# Dobavljači sadržaja

## Mobilne aplikacije

Stevan Gostojić

Fakultet tehničkih nauka, Novi Sad

22. novembar 2022.

# Pregled sadržaja

# Deljena podešavanja

- Deljena podešavanja (SharedPreferences) olakšavaju perzistentno skladištenje prostih tipova podataka
- Ti podaci se skladište u datoteci kao uređeni parovi (ključ, vrednost)

# Deljena podešavanja

- Deljenim podešavanjima se može pristupiti metodom `SharedPreferences getSharedPreferences(String name, int mode)`
- Moguće je koristiti više deljenih podešavanje čija imena se navode kao parametar

# Deljena podešavanja

| Konstanta | Opis |
|---|---|
| `MODE_PRIVATE` | The created file can only be accessed by the calling application |

Table 1: Vrednosti parametra "mode".

# Deljena podešavanja

Vrednosti prostog tipa T mogu se zapisati u tri koraka:

1. Pozvati `edit()` metodu koja započinje transakciju
2. Dodati vrednost(i) tipa T metodama oblika
   `SharedPreference.Editor putT(String key, T value)`
3. Pozvati `commit()` metodu koja završava transakciju

# Deljena podešavanja

- Zapisane vrednosti mogu se pročitati metodama oblika `T getT(String key, T defaultValue)`

# ExampleActivity.java

```
1 SharedPreferences settings = getPreferences(Context.
      MODE_PRIVATE);
2 SharedPreferences.Editor editor = settings.edit();
3 editor.putBoolean("silentMode", silentMode);
4 editor.commit();
5
```

# ExampleActivity.java

```
1 SharedPreferences settings = getPreferences(Context.MODE_PRIVATE)
      ;
2 boolean silentMode = settings.getBoolean("silentMode", false);
3
```

# PreferenceActivity

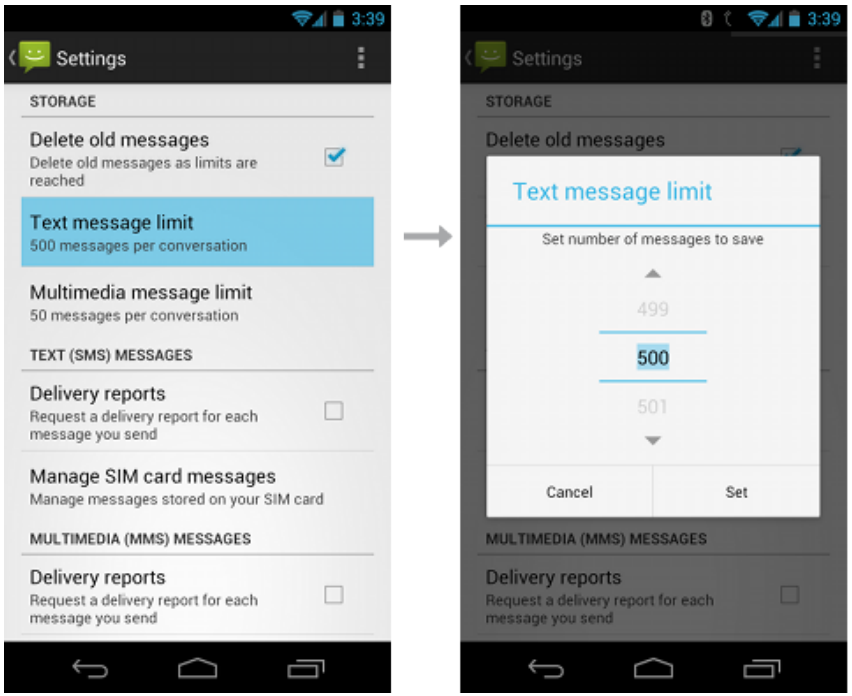


Figure 1: PreferenceActivity.

- Mnoge aplikacije omogućavaju korisnicima da prilagode svoje ponašanje
- U tu svrhu treba koristiti PreferenceActivity kako bi korisnici imali konzistentan grafički korisnički interfejs (i da bi sebi olakšali posao)

# AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest ... >
3   <application ... >
4     <activity android:name=".ExampleActivity">
5       <!-- ... -->
6     </activity>
7   </application>
8 </manifest>
9
```

# ExampleActivity.java

```
1 public class ExampleActivity extends PreferenceActivity
      {
2   @Override
3   public void onCreate(Bundle bundle) {
4     super.onCreate(bundle);
5     addPreferencesFromResource(R.xml.preferences);
6   }
7 }
8
```

# preferences.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/
       android">
3    <CheckBoxPreference
4      android:key="pref_sync"
5      android:title="@string/pref_sync"
6      android:summary="@string/pref_sync_summ"
7      android:defaultValue="true" />
8
9    <ListPreference
10     android:dependency="pref_sync"
11     android:key="pref_syncConnectionType"
12     android:title="@string/pref_syncConnectionType"
13     android:entries="@array/pref_syncConnectionTypes_entries"
14     android:entryValues="@array/pref_syncConnectionTypes_values"
15     android:defaultValue="@string/pref_syncConnectionTypes_default" />
16
17 </PreferenceScreen>
18
```

# Pregled sadržaja

# Datoteke

- Podaci koji se nalaze u operativnoj memoriji se ne čuvaju kada se uništi proces
- Komponente koje se nalaze u različitim procesima ne mogu da razmenjuju podatke koji se nalaze u operativnoj memoriji (ne dele adresni prostor)
- Najjednostavniji način da se prevaziđu ova ograničenja je korišćenje datoteka (Android je ipak distribucija Linux-a)

# Datoteke

- Za rad sa datotekama koriste se klase iz java.io paketa (na isti način na koji se koriste u Java SE)
- Međutim, mogu se koristiti metode klase Context koje olakšavaju pristup internom i/ili eksternom skladištu podataka, rad sa privremenim datotekama i upravljanje pravima pristupa
  - `FileInputStream openFileInput(String name)`
  - `FileOutputStream openFileOutput(String name, int mode)`
  - `String[] fileList()`
  - `boolean deleteFile(String name)`
  - `File getDir(String name, int mode)`
  - `File getCacheDir()`
  - `File getExternalCacheDir()`
  - `File getFilesDir()`
  - `File getExternalFilesDir(String type)`

# ExampleService.java

```
1  private void write() {
2    FileOutputStream fos = null;
3    try {
4      fos = openFileOutput("test.txt", Context.
       MODE_PRIVATE);
5      fos.write(bytes);
6    } catch (FileNotFoundException e) {
7      Log.e(Constants.LOG_TAG, "File not found", e);
8    } catch (IOException e) {
9      Log.e(Constants.LOG_TAG, "IO problem", e);
10   } finally {
11     try {
12       fos.close();
13     } catch (IOException e) {
14     }
15   }
16 }
17
```

# Datoteke

| Konstanta | Opis |
|---|---|
| `MODE_PRIVATE` | The created file can only be accessed by the calling application |
| `MODE_APPEND` | if the file already exists then write data to the end of the existing file instead of erasing it |
| `MODE_WORLD_READABLE` | Allow all other applications to have read access to the created file (deprecated) |
| `MODE_WORLD_WRITEABLE` | Allow all other applications to have write access to the created file (deprecated) |

Table 2: Vrednosti parametra "mode".

# ExampleService.java

```
1  private void read() {
2    FileInputStream fis = null;
3    Scanner scanner = null;
4    StringBuilder sb = new StringBuilder();
5
6    try {
7      fis = openFileInput("test.txt");
8      scanner = new Scanner(fis);
9      while (scanner.hasNextLine()) {
10       sb.append(scanner.nextLine() + LINE_SEP);
11     }
12   } catch (FileNotFoundException e) {
13     Log.e(Constants.LOG_TAG, "File not found", e);
14   } finally {
15     if (fis != null) {
16       try {
17         fis.close();
18       } catch (IOException e) {
19       }
20     }
21     if (scanner != null) {
22       scanner.close();
23     }
24   }
25 }
26
```

# Datoteke

- Interno skladište podataka se nalazi u mobilnom uređaju
  - Uvek je dostupno
  - Obično je manjeg kapaciteta (i nije ga moguće proširiti)
  - Privatno je
- Eksterno skladište podataka se (obično) nalazi na SD kartici
  - Nije uvek dostupno
  - Obično je većeg kapaciteta (i moguće ga je proširiti)
  - Javno je

# ExampleService.java

```
1  String state = Environment.getExternalStorageState();
2  if (Environment.MEDIA_MOUNTED.equals(state)) {
3      // We can read and write the media
4  } else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
5      // We can only read the media
6  } else {
7      // Something else is wrong. It may be one of many other states,
8      // but all we need to know is we can neither read nor write
9  }
10
```

# Datoteke

- Privremene datoteke treba skladištiti u cache direktorijumu (Android ih automatski briše kada ponestane slobodnog prostora)
  - `File getCacheDir()`
  - `File getExternalCacheDir()`
- Datoteke koje deli više aplikacija treba snimiti u javni eksterni direktorijum
  - `File getExternalStoragePublicDirectory(String type)`

# Datoteke

| Konstanta | Opis |
|---|---|
| DIRECTORY_ALARMS | Standard directory in which to place alarms |
| DIRECTORY_DCIM | The traditional location for pictures and videos when mounting the device as a camera. |
| DIRECTORY_DOCUMENTS | Standard directory in which to place documents that have been created by the user. |
| DIRECTORY_DOWNLOADS | Standard directory in which to place files that have been downloaded by the user. |
| DIRECTORY_MOVIES | Standard directory in which to place movies that are available to the user. |

Table 3: Tip javnog eksternog direktorijuma.

# Datoteke

| Konstanta | Opis |
|---|---|
| DIRECTORY_MUSIC | Standard directory in which to place any audio files that should be in the regular list of music for the user. |
| DIRECTORY_NOTIFICATIONS | Standard directory in which to place any audio files that should be in the list of notifications that the user can select. |
| DIRECTORY_PICTURES | Standard directory in which to place pictures that are available to the user. |
| DIRECTORY_PODCASTS | Standard directory in which to place any audio files that should be in the list of podcasts that the user can select (not as regular music). |
| DIRECTORY_RINGTONES | Standard directory in which to place any audio files that should be in the list of ringtones that the user can select (not as regular music). |

Table 4: Tip javnog eksternog direktorijuma.

# 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

# sqlite3

| Naredba | Opis |
|---|---|
| `.databases` | Lists names and files of attached databases. |
| `.tables ?TABLE?` | Lists names of tables (if TABLE is specified, only dumps tables matching LIKE pattern TABLE). |
| `.dump ?TABLE? ...` | Dumps the database in an SQL text format (if TABLE is specified, only dumps tables matching LIKE pattern TABLE). |
| `.schema ?TABLE?` | Shows the CREATE statements (if TABLE is specified, only dumps tables matching LIKE pattern TABLE). |
| `.backup ?DB? FILE` | Backups database (default "main") to FILE. |
| `.restore ?DB? FILE` | Restores content of the database (default "main") from FILE. |

Table 5: Sqlite3 naredbe.

# sqlite3

| Naredba | Opis |
|---|---|
| `.read FILENAME` | Executes SQL in FILENAME. |
| `.import FILE TABLE` | Imports data from FILE into TABLE. |
| `.headers on\|off` | Turns display of headers on or off. |
| `.mode MODE ?TABLE?` | Set output mode where MODE is one of: csv (comma-separated values), column (left-aligned columns), html (HTML <table> code), insert (SQL insert statements for TABLE), line (one value per line), list (values delimited by .separator string), tabs (tab-separated values) or tcl (TCL list elements) |
| `.nullvalue STRING` | Use STRING in place of NULL values. |
| `<sql statement>` | Može se izvršiti i proizvoljna SQL naredba. |

Table 6: Sqlite3 naredbe.

# sqlite3

```
1 > adb -s device_name shell
2 > sqlite3 /path_to_database/db_name.db
3 SQLite version 3.3.12
4 Enter ".help" for instructions
5 .... enter commands, then quit...
6 sqlite>.exit
7 > _
8
```

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

# 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
```

# SQLiteDatabase

```
1 // Demonstrates the usage of instert 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, TITLE, TIMESTAMP, TEXT},
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(
3   DATABASE_TABLE,
4   "_ID = ?",
5   {id});
6
```

# 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)`

# 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 7: Primer dobavljenih podataka.

| word | app id | frequency | locale | _ID |
|---|---|---|---|---|
| mapreduce | user1 | 100 | en_US | 1 |
| precompiler | user14 | 200 | fr_FR | 2 |
| applet | user2 | 255 | fr_CA | 3 |
| const | user1 | 255 | pt_BR | 4 |
| int | user5 | 100 | en_UK | 5 |

# 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 obezbeđen 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_USER_DICTIONARY"
      >
4   <uses-permission android:name="android.permission.WRITE_USER_DICTIONARY
      ">
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  String[] projection =
2  {
3     UserDictionary.Words._ID,
4     UserDictionary.Words.WORD,
5     UserDictionary.Words.LOCALE
6  };
7
8  String selectionClause = null;
9
10 String[] selectionArgs = {""};
11
12 String sortOrder = null;
13
14 cursor = getContentResolver().query(
15    UserDictionary.Words.CONTENT_URI,
16    projection,
17    selectionClause,
18    selectionArgs,
19    sortOrder);
20
```

# 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(UserDictionary.Words.APP_ID, "example.user
       ");
6  newValues.put(UserDictionary.Words.LOCALE, "en_US");
7  newValues.put(UserDictionary.Words.WORD, "insert");
8  newValues.put(UserDictionary.Words.FREQUENCY, "100");
9
10 newUri = getContentResolver().insert(
11    UserDictionary.Word.CONTENT_URI,
12    newValues);
13
```

# Dobavljači sadržaja

```
1  // Demonstrates the usage of update method
2  ContentValues updateValues = new ContentValues();
3  String selectionClause = UserDictionary.Words.LOCALE +  "
       LIKE ?";
4  String[] selectionArgs = {"en_%"};
5  int rowsUpdated = 0;
6  updateValues.putNull(UserDictionary.Words.LOCALE);
7  rowsUpdated = getContentResolver().update(
8    UserDictionary.Words.CONTENT_URI,
9    updateValues,
10   selectionClause,
11   selectionArgs);
12
```

# Dobavljači sadržaja

```
1 // Demonstrates the usage of delete method
2 String selectionClause = UserDictionary.Words.APP_ID + " LIKE
      ?";
3 String[] selectionArgs = {"user"};
4 int rowsDeleted = 0;
5 rowsDeleted = getContentResolver().delete(
6   UserDictionary.Words.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 dobavljače 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="rs.ac.uns.ftn.informatika.ExampleContentProvider"
6       android:authorities="rs.ac.uns.ftn.informatika.
      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 operaciju u `onCreate()` metodi!
- Ne postoji `onDestory` 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 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

```
1  public final class ExampleContract {
2
3    /*
4     * The authority of the content provider.
5     */
6    public static final String AUTHORITY = "de.openminds.
       samples.cpsample.lentitems";
7
8    /*
9     * The content URI for the top-level authority.
10    */
11   public static final Uri CONTENT_URI = Uri.parse("content://
       " + AUTHORITY);
12
13   /*
14    * Constants for an example table.
15    */
16   public final class ExampleTable {
17
18     /*
19      * The content URI for the top-level authority.
20      */
21     public static final Uri TABLE_NAME = "TABLE_NAME";
22
23     /*
24      * The content URI for the top-level authority.
25      */
26     public static final Uri TABLE_URI = Uri.parse("content://
       " + AUTHORITY + "/" + TABLE_NAME);
27
28     /*
29      * The mime type of the content provider.
30      */
31     public static final Uri MIME_TYPE = "MIME_TYPE";
32
33     /*
34      * The mime type of the content provider.
35      */
36     public static final Uri COLUMN_NAME = "COLUMN_NAME";
37
38     /*
39      * The content URI for the top-level authority.
40      */
41     public static final Uri COLUMN_URI = Uri.parse("content
       ://" + AUTHORITY + "/" + TABLE_NAME + "/" + COLUMN_NAME);
42   }
43 }
44
```

# Pregled sadržaja

# Punjači

- Punjači (`Loader`) omogućavju asinhrono učitavanje podataka u aktivnosti ili fragmente
- Nadgledaju izvore podataka i isporučuju sadržaj kada se podaci promene
- Vode računa o promeni stanja aktivnosti ili fragmenta

# Punjači

- `SimpleCursorAdapter` povezuje kursor sa `ListView` ili `GridView` pogledom
- `LoaderManager` upravlja punjačima
- Metode `LoaderManager.LoaderCallbacks` interfejsa se koriste za komunikaciju sa punjačem
- `CursorLoader` je implementacija punjača koja učitava podatke iz dobavljača sadržaja u pozadinskoj niti

# ExampleActivity.java

```
1  protected void onCreate(Bundle bundle) {
2    // Create an empty adapter we will use to display the
       loaded data.
3    adapter = new SimpleCursorAdapter(
4      getActivity(),
5      android.R.layout.simple_list_item_2,
6      null,
7      new String[] {Contacts.DISPLAY_NAME, Contacts.
       CONTACT_STATUS},
8      new int[] {android.R.id.text1, android.R.id.text2},
9      0);
10
11   setListAdapter(adapter);
12
13   // Either re-connect with an existing one, or start a
       new one.
14   getLoaderManager().initLoader(
15     0,
16     null,
17     this);
18   }
19
```

# Punjači

Table 8: Parametri konstruktora SimpleCursorAdapter-a

| Parametar | Opis |
|---|---|
| `context` | Context of the `ListView`. |
| `layout` | Resource identifier of a layout defining the views for this item. |
| `cursor` | Database cursor. |
| `from` | List of column names representing the binded data. |
| `to` | Views that should display cursor columns. |
| `flags` | Flags used to determine the behavior of the adapter. |

# Punjači

Table 9: Parametri initLoader metode.

| Parametar | Opis |
|---|---|
| `id` | A unique identifier for this loader (scoped to a particular `LoaderManager` instance). |
| `args` | Optional arguments to supply to the loader at construction. |
| `callback` | Interface the `LoaderManager` will call to report about changes in the state of the loader. |

# ExampleActivity.java

```
1  public Loader<Cursor> onCreateLoader(int id, Bundle args) {
2    // This is called when a new Loader needs to be created.
3    // This sample only has one Loader, so we don't care about the
       ID.
4
5    // Create and return a CursorLoader that will take care of
6    // creating a Cursor for the data being displayed.
7    return new CursorLoader(
8      getActivity(),
9      dataUri,
10     projection,
11     selection,
12     selectionArgs,
13     sortOrder);
14 }
15
```

# ExampleActivity.java

```
1 public void onLoadFinished(Loader<Cursor> loader, Cursor data
      ) {
2   // Swap the new cursor in (the framework will take care of
3   // closing the old cursor once we return).
4   adapter.swapCursor(data);
5 }
6
```

# ExampleActivity.java

```
1 public void onLoaderReset(Loader<Cursor> loader) {
2   // This is called when the last Cursor provided to
3   // onLoadFinished() above is about to be closed. We
4   // need to make sure we are no longer using it.
5   adapter.swapCursor(null);
6 }
7
```

# Punjači

Table 10: Parametri konstruktora CursorLoader-a.

| Parametar | Opis |
|---|---|
| `flags` | The URI for the content to retrieve. |
| `projection` | A list of which columns to return. |
| `selection` | A filter declaring which rows to return. |
| `selectionArgs` | You may include ?s in the selection, which will be replaced by the values from selectionArgs. |
| `sortOrder` | How to order the rows. |