Androidskolen Sesjon 4

ctrl-s

bouvet

Agenda

- File I/O
- Shared preferences
- Praktisk oppgave # 1
- SQLite
- Praktisk oppgave # 2
- ROOM
- Praktisk oppgave # 3
- Content Providers

Dagens målsetning

En applikasjon krever stort sett alltid en eller annen form for persistens og vi tar for oss ulike måter å håndtere dette på i en Android device. Primært er målet å være kjent med funksjonaliteten som støttes i de ulike fremgangsmåtene, og å bli kjent med fordeler og ulemper med hver av dem.

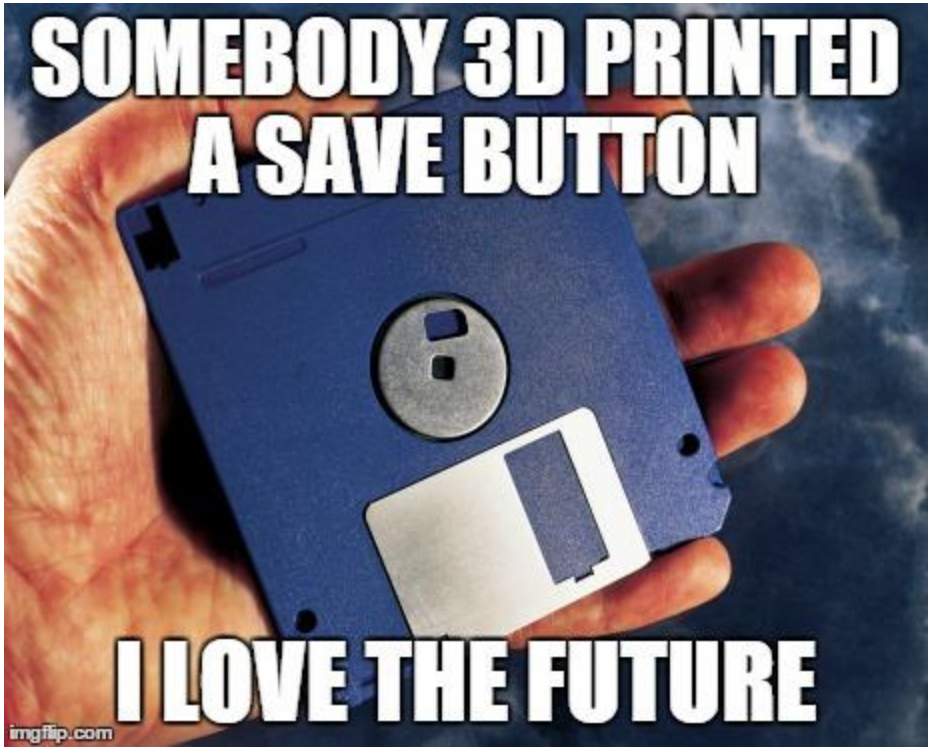
bouvet

File I/O

Bruk dine kjente og kjære java metoder for å lese, skrive osv.

Alternative lokasjoner:

- Internal storage
- Primary external storage
- Secondary external storage
- Cache



Shared preferences

- For lagring av “key-value” variabler
  - booleans
  - floats
  - ints
  - longs
  - Strings
- Aktivity preferences

```
val sharedPref = activity?.getPreferences(Context.MODE_PRIVATE)
```

- Fil preferences

```
val sharedPref = activity?.getSharedPreferences(
     getString(R.string.preference_file_key), Context.MODE_PRIVATE)
```

Shared preferences

- Skrive til shared preferences

```
with (sharedPref.edit()) {
    putInt(getString(R.string.saved_high_score_key), newHighScore)
    apply()
}
```

- Lese fra shared preferences

```
val highScore = sharedPref.getInt(KEY_HIGH_SCORE, defaultValue)
```

Praktisk oppgave #1

- Misbruk Shared preferences på det groveste for å lage en “database”
- Se SharedPreferencesContactStorage

SQLite

- Open Source DB embedded i Android
- Enkel måte å gi app'en strukturert lagring av data
- Rimelig enkel å komme i gang med.
- De viktigste klassene som benyttes er:
  - SQLiteOpenHelper
  - SQLiteDatabase
  - Cursor

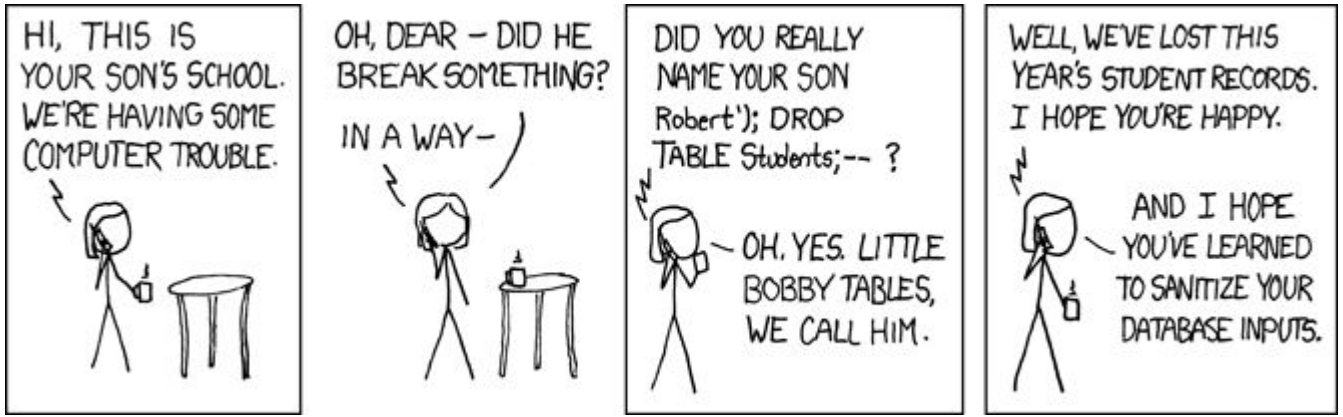
SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.

SQLite - opprette database

- Extend SqliteOpenHelper
- onCreate kalles hver gang en ny databasefil opprettes
- onUpdate kalles hver gang versjonsnummeret endres (migrering)
- Tabeller opprettes manuelt

SQLite - legge inn data

- Bruk en “writableDatabase” fra OpenHelper
- Legg til parametre i ContentValues



SQLite - uthenting av data

- Bruk en “readableDatabase” fra OpenHelper
- Støtter både parameterbasert query og ren SQL
- Gir Cursor å iterere over for å hente data

Praktisk oppgave #2

- Samme som i oppgave #1, bare nå med skikkelige verktøy
- Se SqliteContactStorage

SQLite

Caution: Although these APIs are powerful, they are fairly low-level and require a great deal of time and effort to use:

- There is no compile-time verification of raw SQL queries. As your data graph changes, you need to update the affected SQL queries manually. This process can be time consuming and error prone.
- You need to use lots of boilerplate code to convert between SQL queries and data objects.

For these reasons, we **highly recommended** using the Room Persistence Library as an abstraction layer for accessing information in your app's SQLite databases.

Room

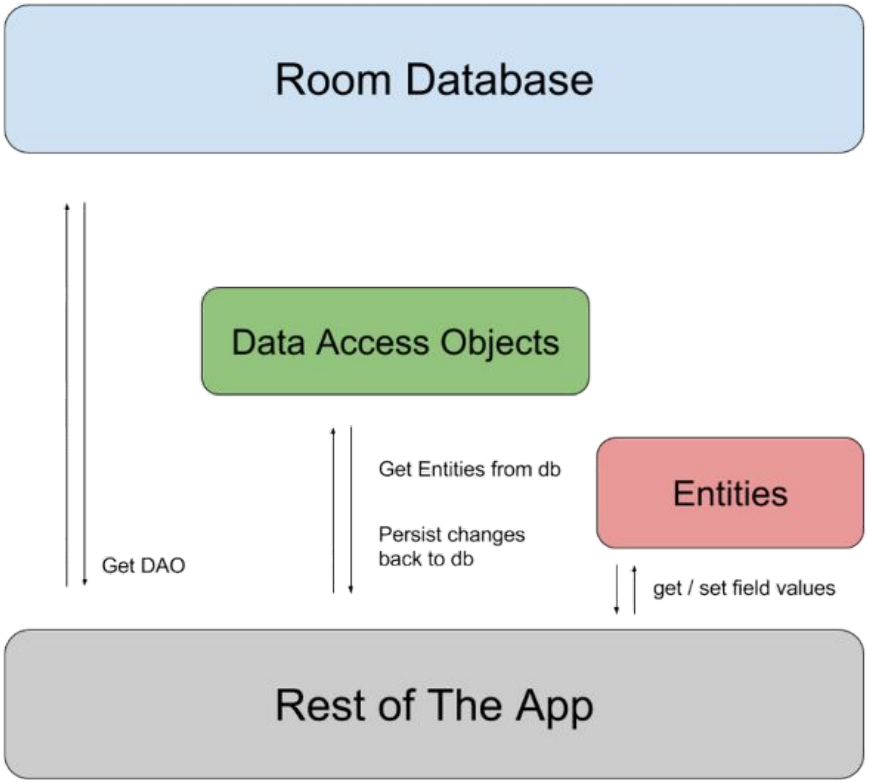
- Google sitt *persistence library* over SQLite
- Abstraksjonslag over SQLite
  - Mindre utsatt for feil
  - Raskt (vs alternativer)
- Cache / Offline av app data
- “Highly recommend” av Google

bouvet

Room

Hovedkomponenter i Room:

- Database
- Entity
  - Representerer tabellen i DB
- DAO
  - Metoder for tilgang til DB



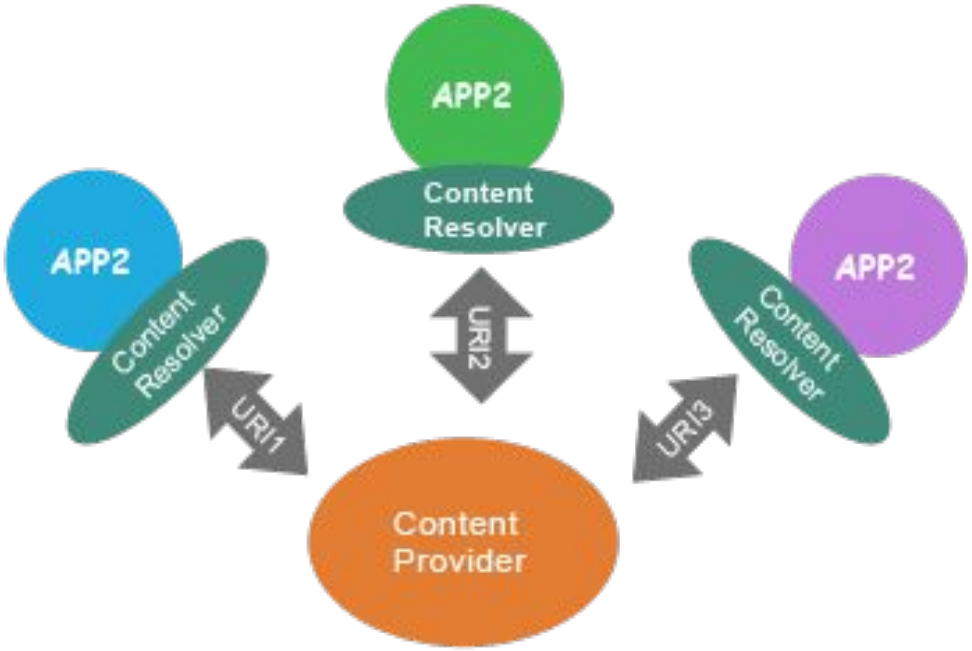
Room

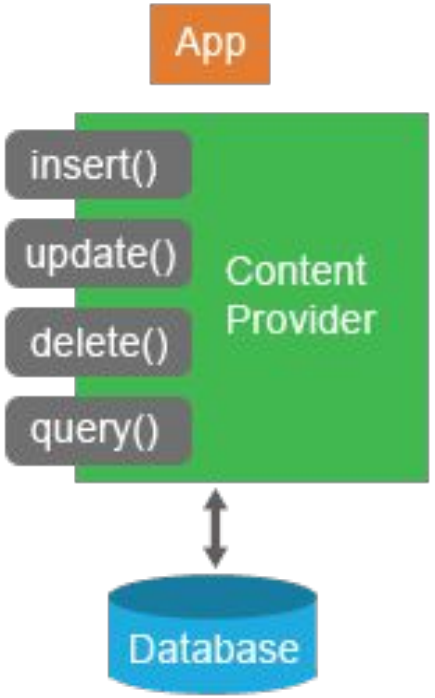
Annotations	Purpose
@Entity	Creates a SQLite table in the database using a data model class.
@Dao	Create a Data Access Object in the database using an interface class.
@Database	A class with this annotation will create an abstraction for the Data Access Object.
@PrimaryKey	A variable with this annotation will set a primary key for the table.
@Insert	Inserts parameters into the table.
@Update	Updates parameters of an existing table.
@Delete	Deletes parameters of an existing table
@Query	Running SQL query method within the table
@Ignore	Ignores the parameter form the Room database

Praktisk oppgave #3

- Samme som i oppgave #2, men nå med Room rammeverket
- Se RoomContactStorage

Content providers





Content provider - query

```
String[] EVENT_PROJECTION = new String[] {
        CalendarContract.Calendars._ID,                           // 0
        CalendarContract.Calendars.ACCOUNT_NAME,                  // 1
        CalendarContract.Calendars.CALENDAR_DISPLAY_NAME,         // 2
        CalendarContract.Calendars.OWNER_ACCOUNT                  // 3
};

// Run query
Cursor cur = null;
ContentResolver cr = getContext().getContentResolver();
Uri uri = CalendarContract.Calendars.CONTENT_URI;
String selection = "((" + CalendarContract.Calendars.ACCOUNT_NAME + " = ?) AND ("
        + CalendarContract.Calendars.ACCOUNT_TYPE + " = ?) AND ("
        + CalendarContract.Calendars.OWNER_ACCOUNT + " = ?))";


String[] selectionArgs = new String[] {"sampleuser@gmail.com", "com.google",
        "sampleuser@gmail.com"};
// Submit the query and get a Cursor object back.
cur = cr.query(uri, EVENT_PROJECTION, selection, selectionArgs, null);
```

Content provider - insert, update, delete

En rad

```
ContentValues values = new ContentValues();
values.put(ContactsContract.Data.RAW_CONTACT_ID, rawContactId);
values.put(ContactsContract.Contacts.Data.MIMETYPE, ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);
values.put(ContactsContract.CommonDataKinds.Phone.NUMBER, "1-800-GOOG-411");
values.put(ContactsContract.CommonDataKinds.Phone.TYPE, ContactsContract.CommonDataKinds.Phone.TYPE_CUSTOM);
values.put(ContactsContract.CommonDataKinds.Phone.LABEL, "free directory assistance");
Uri dataUri = getContext().getContentResolver().insert(ContactsContract.Data.CONTENT_URI, values);
```

Mange rader

```
ops.add(ContentProviderOperation.newUpdate(ContactsContract.Data.CONTENT_URI)
        .withSelection(ContactsContract.Contacts.Data._ID + "=?", new String[]{String.valueOf(dataId)})
        .withValue(ContactsContract.CommonDataKinds.Email.DATA, "somebody@android.com")
        .build());
getContext().getContentResolver().applyBatch(ContactsContract.AUTHORITY, ops);
```

bouvet

Content providers - Contact provider

- ContactsContract.Contacts

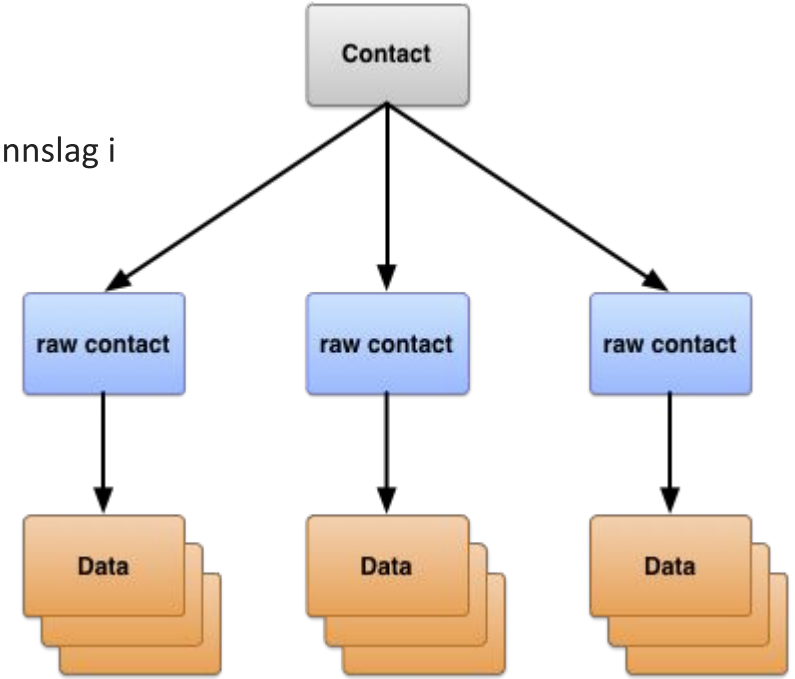
Rader representerer ulike kontakter som er basert på aggregerte innslag i rawcontacts.

- ContactsContract.RawContacts

Rader som representerer en persons data knyttet til en konto.

- ContactsContract.Data

Generisk tabell som inneholder rader for ulike data.



Praktisk oppgave #4

- Samme som i oppgave #3, men nå bruker vi Contacts-appen på enheten for lagring av våre egne Contacts
- Se ContentProviderContactStorage

Oppsummering

- Mange måter å gjøre dette på
- Vurder: Trenger appen lokal storage?
  - (Hva med SKYEN?)