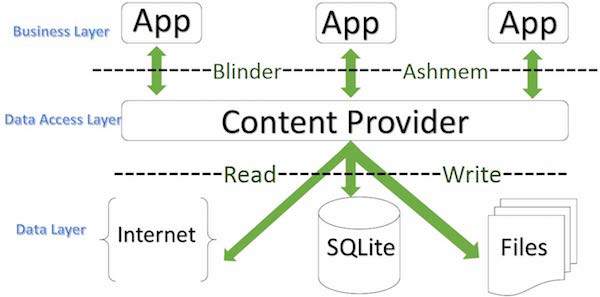
**Content Provider**

# Uvod

**Content Provider** je Android komponenta koja omogućava siguran mehanizam pristupa resursima, npr. bazi podataka kao i deljenje resursa sa drugim aplikacijama. Content Provider može biti privatan, vidljiv samo za datu aplikaciju i tada se koristi kao abstrakcija nivoa za rukovanje resursima, međutim u češćem slučaju koristi se za deljenje podataka sa drugim aplikacijama. U drugom slučaju omogućava pristupanje podacima od strane drugih, klijentskih aplikacija koje šalju zahteve za pribavljanje ili menjanje podataka. Content Provider može da čuva i deli podatke koji se skladište unutar baze podataka, u fajlovima ili preko mreže. On omogućava centralizovanje podataka na jednom mestu odakle više aplikacija može da pristupa podacima po potrebi. Slično kao i rukovanje sa bazom podataka, Content Provider omogućava osnovne metode za dobavljanje i rukovanje podacima. U najčešćem slučaju podaci su smešteni u SQLite bazi podataka.



# Pristupanje podacima

Kako bi aplikacija pristupila podacima preko content provider-a potrebno je da zna njegov URI (eng. Uniform Resource Identifier). URI je jedinstveni identifikator resursa koji se sastoji iz više delova od kojih svaki ima bitno značenje za pristupanje podacima. Njegova forma je :

<PREFIX>://<AUTHORITY>/<DATA\_TYPE>/<ID>

* **PREFIX** – uvek se koristi: *content://*
* **AUTHORITY** – Specificira naziv content provider-a, najčešće se koristi naziv paketa aplikacije kako bi naziv bio jedinstven
* **DATA\_TYPE** - Označava tip podataka koji content provider omogućava
* **ID** – Specificira tačan identifikator resursa koji aplikacija zahteva.

Primeri URI-a:

***content://contacts/people/5***

***content://com.rtrk.pnrs.project/students/11***

# ContentProvider klasa

Kada se implementira Content Provider potrebno je naslediti ContentProvider klasu koja definiše osnovne metode koje je potrebno implementirati kako bi se sa njim rukovalo.

* onCreate – callback metoda koja se poziva kada je content provider pokrenut
* query – ova metoda se poziva kada klijent zahteva selekciju podataka, i vraća Cursor objekat
* insert – omogućava stavljanje novog podatka u bazu preko Content Provider-a
* update – omogućava ažuriranje podataka
* delete – briše podatke
* getType – vraća MIME tip podatka definisan zadatom URI adresom (MIME označava tip podataka koji se preuzima, sastoji se iz dva dela, i najčešće se koristi u različitim WEB protokolima kao što je HTTP, npr. text/html, image/png, itd...)

# Registrovanje Content Provider komponente

Kako bi se Content Provider koristio, potrebno je njegovo registrovanje u *AndroidManifest.xml* fajlu. Ukoliko se klasa generiše uz pomoć Android Studio okruženja, automatski će biti registrovan.

Generisanje klase : file > new > other > content provider

Postoje više atributa koji definišu kako će se pristupati podacima, npr. atribut ***permission*** definiše permisije koje klijent mora imati kako bi čitao ili menjao podatke preko Content Provider-a.

Moguće je odabrati da li će dati Content Provider biti privatan ili public preko atributa "**exported**".

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.MyApplication">

<application

. . .>

. . .

**<provider android:name="StudentsProvider"**

**android:authorities="com.example.MyApplication.StudentsProvider"**

**android:exported="false"/>**

</application>

</manifest>

# UriMatcher klasa

Ova klasa omogućava definisanje svih mogućih URI-a koje dati Content Provider koristi za pristupanje resursima. Potrebno je formirati stablo koristeći UriMatcher objekat koji sadrži sve URI adrese. Sledi primer inicijalizacije UriMatcher objekta.

private static final int PEOPLE = 1;

private static final int PEOPLE\_ID = 2;

private static final int PEOPLE\_PHONES = 3;

private static final int PEOPLE\_PHONES\_ID = 4;

private static final UriMatcher sURIMatcher = new UriMatcher(UriMatcher.NO\_MATCH);

static {

sURIMatcher.addURI("contacts", "people", PEOPLE);

sURIMatcher.addURI("contacts", "people/#", PEOPLE\_ID);

sURIMatcher.addURI("contacts", "people/#/phones", PEOPLE\_PHONES);

sURIMatcher.addURI("contacts", "people/#/phones/#", PEOPLE\_PHONES\_ID);

}

Ova polja su najčešće definisana kao *static* polja kako bi UriMatcher objekat bio inicijalozovan pre kreiranja bilo kojeg drugog objekta, tj. za vreme učitavanja klase.

Private static konstante iz navedenog primera, koje su definisane kao celi pozitivni brojevi se koriste za odabiranje jedne od selekcija na osnovu šablona iz zahteva.

Takođe je moguće koristiti posebne karaktere **#** i **\***, gde # označava bilo koji broj a \* bilo koji karakter.

# Content Resolver Klasa

Ova klasa omogućava selekciju željenog Content Provider-a i pozivanje njegovih funkcionalnosti (query, insert, delete, update,..itd) na osnovu prosleđene URI adrese. Može se reći da resolver predstavlja spregu između Content Provider-a i aplikacije, koja zahteva pristup resursima. Dobavljanje instance ContentResolver klase se dobavlja na sledeći način:

ContentResolver resolver = getContentResolver();

Sledeća slika prikazuje vezu između aplikacije koja pristupa podacima iz SQLite baze podataka preko Content Provider-a i ContentResolver klase.

