Dobavljači sadržaja

Pregled sadržaja

1 Deljena podešavanja

2 Datoteke

3 SQLite

4 Dobavljači sadržaja

5 Punjači

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 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

Tabela: 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>
```

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);
```

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);
```

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);
```

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);
```

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>
```

ExampleContentProvider.java

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

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!

ExampleContract.java

```java
public final class ExampleContract {

  /*
   * The authority of the content provider.
   */
  public static final String AUTHORITY = "de.openminds.samples.cpsample.lentitems";

  /*
   * 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);
  }
}
```

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

Pregled sadržaja

1. Deljena podešavanja
2. Datoteke
3. SQLite
4. Dobavljači sadržaja
5. Punjači

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   }
```

Punjači

Tabela: 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

Tabela: 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 }
```

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 }
```

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 }
```

Punjači

Tabela: 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.

All images copyrighted by Android Open Source Project (CC BY)