Dobavljači sadržaja i SQLite

Mobilne aplikacije

Stevan Gostojić

Fakultet tehničkih nauka, Novi Sad

5. novembar 2024.

Pregled sadržaja

Dobavljači sadržaja

- Dobavljači sadržaja (`ContentProvider`) enkapsuliraju podatke omogućavajući da im se pristupi na standardizovan način
- Omogućava razmenu podataka između komponenti koje se nalaze u različitim procesima i obezbeđuju bezbedan (a obično i perzistentan) pristup podacima.

Dobavljači sadržaja

- Pružaju podatke drugim komponentama u formi jedne ili više tabela (slično relacionalnom modelu podataka)
- Vrsta u tabeli predstavlja instancu entiteta, a kolona njegovo svojstvo

Dobavljači sadržaja

Table 1: Primer dobavljenih podataka.

date	number	duration	type	_ID
2024-03-01 13:57:20	0641234567	62	1	1
2024-03-15 9:40:15	0635678901	29	2	2
2024-03-19 21:10:44	0612345678	15	2	3
2024-03-25 15:35:46	0609876543	110	1	4
2024-03-27 12:31:05	0623456789	95	2	5

type 1 = INCOMING_TYPE, type 2 = OUTGOING_TYPE

Dobavljači sadržaja

- Podatke opisuje URI i MIME tip
- URI (`content://user_dictionary/words`) se sastoji iz šeme (`content`), imena dobavljača sadržaja (`user_dictionary`) i imena tabele (`words`)
- Pojedinačnoj vrsti se može pristupiti dodavanjem njenog identifikatora na ovaj URI (`content://user_dictionary/words/1`)
- MIME tip specificira tip sadržaja (`text/plain`, `text/pdf`, `image/jpeg`, itd.)

Dobavljači sadržaja

- Za pristup podacima koje pruža dobavljač sadržaja koristi se ContentResolver klasa
- Ova klasa omogućava izvršavanje CRUD operacija nad (perzistentnim) skladištem podataka
- Pri tome se dobavljač sadržaja ne mora nalaziti u istom procesu, a automatski je omogućen bezbedan pristup podacima

Dobavljači sadržaja

Podacima se pristupa u dva koraka:

- Zatraže se odgovarajuća prava pristupa
- Izvrši se upit nad dobavljačem sadržaja

Dobavljači sadržaja

- Da bi mogla da pristupi podacima, komponenta mora da ima "read access permission" nad odgovarajućim provajderom
- Ovo pravo pristupa se mora specificirati u `AndroidManifest.xml` (nije ga moguće specificirati u toku izvršavanja aplikacije)

Prava pristupa

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest ... >
3   <uses-permission android:name="android.permission.READ_CALL_LOG">
4   <uses-permission android:name="android.permission.WRITE_CALL_LOG">
5 </manifest>
6
```

Dobavljači sadržaja

- Upit se postavlja na sličan način na koji se postavlja SQL upit
- Sadrži URI, spisak kolona koje treba vratiti (projekciju), uslov koji vraćene vrste treba da zadovolje (selekciju) i način sortiranja rezultata

Upit

```
1  import android.database.Cursor;
2  import android.provider.CallLog;
3  ...
4
5    String[] projection = {
6            CallLog.Calls.NUMBER,
7            CallLog.Calls.DURATION,
8            CallLog.Calls.DATE
9    };
10   String selectionClause = null;
11   String[] selectionArgs = null;
12   String sortOrder = null;
13
14   Cursor cursor = getContentResolver().query(
15           CallLog.Calls.CONTENT_URI,
16           projection,
17           selectionClause,
18           selectionArgs,
19           sortOrder);
20
21   while(cursor.moveToNext()) {
22       String number = cursor.getString(0);
23       long duration = cursor.getLong(1);
24       Date date = new Date(cursor.getLong(2));
25   }
26
```

Dobavljači sadržaja

- Na sličan način na koji je moguće pristupiti podacima, moguće ih je i promeniti.

Dobavljači sadržaja

```
1  // Demonstrates the usage of insert method
2  Uri newUri;
3
4  ContentValues newValues = new ContentValues();
5  newValues.put(CallLog.Calls.NUMBER, "123456789");
6  newValues.put(CallLog.Calls.DATE, new Date().getTime());
7  newValues.put(CallLog.Calls.DURATION, 30);
8  newValues.put(CallLog.Calls.TYPE, CallLog.Calls.INCOMING_TYPE);
9
10 newUri = getContentResolver().insert(
11    CallLog.Calls.CONTENT_URI,
12    newValues);
13
```

Dobavljači sadržaja

```
1  // Demonstrates the usage of update method
2  ContentValues updateValues = new ContentValues();
3  updateValues.putNull(CallLog.Calls.NUMBER);
4
5  String selectionClause = CallLog.Calls.NUMBER + " LIKE ?";
6  String[] selectionArgs = {"1234567%"};
7
8  int rowsUpdated = getContentResolver().update(
9    CallLog.Calls.CONTENT_URI,
10   updateValues,
11   selectionClause,
12   selectionArgs);
13
```

Dobavljači sadržaja

```
1 // Demonstrates the usage of delete method
2 String selectionClause = CallLog.Calls.NUMBER + " LIKE ?";
3 String[] selectionArgs = {"123456789"};
4
5 int rowsDeleted = getContentResolver().delete(
6   CallLog.Calls.CONTENT_URI,
7   selectionClause,
8   selectionArgs);
9
```

Dobavljači sadržaja

- Razlikuju se sistemski i aplikacioni dobavljači sadržaja
- Sistemski dobavljači sadržaja su uključeni u Android (Browser, Calendar, CallLog, Contacts, MediaStore, Settings, UserDictionary, itd.)
- Aplikacione dobavljače sadržaja pišu programeri koji pišu i ostale komponente aplikacije

Dobavljači sadržaja

Pravljenje aplikacionih dobavljača sadržaja sastoji se iz nekoliko koraka:

- Prvo treba odrediti na koji način će se skladištiti podaci
- Zatim treba naslediti ContentProvider klasu i deklarisati provajder u AndroidManifest.xml
- Na kraju treba definisati odvojenu Contract klasu u kojoj se nalazi ime provajdera, tabela i kolona, kao i prava pristupa

Dobavljači sadržaja

Podaci se mogu skladištiti u bilo kojoj vrsti skladišta. Međutim, trebalo bi skladištiti:

- Strukturirane podatke u SQLite
- Slike (i ostale podatke u binarnom obliku koji zauzimaju dosta prostora) u sistemu datoteka
- Takođe je moguće koristiti klase iz java.net paketa za skladištenje podataka na mreži

AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest ... >
3   <application ... >
4     <provider
5       android:name=".ExampleContentProvider"
6       android:authorities="ftn.mobilneaplikacije.ExampleContentProvider" />
7   </application>
8 </manifest>
9
```

Dobavljači sadržaja

- Prilikom nasleđivanja `ContentProvider` klase obavezno treba implementirati `query()`, `insert()`, `update()`, `delete()`, `getType()` i `onCreate()` metode
- Ove metode imaju iste potpise kao i odgovarajuće metode `ContentResolver` klase
- Sve metode osim metode `onCreate()` moraju biti thread-safe!
- Izbegavati izvršavanje dugačkih operacija u `onCreate()` metodi!
- Ne postoji `onDestroy` metoda (dobavljači sadržaja postoje od početka do kraja procesa)

ExampleContentProvider.java

```
1  public class ExampleContentProvider extends ContentProvider {
2    @Override
3    public boolean onCreate() {
4      // ...
5    }
6
7    @Override
8    public String getType(Uri uri) {
9      // ...
10   }
11
12   @Override
13   public Uri insert(Uri uri, ContentValues values) {
14     // ...
15   }
16
17   @Override
18   public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder) {
19     // ...
20   }
21
22   @Override
23   public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
24     // ...
25   }
26
27   @Override
28   public int delete(Uri uri, String selection, String[] selectionArgs) {
29     // ...
30   }
31 }
32
```

Dobavljači sadržaja

- Contract klasa predstavlja "ugovor" između dobavljača sadržaja i aplikacija koje ga koriste
- Omogućava pristup dobavljaču sadržaja, čak i ako se njegova implementacija promeni (URI, imena tabela, imena kolona, itd.)
- To je javna finalna klasa koja sadrži konstante koje odgovaraju URI-u koji identifikuje podatke, njihovom MIME tipu, imenima tabela i imenima kolona
- Contract klasa takođe olakšava posao programerima zato što obično sadrži konstante koja se lako pamte

ExampleContract.java

```java
public final class ExampleContract {

  // The authority of the content provider.
  public static final String AUTHORITY = "ftn.mobilneaplikacije.ExampleContentProvider";

  // The content URI for the top-level authority.
  public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY);

  // Constants for an example table.
  public final class ExampleTable {

    // The content URI for the top-level authority.
    public static final Uri TABLE_NAME = "TABLE_NAME";

    // The content URI for the top-level authority.
    public static final Uri TABLE_URI = Uri.parse("content://" + AUTHORITY + "/" + TABLE_NAME);

    // The mime type of the content provider.
    public static final Uri MIME_TYPE = "MIME_TYPE";

    // The mime type of the content provider.
    public static final Uri COLUMN_NAME = "COLUMN_NAME";

    // The content URI for the top-level authority.
    public static final Uri COLUMN_URI = Uri.parse("content://" + AUTHORITY + "/" + TABLE_NAME + "/" + COLUMN_NAME);
  }
}

```

Pregled sadržaja

SQLite

- Android aplikacije mogu da koriste ugrađen sistem za upravljanje bazama podataka (SQLite)
- Za razliku od većine sistema za upravljanje bazama podataka, SQLite se izvršava u istom procesu kao i aplikacija koja koristi njegove usluge
- Obezbeđuje referencijalni integritet i omogućava rad u transakcijama

SQLite

- Za pravljenje, izmenu i otvaranje baze podataka koristi se `SQLiteOpenHelper` klasa
- Potrebno je implementirati neke od sledećih metoda:
  - `void onCreate(SQLiteDatabase database)`
  - `void onOpen(SQLiteDatabase database)`
  - `void onUpgrade(SQLiteDatabase database, int old_ver, int new_ver)`
  - `void onDowngrade(SQLiteDatabase database, int old_ver, int new_ver)`

SQLiteOpenHelper.java

```
1  public class ExampleOpenHelper extends SQLiteOpenHelper {
2    private static final String DATABASE_NAME = "NOTEBOOK";
3    private static final int DATABASE_VERSION = 1;
4    private static final String DATABASE_TABLE = "NOTES";
5    private static final String CREATE_DATABASE =
6      "create table " + DATABASE_TABLE + " ( " +
7      "  _id integer primary key autoincrement, " +
8      "  naslov text not null, " +
9      "  vreme  text not null, " +
10     "  tekst  text not null);";
11
12   public ExampleOpenHelper(Context context) {
13     super(context, DATABASE_NAME, null, DATABASE_VERSION);
14   }
15
16   @Override
17   public void onCreate(SQLiteDatabase db) {
18     db.execSQL(CREATE_DATABASE);
19   }
20
21   @Override
22   public void onUpgrade(SQLiteDatabase db, int old_ver, int new_ver) {
23     db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
24     onCreate(db);
25   }
26 }
```

SQLite

- Baza podataka predstavljena je klasom `SQLiteDatabase`.
- CRUD operacije nad bazom podataka izvršavaju se pozivom insert, query, update i delete metoda
  - `long insert(String table, String null_hack, ContentValues entry)`
  - `Cursor query(String table, String[] columns, String whereClause, String[] whereArgs, String groupBy, String having, String orderBy, String limit)`
  - `int update(String table, ContentValues values, String whereClause, String[] whereArgs)`
  - `int delete(String table, String whereClause, String[] whereArgs)`

SQLiteDatabase

```
1 // Connects to the database in write mode
2 SQLiteOpenHelper helper = new ExampleOpenHelper(this.context);
3 SQLiteDatabase db = helper.getWritableDatabase();
4
5 // method getReadableDatabase() gives read-only access
6
```

SQLiteDatabase

```
1 // Demonstrates the usage of insert method
2 ContentValues entry = new ContentValues();
3 entry.put("naslov", "Namirnice");
4 entry.put("vreme", "00:53");
5 entry.put("tekst", "Kupiti hleb i mleko.");
6 long id = db.insert(DATABASE_TABLE, null, entry);
7
```

SQLiteDatabase

```
1  // Demonstrates the usage of query method
2  Cursor c = db.query(
3    DATABASE_TABLE,
4    new String[] {"_id", "naslov", "vreme", "tekst"},
5    "_id = ?",
6    {id},
7    groupBy,
8    having,
9    orderBy,
10   limit);
11
```

SQLiteDatabase

```
1 // Demonstrates the usage of update method
2 ContentValues entry = new ContentValues();
3 entry.put("naslov", "Namirnice");
4 entry.put("vreme", "00:53");
5 entry.put("tekst", "Kupiti hleb i mleko.");
6 long id = db.update(DATABASE_TABLE, entry, whereClause, whereArgs);
7
```

SQLiteDatabase

```
1 // Demonstrates the usage of delete method
2 long id = db.delete(DATABASE_TABLE, "_id = ?", {id});
3
```

Kursori

- Relacija koja je rezultat SQL upita predstavljena je kursorom (`Cursor`)
- Kursori se koriste za navigaciju kroz rezultat upita:
  - `boolean move(int offset)`
  - `boolean moveToFirst()`
  - `boolean moveToLast()`
  - `boolean moveToNext()`
  - `boolean moveToPrevious()`
- kao i za čitanje rezultata upita:
  - `int getCount()`
  - `int getColumnIndex(String column_name)`
  - `String getColumnName(int column_index)`
  - `String getString(int column_index)`
  - `int getInt(int column_index)`
  - `long getLong(int column_index)`
  - `float getFloat(int column_index)`
  - `double getDouble(int column_index)`

SQLiteDatabase

```
1  // Demonstrates reading query results using cursor
2  Cursor cursor = db.query(
3    DATABASE_TABLE,
4    new String[] {"_id", "naslov", "vreme", "tekst"},
5    null, null, null, null, null);
6
7  while(cursor.moveToNext()) {
8    String id = cursor.getInt(0);
9    String naslov = cursor.getString(1);
10   String vreme = cursor.getString(2);
11   String tekst = cursor.getString(3);
12   // ...
13 }
14
```