NTNU
Kunnskap for en bedre verden

Stedsbaserte tjenester, innholdstilbydere og sensorer

Tomas Holt, Institutt for datateknologi og informatikk ved NTNU
20.09.2018
Lærestoffet er utviklet for faget IFUD1042 Applikasjonsutvikling for mobile enheter

Resymé: Denne leksjonen tar for seg stedsbaserte tjenester (Location-Based Services - LBS), innholdstilbydere (Content Providers) og sensorer på Android-enheter. I tillegg avrundes det hele med å henvise til kilder for å kunne distribuere Android-applikasjoner og hva som er "best practice" for slike applikasjoner.

Contents

10 Stedsbaserte tjenester, innholdstilbydere og sensorer

10.1 Om leksjonen

I denne leksjonen kikker vi litt på innholdstilbydere, stedsbasertetjenester, sensorer, publisering av applikasjoner og "best practice" for Android-applikasjoner.

Vi har da vært igjennom mye av læreboka og også tatt opp en del som ikke er tatt opp der. Det som i hovedsak ikke blir er blitt gjennomgått er da det som har med bilder, video og lyd (kap. 19,20,21), samt noe om grafisk grensesnitt.

10.2 Innholdstilbyder

Kapittel 27 i boka omhandler innholdstilbydere (eng; Content Provider). Innholdstilbydere er spesielt egnet om man ønsker å dele data mellom applikasjoner. Et eksempel på slik deling av informasjon er kontaktinformasjon. Som brukere lagrer vi gjerne personer vi kjenner med telefonnummer, epost-adresse osv. Denne informasjonen lagres i en database på telefonen og kan være interessant for mange forskjellige applikasjoner (f.eks. epost-applikasjon, telefonapplikasjon osv.). En innholdstilbyder gjør altså innholdet i databasen (i dette tilfellet) tilgjengelig for alle applikasjoner som har rettigheter til å bruke denne innholdstilbyderen (vi som brukere avgjør det ved installasjon av applikasjonen).

Bok har et greit eksempel som går på å bruke en innholdstilbyder. Her vises hvordan man søker, sorterer og filtrerer innhold. Det er altså mulig å lage innholdstilbyder for å dele ”egne data”. Det kan være data fra databaser, filer og over nettverk. Min umiddelbare erfaring er at rammeverket (klassene) for å lage innholdstilbydere er knyttet rimelig sterkt mot databaser. Bruker man en SQLite database så er det ikke alt for mye jobb å lage en egen innholdstilbyder. Skal man derimot gjøre det samme for filer eller nettverksdata så er saken en annen. Her må man implementere mye fra bunnen av og feltene som er naturlig i forbindelse med databaseaksess (sql-kode) er ikke lenger like naturlige. Ergo kan det også bli uvant/unormalt for de som ønsker å bruke innholdstilbyderen.

Man bør derfor tenke seg om før man starter jobben med å lage en egen innholdstilbyder. Ønsker man å dele data mellom applikasjoner og disse ligger i en database så lag en innholdstilbyder. I de andre tilfellene bør man nok tenke seg om før man tyr til å lage en innholdstilbyder. I ”mindre proffe løsninger” kan man i mange tilfeller gjøre det man ønsker via SharedPreferences eller filer hvor tilgang gis alle applikasjoner (husk også sikkerhet). Andre alternativer kan være web-tjenester eller andre typer nettverksløsninger (her kan sikkerhet styres på tjenersiden).

Før du leser resten av dette kapitlet i leksjonen kan det være lurt å lese kapittel 27 i boka. Du kan imidlertid fint hoppe til neste kapittel.

Jeg har laget en liten innholdstilbyder som illustrerer noe av det som må på plass for å tilby aksess til data på fil (bok/forfattereksemplet fra tidligere leksjon/øving). File ligger i eksempel-prosjektet under res -> raw -> books.txt. **Jeg gjør imidlertid oppmerksom på at denne kun er ment som et ufullstendig eksempel** (utskrift skjer f.eks. til loggen), slik at man ser hva man begir seg ut på - og kanskje hva man ikke bør gjøre - om man

velger å gjøre denne typen implementasjon. La oss se på eksemplet (koden ligger også vedlagt).

Den naturlige plassen å starte er på klassen som arver fra ContentProvider, her FileBookContentProvider. I denne implementasjonen er det kun tre metoder som er av interesse; onCreate(), getType() og query(). Førstnevnte vil ikke kalles direkte av oss, men av Android-implementasjonen. Her har vi mulighet til å gjøre initialiseringer.

Metoden getType() gir mulighet for klientapplikasjoner å finne ut hva slags data denne innholdstilbyderen faktisk tilbyr.

Den interessante metoden er query(). Det er denne vi bruker for å hente ut informasjon. Metoden tar mange innargumenter som gir god mening når det gjelder databaser. I denne implementasjonen tar vi kun hensyn til uri. Vi tar ikke hensyn sorteringsrekkefølge (sortOrder), filtrering (selection/selectionArgs) og felter (projection) som ønskes. Her returneres ganske enkelt en egenlaget Cursor (FileCursor) som inneholder alle bøker og tilhørende forfattere lest inn fra fil.

```
1  package no.hist.aitel.android.leksjon.lagringstilbyder;
2
3  import java.util.ArrayList;
4
5  import android.content.ContentProvider;
6  import android.content.ContentValues;
7  import android.content.UriMatcher;
8  import android.database.Cursor;
9  import android.net.Uri;
10 import android.util.Log;
11 import no.hist.itfag.oving5.FileHandler;
12
13
14 public class FileBookContentProvider extends ContentProvider {
15         public static final String TAG="FileBookContentProvider";
16     public static final String PROVIDER_NAME =
17         "no.hist.aitel.android.leksjon.lagringstilbyder.FileBookContentProvider";
18
19      public static final Uri CONTENT_URI =
20         Uri.parse("content://"+ PROVIDER_NAME + "/books");
21
22      private static final int BOOKS = 1;
23      private static final int BOOK_ID = 2;
24
25      private static final UriMatcher uriMatcher;
26      static{ //request string (uri) tells us what (cursor to return) - used in the
             getType() and query() method
27         uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
28         uriMatcher.addURI(PROVIDER_NAME, "books", BOOKS);
29         uriMatcher.addURI(PROVIDER_NAME, "books/#", BOOK_ID);
30      }
31
32     @Override
33     public String getType(Uri uri){
34        Log.i(TAG,"getType " + uri);
35        switch (uriMatcher.match(uri)){
36           //---get all books---
37           case BOOKS:
38              return "vnd.android.cursor.dir/vnd.books "; //this is explanation of
                    what's returned
```

```
39            //---get a particular book---
40            case BOOK_ID:
41               return "vnd.android.cursor.item/vnd.books ";//this is explanation of
                     what's returned
42            default:
43               throw new IllegalArgumentException("Unsupported URI: " + uri);
44         }
45      }
46
47      @Override
48      public boolean onCreate() {
49         Log.i(TAG,"onCreate()");
50         return true;
51      }
52
53      /* Returns a book or books with its authors. Projection,selection,selectionArgs
            and sortOrder are not "implemented". */
54      @Override
55      public Cursor query(Uri uri, String[] projection, String selection,String[]
            selectionArgs, String sortOrder) {
56         Log.i(TAG,"query() start.... mot URI=" + uri.getPath());
57         FileCursor cursor = null;
58         ArrayList<Book> books = new ArrayList<Book>();//keep books
59         books = loadAllBooksFromFile();
60
61         if (books.size() <= 0) return null;
62
63         int requestType = uriMatcher.match(uri);
64         if (requestType == BOOK_ID){ //BOOK_ID ---if getting a particular book---
65             int i = Integer.parseInt(uri.getLastPathSegment());//get book number
66             cursor =  new FileCursor(books.subList(i-1, i));//only one book in cursor
67         }else if (requestType == BOOKS){
68             cursor =  new FileCursor(books);//all books in cursor
69         }
70         return cursor;
71      }
72
73      //loads all the books on file!
74      private ArrayList<Book> loadAllBooksFromFile(){
75         ArrayList<Book> books = new ArrayList<Book>();//keep books
76         FileHandler fh = new FileHandler(getContext(),R.raw.books);
77         String fromFile = fh.readAll();
78         Log.i(TAG,"FromFile=" + fromFile);
79         String[] allFileArray = fromFile.split("\n");
80
81         ArrayList<String> authors = new ArrayList<String>();
82
83         int id = 0;
84         for (String s : allFileArray){
85             if (s.contains("Author")){
86                     int indexOfAuthorName = s.indexOf(": ") + 2;
87                     authors.add(s.substring(indexOfAuthorName));
88             }else{
89                     int indexOfBookName = s.indexOf(": ") + 2;
90                     String bookName = s.substring(indexOfBookName);
91                     Book b = new Book(++id,bookName,authors);
92                     books.add(b);
93                     authors.clear();//next iteration will get a new book and new authors
94             }
95         }
96         return books;
97      }
```

```
98
99      /* NOT implemented => not possible to add, update or delete content */
100         @Override
101         public Uri insert(Uri uri, ContentValues values) {
102             return null;
103         }
104
105         @Override
106         public int update(Uri uri, ContentValues values, String selection, String[]
                selectionArgs) {
107            return 0;
108         }
109
110         @Override
111         public int delete(Uri uri, String selection, String[] selectionArgs) {
112                 // TODO Auto-generated method stub
113                 return 0;
114         }
115
116 }
```

Neste steg blir å lage en implementasjon for Cursor som returneres fra query()-metoden. Denne inneholder kun det aller nødvendigste for å fungere og jeg oppfordrer derfor til å kikke i CursorWrapper-klassen (som klassen arver fra) for å få en bedre ide om alle metodene det kan være aktuelt å implementere i en mer generell og fullstendig implementasjon. Klassen er her slik:

```
1  /* This is a very simple and incomplete example of a cursor implementation.
2   * There are a lot of methods inherited from CursorWrapper which should/could be
3   * implemented in this class.
4   */
5
6  package no.hist.aitel.android.leksjon.lagringstilbyder;
7
8  import java.util.List;
9  import android.database.CursorWrapper;
10
11 public class FileCursor extends CursorWrapper{
12         public final static String TAG="FileCursor";
13         private List<Book> books;
14         private int position=0;//which books are wanted
15
16
17
18         public FileCursor(List<Book> books_Authors){
19                 super(null);
20                 this.books = books_Authors;
21         }
22
23         public String[] getColumnNames(){
24                 String[] s = {"Book","Authors"};
25                 return s;
26         }
27
28         public int getPosition(){
29                 return position;
30         }
31
32         public String getString(int columnIndex){
33                 return books.get(position).toString();
```

```
34          }
35
36          /* This method must be implemented to get example working */
37          @Override
38          public int getCount(){
39                  return books.size();
40          }
41
42          /* This method must be implemented to get example working */
43          @Override
44          public boolean moveToNext(){
45                  boolean isOk = false;
46                  if (position < books.size()-1){
47                          position++;
48                          isOk = true;
49                  }
50                  return isOk;
51          }
52  }
```

Så kommer en hjelpeklassen for å holde på bok og dets forfattere:

```
1   package no.hist.aitel.android.leksjon.lagringstilbyder;
2
3   import java.util.ArrayList;
4
5   public class Book {
6           private int id;
7           private String name;
8           private ArrayList<String> authors;
9
10          public Book(int id, String name, ArrayList<String> authors){
11                  this.name = name;
12                  this.authors = authors;
13          }
14
15          public int getId(){
16                  return id;
17          }
18
19          public String getName(){
20                  return name;
21          }
22
23          public String getAuthors(){
24                  String tmp = "";
25                  for (String s : authors){
26                          tmp += s;
27                  }
28                  return tmp;
29          }
30
31          public String toString(){
32                  String ret = name + "\n";
33                  for (String s : authors){
34                          ret += s + "\n";
35                  }
36                  return ret;
37          }
38  }
```

Tilslutt så har vi en veldig enkel aktivitet som tester det hele:

```
1  /* T.H 19/10-2012 */
2
3  package no.hist.aitel.android.leksjon.lagringstilbyder;
4
5  import android.app.Activity;
6  import android.content.ContentUris;
7  import android.database.Cursor;
8  import android.net.Uri;
9  import android.os.Bundle;
10 import android.util.Log;
11
12 public class ContentTestActivity extends Activity {
13         public final static String TAG = "TEST_ACTIVITY";
14     @Override
15     public void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.main);
18         Log.i(TAG,"Starting ....");
19
20         Uri uri = FileBookContentProvider.CONTENT_URI;//gets all books - use uri below
               to get only one book
21
22         //Uri uri = ContentUris.withAppendedId(FileBookContentProvider.CONTENT_URI, 1);
23         //=>
               content://no.hist.aitel.android.leksjon.lagringstilbyder.FileBookContentProvider/books/1
24
25         Cursor fileCursor = getContentResolver().query(uri, null, null, null, null);
26         //for new versions of android use CursorLoader.loadInBackground();
27
28         int column = 1;//doesn't matter with this implementation
29         String bookWithAuthors = fileCursor.getString(column);
30         Log.i(TAG,bookWithAuthors);//write book/authors to log
31
32         //methods as isAfterLast, isBeforeFirst() and many other are not
               implemented!!!! only moveToNext()
33         fileCursor.moveToNext();
34         bookWithAuthors = fileCursor.getString(column);
35         Log.i(TAG,bookWithAuthors);//write book/authors to log
36     }
37 }
```

Ved kjøring av koden vil man se at bok/bøker og tilhørende forfattere skrives ut. En slik implementasjon kan virke naturlig grunnet hvordan dataene er lagret på fila (bok - forfattere - bok - forfattere osv.). I forhold til måten man har lagt opp bruken av ContentProvider og Cursor blir derimot dette ikke logisk. Du kan se et eksempel på det på følgende linjer i aktiviteten over:

```
int column = 1;//doesn't matter with this implementation
String bookWithAuthors = fileCursor.getString(column);
```

Det som skjer her er at vi henter ut bok sammen med forfattere. I Cursor er det lagt opp til at man skal kunne velge hvilke kolonner man vil vise. Bok kan da kanskje være kolonne 0 mens, forfatter er kolonne 1. Her er det imidlertid ikke en-til-en forhold, men mange forfattere for hver bok. Å bruke kolonner vil derfor være meningsløst (da man kun kan vise en av forfatterene). Vi ser dermed nok et eksempel på hvordan databasetankegangen gjør seg gjeldende. Den mest fornuftige løsningen vil derfor være

å håndtere bøker og forfatter hver for seg. Da kan man hente en bok for så å gjøre en spørring som henter forfatterene til boka (evt. motsatt). Om vi skal implementere en slik løsning vil det imidlertid være naturlig å bruke en database, da det er nettopp denne typen sammenhenger/spørringer de gjør så bra - samtidig som vi slipper å skrive unødvendig mye kode.

10.3 Stedsbasertetjenester

Stedsbaserte tjenester (eng; location-based services) ofte forkortet LBS har etterhvert blitt veldig vanlig.

LBS [Android LBS] brukes f.eks. for å finne ut og vise hvor man befinner seg. Ettersom mange enheter i dag kommer med GPS så kan denne brukes. Det er imidlertid også mulig å bestemme posisjon på andre måter. Dette kan f.eks. være krysspeiling av mobilsendere eller bruk av WiFi-nettverk. Hva som er mest hensiktsmessig å bruke kan variere avhengig av situasjon (batteribruk, nøyaktighet osv.).

For å visualisere posisjoner er det vanlig å bruke kart. I standard Android-emulatorer er det ikke med kartmuligheter, men om man lager en emulator/AVD med støtte for Google API så er Kart(eng; Maps)-applikasjonen med. Du kan selv starte denne applikasjonen (den ligger sammen med de andre applikasjonene i emulatoren) og trykke meny for å se hvilket muligheter som finnes.

Fra din egen Android-kode kan du bruke Kart-applikasjonen via en intent. Følgende kode angir lengde- og breddegrad for så å vise dette i Kart:

```
double latitude = 59.916667;//lengdegrad
double longitude = 10.75;//breddegrad
String strLocation = String.format("geo:%f,%f",latitude,longitude);//formattering
Uri location = Uri.parse(strLocation);//omformer til URI
Intent map = new Intent(Intent.ACTION_VIEW,location);//vis i kart-applikasjonen
startActivity(map);
```

Koden kan f.eks. puttes i onCreate() i en aktivitet.

Når du bruker Kart kan du velge Meny -> My Location. Dette vil vise posisjonen til enheten, forutsatt at den kan bestemmes. Når vi bruker emulator har denne ingen mulighet til å vite dette (mangler nødvendig maskinvare), men vi kan imidlertid som utviklere selv sette ønsket posisjon. Dette kan gjøres via DDMS, hvor man f.eks. kan angi bredde- og lengdegrad via feltene latitude og longitude. Husk å trykk Send når dette er gjort for å oppdatere emulatoren.

Det er også mulig å integrere kartfunksjonalitet inn i din egen applikasjon, se [Google Maps].

10.4 Sensorer

Handholdte enheter har etterhvert fått diverse sensorer. La oss kort gå igjennom hvilke ulike typer sensorer som kan finnes. I Android opererer man med følgende hovedtyper:

- Posisjonssensorer. Dette er sensorer som angir posisjonsdata. Det kan f.eks. være GPS eller kompass (magnetisme).
- Bevegelsesensorer. Dette er sensorer som angir akselerasjon og rotasjon. Det kan f.eks. være gyroskop, akselerometer og gravitasjonsmåler.
- Miljøsensorer. Dette er sensorer som kan måle miljøvariabler som f.eks. temperatur, trykk, lysmengde, fukt osv. Barometer, termometer og fotometer er mulige typer sensorer i denne gruppen.

Du kan finne mer informasjon om bruk av slike sensorer på [Android Sensors].

10.5 Til slutt

Når er vi kommet så langt som vi rekker i dette faget. Det er selvsagt masse å lære om Android-utvikling fortsatt. Her er vel praktisering sannsynligvis den beste vei videre. Før du starter å lage applikasjoner for salg eller et bredere publikum kan det imidlertid vært lurt å lese om ”best practice” for Android apps. Du finner en del (forhåpentligvis interessant) informasjon på [Android Best Practice].

10.6 Etter slutt ;-)

Nå er vi gjennom faget gitt :-)

Vi faglærerne håper at du har fått den faglige utfordringen du har vært ute etter. Forhåpentligvis har du nå en god plattform å bygge videre på når du skal lage applikasjoner for mobile enheter. Dette er utvilsomt et marked der mye vil skje i årene framover.

Lykke til videre :-)

10.7 Referanser

[Android Best Practice] - https://developer.android.com/guide/ **Velg ”Best practicies”**

[Android LBS] - https://developer.android.com/guide/topics/location/strategies

[Android Resources] -
http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources
[Android Sensors] - http://developer.android.com/guide/topics/sensors/sensors_overview.html
[Google Maps] - https://developers.google.com/maps/documentation/android-sdk/intro