prvom koraku biće kreirana klasa provajdera sadržaja sa pomoćnom klasom *DataBaseHelper* i mehanizmima za kreiranje baze podataka. Inicijalni kod klase prikazan je sledećom slikom.

```
package net.learn2develop.ContentProviders;
import android.content.ContentProvider;
public class BooksProvider extends ContentProvider{
    static final String PROVIDER_NAME =
        "net.learn2develop.provider.Books";

    static final Uri CONTENT_URI =
        Uri.parse("content://" + PROVIDER_NAME + "/books");

    static final String _ID = "_id";
    static final String TITLE = "naslovi";
    static final String ISBN = "isbn";

    static final int BOOKS = 1;
    static final int BOOK_ID = 2;

    private static final UriMatcher uriMatcher;
    static{
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(PROVIDER_NAME, "books", BOOKS);
        uriMatcher.addURI(PROVIDER_NAME, "books/#", BOOK_ID);
    }
}
```

```
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;
import android.util.Log;
```

```
///---kreiranje baze podataka---
SQLiteDatabase booksDB;
static final String DATABASE_NAME = "Books";
static final String DATABASE_TABLE = "knjige";
static final int DATABASE_VERSION = 1;
static final String DATABASE_CREATE =
    "create table " + DATABASE_TABLE +
    " (_id integer primary key autoincrement, "
    + "naslovi text not null, isbn text not null);";
private static class DatabaseHelper extends SQLiteOpenHelper{
    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db){
        db.execSQL(DATABASE_CREATE);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
        int newVersion) {
        Log.w("Baza podataka provajera sadržaja",
            "Ažuriranje verzije sa " +
            oldVersion + " na verziju " + newVersion +
            ", stari podaci biće uništeni");
        db.execSQL("DROP TABLE IF EXISTS titles");
        onCreate(db);
    }
}
```

Slika-2 Klasa provajdera sadržaja

KLASA PROVAJDERA SADRŽAJA – METODE GETTYPE(), ONCREATE(), QUERY()

Klasu provajdera sadržaja je neophodno proširiti metodama za CRUD podršku.

Pored podrške za kreiranje baze podataka, klasu provajdera sadržaja je neophodno snabdeti metodama koje obezbeđuju izvođenje operacija nad bazom podataka. Prvo će biti implementirane metode *getType()*, *onCreate()* i *query()*.

```
@Override
public String getType(Uri uri) {
    switch (uriMatcher.match(uri)){
        //---get all books---
        case BOOKS:
            return "vnd.android.cursor.dir/vnd.learn2develop.books ";

        //---get a particular book---
        case BOOK_ID:
            return "vnd.android.cursor.item/vnd.learn2develop.books ";

        default:
            throw new IllegalArgumentException("Unsupported URI: " + uri);
    }
}

@Override
public boolean onCreate() {
    Context context = getContext();
    DatabaseHelper dbHelper = new DatabaseHelper(context);
    booksDB = dbHelper.getWritableDatabase();
    return (booksDB == null)? false:true;
}

@Override
public Cursor query(Uri uri, String[] projection, String selection,
    String[] selectionArgs, String sortOrder) {
    SQLiteDatabase sqlBuilder = new SQLiteDatabase();
    sqlBuilder.setTables(DATABASE_TABLE);

    if (uriMatcher.match(uri) == BOOK_ID)
        //---ako se pruzima konkretna knjiga---
        sqlBuilder.appendWhere(
            "_ID + " = " + uri.getPathSegments().get(1));

    if (sortOrder==null || sortOrder=="")
        sortOrder = TITLE;

    Cursor c = sqlBuilder.query(
        booksDB,
        projection,
        selection,
        selectionArgs,
        null,
        null,
        sortOrder);
    //---registrovanje praćenja promena URI identifikatora---
    c.setNotificationUri(getContext().getContentResolver(), uri);
    return c;
}
```

Slika-3 Metode *getType()*, *onCreate()*, *query()*

KLASA PROVAJDERA SADRŽAJA – METODE *INSERT()*, *DELETE()* I *UPDATE()*.

Definicija klase proširuje se sa još tri metode.

Pored priloženih metoda *getType()*, *onCreate()* i *query()*, neophodno je dodati i metode *insert()*, *delete()* i *update()*.

```
@Override
public int delete(Uri arg0, String arg1, String[] arg2) {
    int count=0;
    switch (uriMatcher.match(arg0)){
        case BOOKS:
            count = booksDB.delete(
                DATABASE_TABLE,
                arg1,
                arg2);
            break;
        case BOOK_ID:
            String id = arg0.getPathSegments().get(1);
            count = booksDB.delete(
                DATABASE_TABLE,
                _ID + " = " + id +
                (!TextUtils.isEmpty(arg1) ? " AND (" +
                    arg1 + ')' : ""),
                arg2);
            break;
        default: throw new IllegalArgumentException("Unknown URI " + arg0);
    }
    getContext().getContentResolver().notifyChange(arg0, null);
    return count;
}

@Override
public Uri insert(Uri uri, ContentValues values) {
    ///---dodaje novu knjigu---
    long rowID = booksDB.insert(DATABASE_TABLE,
        "",values);
    ///---ako je dodavanje uspešno---
    if (rowID>0){
        Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
        getContext().getContentResolver().notifyChange(_uri, null);
        return _uri;
    }
    throw new SQLException("Dodavanje u vrstu nije uspeo " + uri);
}

@Override
public int update(Uri uri, ContentValues values, String selection,
    String[] selectionArgs) {
    int count = 0;
    switch (uriMatcher.match(uri)){
        case BOOKS:
            count = booksDB.update(
                DATABASE_TABLE,
                values,
                selection,
                selectionArgs);
            break;
        case BOOK_ID:
            count = booksDB.update(
                DATABASE_TABLE,
                values,
                _ID + " = " + uri.getPathSegments().get(1) +
                (!TextUtils.isEmpty(selection) ? " AND (" +
                    selection + ')' : ""),
                selectionArgs);
            break;
        default: throw new IllegalArgumentException("Nepoznat URI " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return count;
}
```

Slika-4 Metode insert(), delete(), update()

PROVAJDER SADRŽAJA – ANDROIDMANIFEST.XML DATOTEKA

Provajdera sadržaja je neophodno uključiti AndroidManifest.xml datotekom.

Da bi rad sa provajderom sadržaja bio moguć, pored kreiranja klase provajdera sadržaja, neophodno je napraviti i izvesne modifikacije u datoteci *AndroidManifest.xml*. Koristeći XML tag `<provider> ... </provider>`, ovom XML datotekom se uključuje provajder sadržaja. Kod datoteke je dat sledećom slikom.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.ContentProviders"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".ContentProvidersActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <provider android:name="BooksProvider"
            android:authorities="net.learn2develop.provider.Books">
        </provider>
    </application>
</manifest>
```

Slika-5 AndroidManifest.xml i provajder

NAČIN FUNKCIONISANJA KLASE PROVAJDERA SADRŽAJA

Funkcionisanje klase provajdera sadržaja bazira se na funkcionalnostima predefinisanih metoda osnovne klase `ContentProvider`.

Nakon prvog koraka, kreiranja klase naslednice osnovne klase `ContentProvider`, predefinisano je nekoliko metoda bazne klase:

- `getType()` – vraća MIME tip sa odgovarajućim URI;
- `onCreate()` – izvršava se sa pokretanjem provajdera;
- `query()` – učitava zahtev klijenta i vraća `Cursor` objekat;
- `insert()` – unosi novi zapis u provajder sadržaja;
- `delete()` – briše zapis pomoću provajdera sadržaja;
- `update()` – koristi provajder sadržaja za ažuriranje zapisa.

U inicijalnom kodu klase moguće je primetiti da je korišćen objekat `UriMatcher` (videti kod) za analiziranje sadržaja URI identifikatora koji je prosleđen provajderu objektom `ContentResolver`. Na primer, sledećim URI identifikatorima su zatraženi zahtevi za učitavanjem svih knjiga i knjige čiji identifikator ima vrednost 1, respektivno.

```
Uri.parse("content://" + PROVIDER_NAME + "/books");
```

```
Uri.parse("content://" + PROVIDER_NAME + "/books/1");
```

Provajder koristi `SQLite` bazu podataka pa je iskorišćena `SQLiteHelper` pomoćna klasa za lakše upravljanje bazom podataka (videti kod).

Predefinisanjem metode `getType()` tako što mu se predaju `URI` objekat, dobijen je unikatan način za opisivanje tipa podataka za provajder sadržaja. Primenom `UriMatcher` objekta više knjiga se učitava pomoću: `vnd.android.cursor.dir/vnd.learn2develop.books`, a pojedinačne knjige pomoću: `vnd.android.cursor.item/vnd.learn2develop.books` (videti kod metode `getType`).

U sledećem koraku predefinisana je `onCreate()` metoda sa ciljem omogućavanja konekcije sa bazom podataka nakon pokretanja provajdera sadržaja (videti priloženi kod metode).

Takođe, predefinisana je i `query()` metoda kojom je omogućeno klijentima da postavljaju upite za knjige. Metoda je podešena tako da se rezultat upita vraća kao tip `Cursor`, sortiran po polju `TITLE`.

NAČIN FUNKCIONISANJA KLASE PROVAJDERA SADRŽAJA - NASTAVAK

Unošenje nove knjige, brisanje iz baze i ažuriranje omogućeno je predefinisanjem metoda `insert()`, `delete()` i `update()`.

Da bi nov podatak bio unešen u bazu podataka, primenom provajdera sadržaja, neophodno je koristiti predefinisano metodu `insert()` koja preuzima kao argumente dva objekta `Uri` i `ContentValues`. Kao rezultat, metoda vraća tip podataka `Uri`. Nakon obavljenog dodavanja, novog zapisa u bazu, izvršava se metoda `notifyChange()` objekta klase `ContentResolver` (videti priloženi kod).

Za uklanjanje zapisa, u konkretnom slučaju knjige, iz baze podataka, a primenom provajdera sadržaja, predefinisana je i upotrebljena metoda `delete()`. Takođe, metoda `delete()`, omogućava da se izvrši metoda `notifyChange()` objekta klase `ContentResolver` (videti priloženi kod) nakon izvršenog uklanjanja podataka. Na ovaj način se obaveštavaju registrovani posmatrači da je obrisana odgovarajuća vrsta.

U nastavku je predefinisana i iskorišćena metoda `update()` koja, slično kao i prethodne dve metode, izvršava metodu `notifyChange()` objekta klase `ContentResolver` (videti priloženi kod). Kroz ovu akciju obaveštavaju se registrovani posmatrači da je ažurirana odgovarajuća vrsta.

Na samom kraju, da bi provajder sadržaja bio registrovan i angažovan u Android sistemu, modifikovana je datoteka `AndroidManifest.xml` dodavanjem XML elementa `<provider>`.

KLASA AKTIVNOSTI I UPOTREBA PROVAJDERA SADRŽAJA.

Nakon definisanja provajdera sadržaja neophodno je omogućiti mehanizme implementacije u Android aplikaciji.

Za primenu kreiranog provajdera sadržaja, u Android aplikaciji, neophodno je kreirati klasu aktivnosti i korisnički interfejs aplikacije preko kojeg će korisnik i aplikacija komunicirati. Sledećom slikom dat je kod odgovarajuće klase aktivnosti koja omogućava angažovanje provajdera sadržaja.

```
package net.learn2develop.ContentProviders;
import android.app.Activity;
public class ContentProvidersActivity extends Activity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onClickAddTitle(View view) {
        /*
        ///---add a book---
        ContentValues values = new ContentValues();
        values.put(BooksProvider.TITLE, ((EditText)
            findViewById(R.id.txtTitle)).getText().toString());
        values.put(BooksProvider.ISBN, ((EditText)
            findViewById(R.id.txtISBN)).getText().toString());
        Uri uri = getContentResolver().insert(
            BooksProvider.CONTENT_URI, values);
        */
        ContentValues values = new ContentValues();
        values.put("naslovi", ((EditText)
            findViewById(R.id.txtTitle)).getText().toString());
        values.put("isbn", ((EditText)
            findViewById(R.id.txtISBN)).getText().toString());
        Uri uri = getContentResolver().insert(
            Uri.parse("content://net.learn2develop.provider.Books/books"),
            values);

        Toast.makeText(getBaseContext(), uri.toString(),
            Toast.LENGTH_LONG).show();
    }

    public void onClickRetrieveTitles(View view) {
        ///---vraća naslov---
        Uri allTitles = Uri.parse("content://net.learn2develop.provider.Books/books");
        Cursor c;

        CursorLoader cursorLoader = new CursorLoader(this, allTitles, null, null,
            null, "naslovi desc");
        c = cursorLoader.loadInBackground();
        if (c.moveToFirst()) {
            do {
                Toast.makeText(this,
                    c.getString(c.getColumnIndex(
                        BooksProvider._ID)) + ", " +
                    c.getString(c.getColumnIndex(
                        BooksProvider.TITLE)) + ", " +
                    c.getString(c.getColumnIndex(
                        BooksProvider.ISBN)),
                    Toast.LENGTH_SHORT).show();
            } while (c.moveToNext());
        }

        public void updateTitle() {
            ContentValues editedValues = new ContentValues();
            editedValues.put(BooksProvider.TITLE, "Android Tipovi i trikovi.");
            getContentResolver().update(
                Uri.parse("content://net.learn2develop.provider.Books/books/2"),
                editedValues, null, null);
        }

        public void deleteTitle() {
            ///---brisanje naslova---
            getContentResolver().delete(
                Uri.parse("content://net.learn2develop.provider.Books/books/2"),
                null, null);
            ///---brisanje svih naslova---
            getContentResolver().delete(
                Uri.parse("content://net.learn2develop.provider.Books/books"),
                null, null);
        }
    }
}
```

Slika-6 Klasa aktivnosti za korišćenje provajdera

XML DATOTEKA KORISNIČKOG INTERFEJSA ZA IMPLEMENTACIJU PROVAJDERA SADRŽAJA.

Aplikacija za primenu provajdera sadržaja kompletirana je definicijom korisničkog interfejsa.

U folderu projekta, podfolder *res/layout*, bira se XML datoteka *main.xml* i menjaju se njena osnovna podešavanja dodavanjem sledećeg koda koji odgovara elementima korisničkog interfejsa i njihovom rasporedu na ekranu. Novom definicijom ove datoteke, kompletirana je aplikacija za demonstraciju primene vlastitog provajdera sadržaja.

Sledećim kodom data je *main.xml* datoteka.

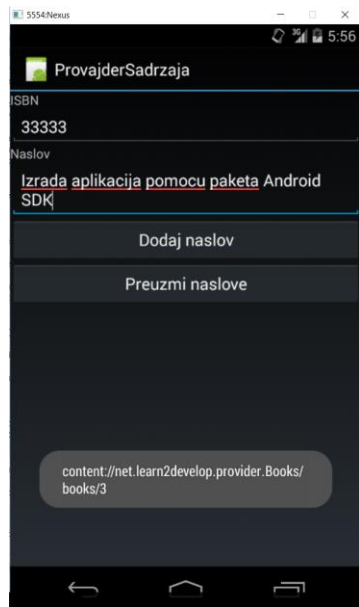
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="ISBN" />
    <EditText
        android:id="@+id/txtISBN"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Naslov" />
    <EditText
        android:id="@+id/txtTitle"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <Button
        android:text="Dodaj naslov"
        android:id="@+id/btnAdd"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickAddTitle" />
    <Button
        android:text="Preuzmi naslove"
        android:id="@+id/btnRetrieve"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:onClick="onClickRetrieveTitles" />
</LinearLayout>
```

Slika-7 *main.xml* datoteka projekta

FUNKCIONISANJE APLIKACIJE ZA ANGAŽOVANJE PROVAJDERA SADRŽAJA

Funkcionisanje programa počinje izvršavanjem metode onCreate() i učitavanjem UI za upravljanje provajderom sadržaja.

Klikom na F11, program je preveden i pokrenut emulatorom. Pokreće se aktivnost koja je modifikovana tako da korisniku omogućava da unese ISBN broj i naslov knjige pomoću prethodno kreiranog provajdera sadržaja.



Slika-8 Unos nove knjige

Da bi provajder sadržaja dodao novu knjigu, neophodno je kreirati, u metodi *onClickAddTitle()*, objekat tipa *ContentValues* u koji se pakuju informacije koje se odnose na konkretnu knjigu (videti priloženi programski kod).

Takođe, kada se pogleda priloženi kod klase aktivnosti, za navedenu metodu moguće je uočiti da se jedan deo koda pojavljuje obeležen oznakom za komentare, a ispod njega je aktivan kod. Oba koda imaju zadatak da ukažu na polja *ISBN* i *naslovi* sa razlikom da prvi blok koda koristi konstante *BooksProvider.ISBN* i *BooksProvider.TITLE*, respektivno, za pristupanje poljima, ukoliko je provajder sadržaja kreiran u istom paketu kao aplikacija, dok drugi blok koda omogućava pristupanje provajderu sadržaja, koji ne mora da se nalazi u paketu aplikacije, direktnim navođenjem naziva polja i kompletnog URI identifikatora sadržaja. Razlike su prikazane sledećom slikom.

```
/* provajder je u istom paketu
/--add a book--
ContentValues values = new ContentValues();
values.put(BooksProvider.TITLE, ((EditText)
    findViewById(R.id.txtTitle)).getText().toString());
values.put(BooksProvider.ISBN, ((EditText)
    findViewById(R.id.txtISBN)).getText().toString());
Uri uri = getContentResolver().insert(
    BooksProvider.CONTENT_URI, values); */

//provajder ne mora da bude u istom paketu
ContentValues values = new ContentValues();
values.put("naslovi", ((EditText)
    findViewById(R.id.txtTitle)).getText().toString());
values.put("isbn", ((EditText)
    findViewById(R.id.txtISBN)).getText().toString());
Uri uri = getContentResolver().insert(
    Uri.parse("content://net.learn2develop.provider.Books/books"),
    values);
```

Slika-9 Provajderi i paketi aplikacija

FUNKCIONISANJE APLIKACIJE ZA ANGAŽOVANJE PROVAJDERA SADRŽAJA - NASTAVAK

Metode za ažuriranje i brisanje oslanjaju se na navođenje URI identifikatora sadržaja kojim se ukazuje na identifikator konkretne knjige.

Angažovanjem metode `onClickRetrieveTitles()`, omogućeno je učitavanje knjiga koja su prethodno definisane provajderom sadržaja. Takođe, u metodu je ugrađen upit koji kao rezultat vraća niz knjiga sortiran po polju `title` u opadajućem poretku (obeleženo na sledećoj slici).

```
public void onClickRetrieveTitles(View view) {
    //---vraća naslov---
    Uri allTitles = Uri.parse("content://net.learn2develop.provider.Books/books");
    Cursor c;
    CursorLoader cursorLoader = new CursorLoader(this,allTitles, null, null,
        null,"naslovi desc");
    c = cursorLoader.loadInBackground();
    if (c.moveToFirst()) {
        do{
            Toast.makeText(this,
                c.getString(c.getColumnIndex(
                    BooksProvider._ID)) + ", " +
                c.getString(c.getColumnIndex(
                    BooksProvider.TITLE)) + ", " +
                c.getString(c.getColumnIndex(
                    BooksProvider.ISBN)),
                    Toast.LENGTH_SHORT).show();
        } while (c.moveToNext());
    }
}
```

Slika-10 Učitavanje knjiga dodatih provajderom

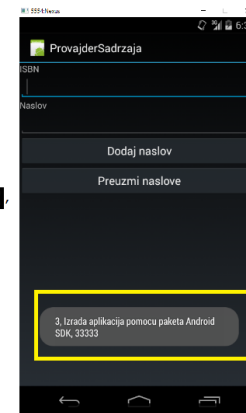
Preuzete kniige prikazuju se na ekranu uređaja (sledeća slika).

Takođe, aplikacija omogućava da se ažuririraju detalji, koji se odnose na konkretnu knjigu, primenom metode `update()` klase aktivnosti aplikacije. Navođenjem URI identifikatora sadržaja ukazuje se na identifikator konkretne knjige (videti sledeću sliku – obeleženo crvenom bojom).

Takođe, primenom URI identifikatora sadržaja, metodom `delete()`, omogućeno je brisanje pojedinačnih ili svih knjiga (videti sledeću sliku – obeleženo crnom bojom).

```
public void updateTitle() {
    ContentValues editedValues = new ContentValues();
    editedValues.put(BooksProvider.TITLE, "Android Tipovi i trikovi.");
    getContentResolver().update(
        Uri.parse("content://net.learn2develop.provider.Books/books/2"),
        editedValues,null,null);
}

public void deleteTitle() {
    //---brisanje naslova---
    getContentResolver().delete(
        Uri.parse("content://net.learn2develop.provider.Books/books/2"),
        null, null);
    //---brisanje svih naslova---
    getContentResolver().delete(
        Uri.parse("content://net.learn2develop.provider.Books/books"),
        null, null);
}
```



Slika-11 Brisanje, ažuriranje i prikaz knjiga