GUIDE BOOK PRAKTIKUM PEMROGRAMAN MOBILE

2020

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS UDAYANA

2019

MODUL I
SHAREDPREFERENCES

`SharedPreferences` adalah mekanisme penyimpanan yang sederhana. Mekanisme ini mendukung struktur penyimpanan *key-value*. Penggunaan sharedpreferences terbilang sederhana. `Sharedpreferences` bersifat atomik. Artinya, setiap data berdiri sendiri. Terdapat 3 hal yang harus diketahui untuk mengimplementasikan `sharedpreferences` yaitu membuat, menyimpan, dan membaca data.

A. Membuat SharedPreferences

```
final String SHARED_PREFERENCES_NAME="shared_preferences";

SharedPreferences sharedPref= getSharedPreferences(SHARED_PREFERENCES_NAME,
Context.MODE_PRIVATE);
```

Membuat `sharedpreferences` pada suatu activity dapat menggunakan metode `getSharedPreferences(NAME,MODE)`. `Sharedpreferences` yang dibuat bersifat `PRIVATE`, yang artinya hanya aplikasi tersebut saja yang dapat mengaksesnya.

B. Menyimpan Data

```
final String KEY_NAMA="nama";
final String KEY_UMUR="umur";

SharedPreferences.Editor editor=sharedPref.edit();

editor.putString(KEY_NAMA,"wayan");
editor.putInt(KEY_UMUR,20);
editor.apply();
```

Penyimpanan data pada `SharedPreferences` memerlukan suatu `SharedPreferences.Editor`. Metode yang digunakan dalam penyimpanan data dalam bentuk `integer` yaitu `putInt(KEY,VALUE)`. Contoh lainnya adalah metode `putString(KEY,VALUE)` untuk menyimpan data bertipe `string`. Ini berlaku untuk tipe data lain seperti `double`, `long`, `boolean`. Untuk merekam penyimpanan yang telah terjadi, dapat dilakukan dengan menjalankan metode `commit()` seperti berikut.

```
editor.commit();
```

Metode `commit()` akan menyimpan data secara *synchronous*. Selain menggunakan metode `commit()`, metode lain yang dapat digunakan yaitu `apply()` seperti berikut.

```
editor.apply();
```

Metode `apply()` akan menyimpan data secara *asynchronous*. Metode ini merupakan pilihan yang tepat apabila ada banyak data yang hendak disimpan Metode `commit()` telah ada sejak API level 1. Sementara `apply()` hadir sejak API level 9.

C. Membaca Data

```
String nama=sharedPref.getString(KEY_NAMA,"");
int umur=sharedPref.getInt(KEY_UMUR,0);
```

Metode yang digunakan untuk membaca data dari `sharedpreferences` sesuai dengan tipe datanya. Ketika ingin membaca data `string`, maka metode yang digunakan yaitu `getString(KEY, DEFAULT_VALUE)`. Ketika ingin membaca data `integer`, metode yang digunakan yaitu `getInt(KEY, DEFAULT_VALUE)`.

MODUL II

SQLITE

SQLite merupakan *database* yang bersifat *open source* yang mendukung operasi relasi standar yang umum terdapat pada *engine database* seperti sintak SQL dan operasi transaksi. SQLite secara *default* sudah terdapat pada semua *device* android dan yang terpenting tidak diperlukan proses autentikasi ataupun setup administrasi seperti yang dilakukan pada *software database* berskala besar. SQLite hanya perlu mendefinisikan *statement* SQL untuk pembuatan dan pembaruan data.

A. Pembuatan dan Pembaruan *Database*

Langkah pertama dalam pembuatan dan pembaruan `SQLite` oleh aplikasi adalah mendifinisikan skema data yang akan diimplementasikan ke dalam *database*. Semua skema yang ada dimasukan pada suatu *file* `java` yang berisikan nama tabel, nama kolom, bahkan *query* yang dilakukan. Ini adalah *best practice* yang direkomendasikan agar memudahkan dalam mengorganisasi kelas-kelas yang digunakan.

```
public final class MahasiswaContract {
    public static class MahasiswaEntry implements BaseColumns{
        public static final String TABLE_NAME="mahasiswa";
        public static final String COLUMN_NAMA="nama";
        public static final String COLUMN_ALAMAT="alamat";

    }
}
```

Variabel nama tabel dan kolom didefiniskan secara global dan dijadikan konstanta, sehingga mempermudah untuk diakses di semua struktur bagian program. Setiap kolom yang bersifat incremental dan menjadi *primary key* seperti `ID` pada SQLite harus berbentuk "`_id`". Dengan mengimplementasikan `BaseColumns`, kolom "`_id`" akan secara otomatis menjadi bagian dari kolom pada sebuah tabel yang telah ditentukan.

Android telah menyediakan satu *set* API yang diperuntukan untuk melakukan pembuatan dan pembaruan aplikasi pada kelas `SQLiteOpenHelper`. Kelas tersebut diperuntukan untuk menjalankan fungsi-fungsi dalam *Data Definition Language* (DDL) pada sebuah *database*.

```
public class DbHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION=1;
    private static final String DATABASE_NAME="praktikum.db";
```

```
    private static final String SQL_CREATE_TABLE_MAHASISWA =
            "CREATE TABLE "+ MahasiswaEntry.TABLE_NAME +" ("+
            MahasiswaEntry._ID+ " INTEGER PRIMARY KEY AUTOINCREMENT,"+
            MahasiswaEntry.COLUMN_NAMA+" TEXT,"+
            MahasiswaEntry.COLUMN_ALAMAT+" TEXT )";

    private static final String SQL_DROP_TABLE_MAHASISWA = "DROP TABLE IF EXISTS "+
            MahasiswaEntry.TABLE_NAME;

    public DbHelper(Context context) {
        super(context, DATABASE_NAME, null,DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL(SQL_CREATE_TABLE_MAHASISWA);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int
newVersion) {
        sqLiteDatabase.execSQL(SQL_DROP_TABLE_MAHASISWA);
        onCreate(sqLiteDatabase);
    }
}
```

B. Proses *Insert* Data

```
DbHelper dbHelper=new DbHelper(this);

SQLiteDatabase db=dbHelper.getWritableDatabase();

ContentValues values=new ContentValues();
values.put(MahasiswaEntry.COLUMN_NAMA,"Wayan");
values.put(MahasiswaEntry.COLUMN_ALAMAT,"Jimbaran");

long newRowId=db.insert(MahasiswaEntry.TABLE_NAME,null,values);
```

Proses *insert* data memerlukan sebuah objek `SQLiteDatabase` yang didapat dari pemanggilan `getWritableDatabase()` dari objek `dbHelper`. Objek `dbHelper` berasal dari kelas `DbHelper` yang telah dibuat sebelumnya. Selanjutnya diperlukan sebuah objek values yang merupakan *instance* objek dari `ContentValues`. `ContentValues` adalah objek map pasangan *KEY* dan *VALUE*, dimana yang menjadi key adalah nama kolom tabel tujuan dan *value*-nya adalah nilai yang ingin dimasukan. Selanjutnya yaitu melakukan proses *insert* menggunakan metode `insert()`.

```
long newRowId=db.insert(MahasiswaEntry.TABLE_NAME,null,values);
```

Terdapat tiga parameter utama dalam metode `insert()` yaitu nama tabel, `nullColumnHack`, dan objek *map* `ContentValues`. Parameter `nullColumnHack` adalah mekanisme dalam meng-*input*-kan objek `ContentValues` berisi `null` pada kolom tertentu.

Sebagai contoh akan dimasukan nilai `null` pada kolom nama, dan kolom alamat di tabel mahasiswa.

```
ContentValues values=new ContentValues();
long newRowId=db.insert(MahasiswaEntry.TABLE_NAME,null,values);
```

Kode Program di atas tidak akan menimbulkan penambahan data pada tabel, dikarenakan `ContentValues` tidak berisikan sebuah data, dan `nullColumnHack` bernilai null. Sehingga program akan meng-*generate* sintak SQL "`INSERT INTO mahasiswa`" yang tidak menimbulkan penambahan data. Agar penambahan data terjadi maka `nullColumnHack` harus didefinisikan.

```
ContentValues values=new ContentValues();

long newRowId=db.insert(MahasiswaEntry.TABLE_NAME, MahasiswaEntry.COLUMN_NAMA,
values);
```

Kode Program di atas akan menggenerate sintak SQL "`INSERT INTO mahasiswa (nama) values (null)`". Sintak ini akan menyebabkan penambahan data dengan `_id` *autoincrement*, nama bernilai `null`, dan alamat bernilai `null`.

C. Proses Update Data

```
DbHelper dbHelper=new DbHelper(this);

SQLiteDatabase db=dbHelper.getWritableDatabase();

ContentValues values=new ContentValues();
values.put(MahasiswaEntry.COLUMN_ALAMAT,"JL. Kampus Unud");

String whereClause= MahasiswaEntry.COLUMN_NAMA+" = ?";

String[] whereArgs={"Wayan"};

int count=db.update(MahasiswaEntry.TABLE_NAME,values,whereClause,whereArgs);
```

SQLite telah menyediakan metode `update()` untuk memperbarui data pada suatu tabel. Terdapat empat parameter utama dalam metode `update()` yaitu nama tabel, `ContentValues`, `whereClause`, dan `whereArgs`. Parameter `whereClause` dan `whereArgs` akan mendefinisikan kondisi ketika memperbarui data. Sintak SQL yang akan di-*generate* yaitu "`UPDATE mahasiswa set alamat='JL. Kampus Unud' where nama='Wayan'`".

D. Proses Delete Data

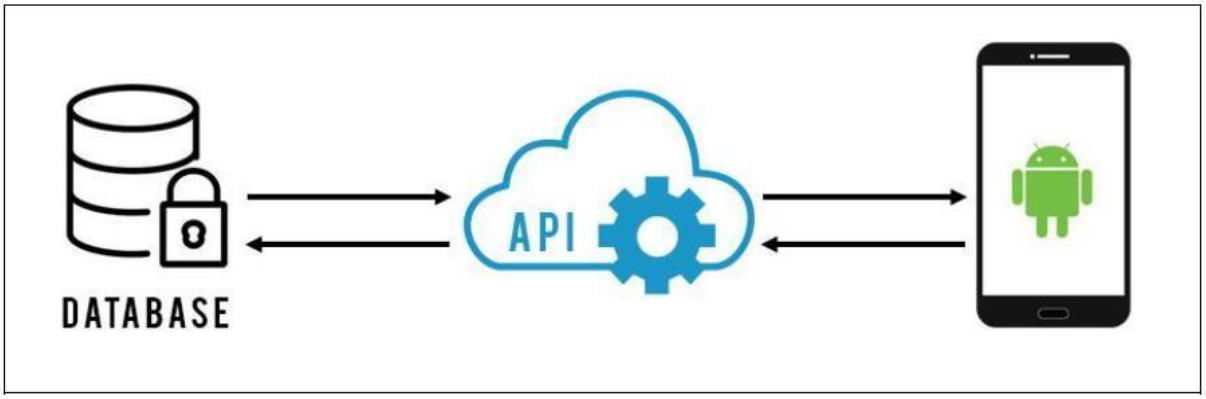
```
DbHelper dbHelper=new DbHelper(this);

SQLiteDatabase db=dbHelper.getWritableDatabase();
```

```
ContentValues values=new ContentValues();
values.put(MahasiswaEntry.COLUMN_ALAMAT,"JL. Kampus Unud");

String whereClause= MahasiswaEntry.COLUMN_NAMA+" = ?";

String[] whereArgs={"Wayan"};

int count=db.delete(MahasiswaEntry.TABLE_NAME,whereClause,whereArgs);
```

SQLite telah menyediakan metode `delete()` untuk menghapus data pada suatu tabel. Terdapat tiga parameter utama dalam metode `delete()` yaitu nama tabel, `whereClause`, dan `whereArgs`. Paramete `whereClause` dan `whereArgs` akan mendefinisikan kondisi ketika menghapus data. Sintak SQL yang akan di-generate yaitu "`DELETE FROM mahasiswa WHERE nama='Wayan'`".

MODUL III
API

API merupakan singkatan dari *Application Programming Interface*, dan memungkinkan *developer* untuk mengintegrasikan dua bagian dari aplikasi atau dengan aplikasi yang berbeda secara bersamaan. Tujuan dari penggunaan API adalah untuk mempercepat proses *development* dengan menyediakan function secara terpisah sehingga *developer* tidak perlu membuat fitur yang serupa.



Aplikasi android berkomunikasi ke *database server* melalui sebuah API. Melakukan *request* data ke sebuah API bisa dibilang sulit apabila tidak menggunakan *library* pihak ketiga. Library yang umum digunakan yaitu Volley atau Retrofit.

MODUL IV
FIREBASE CLOUD MESSAGING

FCM atau Firebase Cloud Messaging merupakan sebuah layanan dari firebase yang memungkinkan *developer* mengelola pengiriman notifikasi dari *server* ke aplikasi kliennya. Firebase sendiri adalah sebuah layanan *cloud backend* milik Google yang mempermudah *developer* dalam mengembangkan sebuah aplikasi. Firebase memiliki berbagai fitur yang sangat menarik, salah satunya Firebase Cloud Messaging.

FCM merupakan transformasi dari GCM (Google Cloud Messanging) dengan berbagai pembaruan yang lebih *powerfull*. FCM memiliki 2 komponen utama untuk mengirim dan menerima pesan.

1. Firebase Cloud Function atau *server* aplikasi untuk menyiapkan, menargetkan, dan mengirim pesan.
2. Sebuah aplikasi *client* (Android, IOS, atau Web) untuk menerima pesan.

Notifikasi dapat dikirimkan melalui Admin SDK atau melalui HTTP & XMPP APIs. Selain itu notifikasi juga dapat dikirimkan melalui *Notification composer* pada *Firebase console* dengan fitur target yang lebih lengkap serta fitur *analytics* yang bisa digunakan untuk memantau status dari pesan yang dikirimkan.

SOAL PRAKTIKUM

Buatlah suatu aplikasi android dengan fitur sebagai berikut:

1. Login (API)
2. Register (API)
3. CRUD Master Data (API)
   - Show Data
   - Show Detail Data
4. Menyimpan data dari API ke SQLite sehingga ketika koneksi terputus akan ditampilkan data dari SQLite.
5. Profil Pengguna
   - Edit Profil
6. Notifikasi melalui Firebase Console.

NB:

Bicarakan terlebih dahulu projek yang akan dibuat kepada asisten praktikum masing-masing ☺

SELAMAT MENGERJAKAN

HAL TERPENTING ADALAH MENGERTI DAN MEMAHAMI BUKAN MENYONTEK UNTUK SEKADAR SELESAI ☺

DAFTAR PUSTAKA

Imaduddin Ahmad & Permana Sidiq, 2018, Menjadi Android Developer Expert, PT. Presentologics, Bandung.

STRUKTUR LAPORAN

Penulisan laporan menggunakan format TA terlampir dan manfaatkan waktu dengan sebaik-baiknya.