CRUD SQLite Menggunakan Room Database di Android

Room atau Room Persistence Library adalah sebuah library yang berfungsi sebagai abstraction layer di atas SQLite database yang memungkinkan kita menggunakan SQLite dengan lebih mudah, efektif dan efisien.

Kita sudah belajar tentang CRUD menggunakan SQLite di pertemuan sebelumnya. Sekarang kita akan menyederhanakan proses CRUD pada SQLite dengan menggunakan Room library ini. Karena sebenarnya kita akan menggunakan SQLite database juga, namun dengan menggunakan Room akan berbeda cara pengaksesannya.

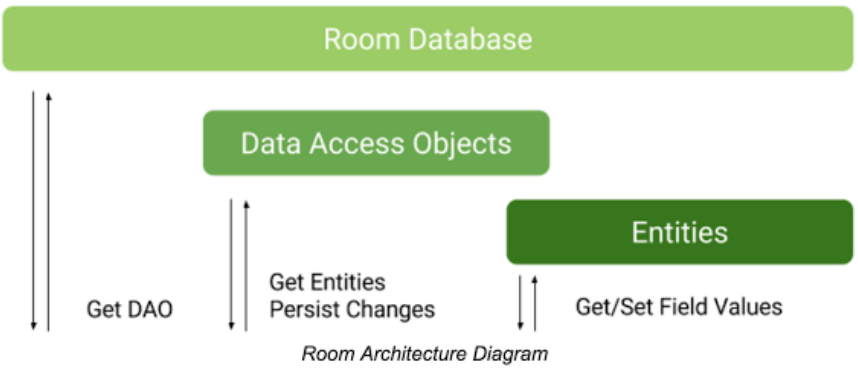
Perbedaan mendasar antara Room dan SQLite adalah :

- **sqlite**—Menggunakan SQLiteOpenHelper dan interface SQLite standar (DBHelper, dsb).
- **room**—Full abstraction menggunakan Room library dan akses yang lebih efisien.

Pada prinsipnya Room dibagi menjadi tiga buah komponen utama, yaitu Database, Entity, dan DAO :

- **Entity** merepresentasikan data pada sebuah tabel seperti di database pada umumnya, dibuat menggunakan annotation pada java data object. Setiap entity mempunya satu tabel sendiri.
- **DAO** (Data Access Object) menggambarkan method-method yang mengakses database, termasuk method standar seperti CRUD (Create, Read, Update, Delete). Menggunakan annotation untuk mengikat SQL query ke suatu method.
- **Database** adalah holder class yang menggunakan annotation untuk menampilkan daftar dari entity-entity yang ada dan juga database version. Kelas ini juga berisi daftar dari DAO yang ada.

Penggambarannya bisa dilihat pada diagram Room Database di bawah ini :



Room Architecture Diagram

Room ini termasuk ke dalam komponen library Architecture Android.

Membuat Aplikasi CRUD Menggunakan Room Database dan Android Studio

Sekarang kita akan mencoba membuat aplikasi inventaris barang sederhana yang berfungsi untuk mencatat barang apa saja yang disimpan. Aplikasi ini berguna untuk mendemokan fungsi CRUD (Create, Read, Update, dan Delete) pada Room Database.

Import library Room pada project baru kalian dengan cara, pada file *build.gradle* project, import google() dependency pada repository di bagian buildscript dan juga allprojects.

```
buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.0.1'
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}
```

Setelah itu pada module-level build.gradle, import Room library pada bagian dependencies seperti di bawah ini

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "android.arch.persistence.room:runtime:1.0.0"
    annotationProcessor "android.arch.persistence.room:compiler:1.0.0"
```

Kemudian, lakukan *sync gradle*. Jika sudah selesai maka library Room sudah terpasang di project aplikasi Android di Android Studio.

Entity

Entity adalah representasi class Java dari suatu object di dunia nyata. Pada aplikasi CRUD yang akan dibuat, entity-nya adalah barang (karena kita akan membuat aplikasi inventaris barang). Object barang tersebut akan mempunyai attribut berupa :

- **id_barang** *(primary key)*
- **nama_barang**
- **merk_barang**

- **harga_barang**

Atribut-atribut itulah yang akan direpresentasikan ke dalam object Barang, yang akan digunakan sebagai entity pada Room database di aplikasi yang akan dibuat.

Membuat Entity, DAO dan Database Menggunakan Room di Android Studio

Setelah sebelumnya kita sudah belajar tentang bagaimana cara mengimport Room library di Android Studio project. Maka kali ini kita akan belajar membuat Entity, DAO dan Database menggunakan Room Persistence Library.

Entity, DAO dan Database pada Room

Walaupun pada materi sebelumnya sudah dijelaskan, Kita akan mengulang kembali definisi dari Entity, DAO dan Database pada Room.

- **Entity** adalah ibaratnya sebuah tabel yang merepresentasikan object di dunia nyata. Pada CRUD ini kita akan memakai entity berupa sebuah *barang*.
- **DAO** / Database Access Object, adalah sebuah kelas yang berisi methods yang digunakan untuk mengakses database.
- **Database** adalah class yang berisi daftar DAO yang bisa dipakai, kelas ini juga berisi versi dari database yang kita gunakan pada aplikasi.

Membuat Entity pada Room Database Android

Selanjutnya buka project Android Studio baru. Pertama-tama kita akan membuat class Entity-nya terlebih dahulu. Karena pada aplikasi CRUD inventaris sederhana ini object yang ingin kita simpan adalah barang, maka kita akan membuat tabel dari object barang tersebut. Dengan atribut-atributnya antara lain, id barang (primary key), nama barang, merk barang dan harga barang.

Buat class Barang.java sebagai entity, dan isikan kode berikut :

Barang.java

```
package co.twoh.roomtutorial.model;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.PrimaryKey;

import java.io.Serializable;

@Entity(tableName = "tbarang")
public class Barang implements Serializable{
```

```
    @PrimaryKey(autoGenerate = true)
    public int barangId;

    @ColumnInfo(name = "nama_barang")
    public String namaBarang;

    @ColumnInfo(name = "merk_barang")
    public String merkBarang;

    @ColumnInfo(name = "harga_barang")
    public String hargaBarang;

    public int getBarangId() {
        return barangId;
    }

    public void setBarangId(int barangId) {
        this.barangId = barangId;
    }

    public String getNamaBarang() {
        return namaBarang;
    }

    public String getMerkBarang() {
        return merkBarang;
    }

    public void setNamaBarang(String namaBarang) {
        this.namaBarang = namaBarang;
    }

    public void setMerkBarang(String merkBarang) {
        this.merkBarang = merkBarang;
    }

    public String getHargaBarang() {
        return hargaBarang;
    }

    public void setHargaBarang(String hargaBarang) {
        this.hargaBarang = hargaBarang;
    }
}
```

Bisa dilihat pada kode program di atas class Entity hampir sama bentuknya dengan object Model Java pada umumnya. Class Barang di atas mempunyai atribut, dan juga mempunyai getter dan setter. Bedanya pada class di atas, terdapat annotation @*Entity* yang diikuti dengan nama tabel. Pada dasarnya class Entity ini merepresentasikan tabel barang pada database yang akan digunakan untuk menyimpan data barang.
Kemudian atribut-atribut pada entity juga bisa di-define kegunaannya, layaknya mendefine nama kolom pada sebuah tabel di database. Atribut barangId di atas kita define sebagai *primary key* dan atribut namaBarang, merkBarang, dan hargaBarang kita define sebagai nama kolom biasa.

Membuat Data Access Object (DAO) pada Room Database Android

Setelah itu, kita akan membuat interface DAO nya atau Data Access Object. Interface ini berisi daftar method-method yang akan kita pakai untuk mengakses tabel barang di database seperti method insert, read, update, delete, select dan semacamnya.

Buatlah sebuah file .java bernama BarangDAO dan masukkan kode seperti di bawah ini :

BarangDAO.java

```
package co.twoh.roomtutorial.data;

import android.arch.persistence.room.Dao;

import co.twoh.roomtutorial.model.Barang;

@Dao
public interface BarangDAO {

// stay tune di next tutorial :D
}
```

Bisa dilihat pada kode program di atas, interface DAO bentuknya seperti interface Java biasa hanya bedanya kita tambahkan annotation *@Dao* yang merupakan bawaan dari Room Database Library. Untuk isinya akan kita kosongkan terlebih dahulu dan akan dibahas pada tutorial selanjutnya.

Jika kita sudah membuat DAO nya. Maka terakhir kita akan membuat class Database Room itu sendiri.

Membuat Abstract Class Database pada Room Android

Class ini semacam gateway satu pintu yang merepresentasikan object *Room Database* itu sendiri dan berisi daftar dari DAO yang bisa kita akses. Sifatnya adalah abstract, dimana class ini hanya berisi header dari suatu method dan tidak ada body method nya. Kodingannya kira-kira seperti di bawah ini, dengan nama kelas nya adalah *AppDatabase.java* :

```
package co.twoh.roomtutorial.data.factory;

import android.arch.persistence.room.Database;
import android.arch.persistence.room.RoomDatabase;

import co.twoh.roomtutorial.data.BarangDAO;
import co.twoh.roomtutorial.model.Barang;

@Database(entities = {Barang.class}, version = 1)
```

```
public abstract class AppDatabase extends RoomDatabase {
    public abstract BarangDAO barangDAO();
}
```

Di situ kita bisa lihat abstract class AppDatabase mempunyai annotation *@Database* yang juga bawaan dari Room Database itu sendiri. Class AppDatabase meng-extends class *RoomDatabase*, dan pada annotation *@Database* di atas kita bisa mendefinisikan database versionnya dan juga entity-entity yang akan kita pakai. Karena entity di app ini hanya satu (Barang), maka kita define *entities = {Barang.class}* seperti di atas.

Isi dari class AppDatabase adalah method untuk mengakses interface *BarangDAO()* yang sudah kita buat sebelumnya.

Mengakses Room Database

Jika kita sudah membuat tiga komponen utama Room Database library, maka sekarang untuk cara pengaksesannya. Caranya sederhana, seperti contoh di bawah :

```
private AppDatabase db;

db = Room.databaseBuilder(getApplicationContext(),
        AppDatabase.class, "barangdb").build();

db.barangDAO().insertBarang(barang);
```

Kita hanya perlu meng-instantiate object AppDatabase dengan cara memanggil Room.databaseBuilder. Di sana kita akan memasukkan "barangdb" sebagai nama file database nya. Nama itu sifatnya constant sebagai identitas database, sehingga harus selalu menggunakan nama yang sama setiap kali kita ingin memanggil database tersebut.

Setelah kita mendapatkan object db, kita akan dapat akses ke *barangDAO()* dan dari situ kita tinggal memilih operasi apa saja yang ingin kita lakukan di database, seperti contoh di atas kita memanggil method *insertBarang()* pada *barangDAO()*.

CRUD Room Database di Android : Membuat Fungsi Insert Data

Kali ini kita akan belajar untuk mengimplementasikan fitur Create/Insert data menggunakan Room Database library dimana materi ini berhubungan dengan tutorial-tutorial tentang Room database library sebelumnya.

Perlu diingat, Room library pada dasarnya hanyalah sebuah abstraction / interface yang memudahkan kita dalam menggunakan database di Android, layer bagian dalam dari Room database library adalah database SQLite seperti biasa. Karena itu ada baiknya jika kalian memahami syntax SQL dan konsep dasar SQLite di Android.

Membuat Fitur Insert Data Menggunakan Room Database Library

Untuk membuat fitur create/insert data, yang kita butuhkan adalah sebuah form untuk menginputkan data-data barang berupa *nama barang*, *merk barang*, dan *harga barang*. Berikut kode XML-nya

File activity_create.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:background="@drawable/bg"
    android:orientation="vertical"
    android:padding="10dp">

    <android.support.v7.widget.CardView
xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:id="@+id/cv_main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_margin="4dp"
        android:padding="10dp"
        card_view:cardBackgroundColor="@color/background_material_light"
        card_view:cardCornerRadius="3dp"
        card_view:cardElevation="2.5dp">

        <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="center_horizontal"
            android:gravity="center_horizontal"
            android:orientation="vertical"
```

```
            android:padding="10dp">

            <TextView

android:textAppearance="@android:style/TextAppearance.Material.Medium"
                android:layout_margin="5dp"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Data"/>

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/et_namabarang"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Nama Barang"
                    android:inputType="text" />
            </android.support.design.widget.TextInputLayout>

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/et_merkbarang"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Merk Barang"
                    android:inputType="text" />
            </android.support.design.widget.TextInputLayout>

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/et_hargabarang"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="Harga Barang"
                    android:inputType="number" />
            </android.support.design.widget.TextInputLayout>

            <Button
                android:id="@+id/bt_submit"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Submit" />
        </LinearLayout>
    </android.support.v7.widget.CardView>
</LinearLayout>
```

Setelah XML untuk create data dibuat. Kita akan menambahkan method untuk insert data pada interface BarangDAO yang berfungsi sebagai *Data Access Object*. Di interface *BarangDAO.java* ini kita akan menempatkan method-method yang bisa kita gunakan untuk mengakses database secara langsung, termasuk method insert Barang.

Buka file *BarangDAO.java* dan masukkan kode untuk insert data seperti di bawah ini :

```
package co.twoh.roomtutorial.data;

import android.arch.persistence.room.Dao;
import android.arch.persistence.room.Insert;
import android.arch.persistence.room.OnConflictStrategy;

import co.twoh.roomtutorial.model.Barang;

@Dao
public interface BarangDAO {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    long insertBarang(Barang barang);

}
```

Bisa dilihat di kodingan di atas, kita menggunakan annotation @*Insert* untuk melakukan insert data, annotation tersebut sudah bawaan dari Room database library, sehingga kita tidak perlu menuliskan raw SQL query untuk create data. Setelah itu ada parameter *onConflict* yang akan menandakan strategy apa yang akan dipilih saat terjadi conflict pada proses insert data, dalam kasus ini kita menggunakan strategy *REPLACE* yang berarti jika ada konflik / data yang ingin kita masukkan di database sudah exist. Maka data yang lama / existing akan *di-replace* dengan data yang baru kita insert.

Untuk penggunaannya, pada Activity create data kalian. Modifikasi kodingan kalian menjadi logic flow seperti di bawah ini :

```
public class RoomCreateActivity extends AppCompatActivity {

    private AppDatabase db;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create);

        // initiate pemanggilan Room database
        db = Room.databaseBuilder(getApplicationContext(),
                AppDatabase.class, "barangdb").build();

        final EditText etNamaBarang   = findViewById(R.id.et_namabarang);
        final EditText etMerkBarang   = findViewById(R.id.et_merkbarang);
        final EditText etHargaBarang  = findViewById(R.id.et_hargabarang);
        Button btSubmit         = findViewById(R.id.bt_submit);

        btSubmit.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Barang b = new Barang();
                    b.setHargaBarang(etHargaBarang.getText().toString());
                    b.setMerkBarang(etMerkBarang.getText().toString());
                    b.setNamaBarang(etNamaBarang.getText().toString());
```

```
                    insertData(b);
                }
        });
    }

    private void insertData(final Barang barang){

        new AsyncTask<Void, Void, Long>(){
            @Override
            protected Long doInBackground(Void... voids) {
                // melakukan proses insert data
                long status = db.barangDAO().insertBarang(barang);
                return status;
            }

            @Override
            protected void onPostExecute(Long status) {
                Toast.makeText(RoomCreateActivity.this, "status row
"+status, Toast.LENGTH_SHORT).show();
            }
        }.execute();
    }

}
```

Jika kalian lihat, sebagian besar kodingan di atas adalah untuk menyambungkan antara form XML dengan Activity .java nya. Kita menerima inputan nama barang, merk dan harga dari form, kemudian memasukkannya ke dalam object Entity barang yang sudah kita buat di tutorial sebelumnya.

Setelah itu untuk proses insert data barangnya sendiri, kita hanya perlu memanggil BarangDAO dari *AppDatabase* dan melakukan eksekusi fungsi *insertData(Barang)* yang baru saja kita buat di atas :

```
db.barangDAO().insertBarang(barang);
```

Sebagai catatan, karena fungsi insert data pada Room library adalah fungsi *async* yang tidak bisa dieksekusi pada *main thread* maka kita menggunakan *AsyncTask* untuk menjalankan fungsi tersebut. Cara lebih lanjutnya bisa menggunakan *RxJava* tapi itu akan bahas dilain waktu.

CRUD Room Database di Android : Membuat Fungsi Read Data

Membuat Fitur Read All Data Menggunakan Room Database Library

Membuat XML layout

Untuk membuat fitur read all data menggunakan Room library, yang kalian butuhkan adalah sebuah Activity baru yang berisi RecyclerView untuk menampilkan semua data yang ada di dalam SQLite database. File XML untuk Activity read all ini bisa dilihat dibawah ini.

activity_read.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:background="@drawable/bg"
    android:layout_height="match_parent">
    <TextView
        android:background="@color/colorAccent"
        android:padding="5dp"

android:textAppearance="@android:style/TextAppearance.Material.Medium"
        android:layout_margin="5dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Daftar Inventaris"/>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/rv_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical" />

</LinearLayout>
```

Untuk item RecyclerView nya, kita hanya akan menampilkan nama barang, file XML layout untuk item Recycler View bisa dilihat di sini.

item_barang.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:orientation="vertical"
    android:padding="10dip">

    <android.support.v7.widget.CardView
xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:id="@+id/cv_main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_margin="4dp"
        android:padding="10dp"
        android:layout_gravity="center_horizontal"
        card_view:cardBackgroundColor="@color/background_material_light"
        card_view:cardCornerRadius="3dp"
        card_view:cardElevation="2.5dp">

        <TextView
            android:layout_margin="8dp"
            android:id="@+id/tv_namabarang"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Example application"
            android:textColor="@android:color/black"
            android:textSize="16sp" />
    </android.support.v7.widget.CardView>
</LinearLayout>
```

Membuat method read all di Data Access Object

Setelah kalian menambahkan dua file XML layout tersebut ke project Android Studio kalian, selanjutnya kita akan menambahkan method untuk read all data pada *BarangDAO.java*. Pada interface Data Access Object inilah kita akan menambahkan method-method yang digunakan untuk mengakses database secara langsung, termasuk method read all data.

Copy-pastekan kode di bawah ini ke file *BarangDAO.java* :

```
package co.twoh.roomtutorial.data;

import android.arch.persistence.room.Dao;
import android.arch.persistence.room.Insert;
import android.arch.persistence.room.Query;

import co.twoh.roomtutorial.model.Barang;

@Dao
public interface BarangDAO {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    long insertBarang(Barang barang);

    @Query("SELECT * FROM tbarang")
    Barang[] selectAllBarangs();
}
```

Bisa dilihat pada kodingan di atas, kita menggunakan annotation *@Query* untuk melakukan read all data. Berbeda dengan annotation *@Insert* yang kita gunakan untuk add/tambah data. Pada annotation *@Query* kita bisa memasukkan query SQLite yang kita inginkan di situ. Karena method nya digunakan untuk mengambil semua data barang, maka kita tulis saja query SQL *select all* standard. Ini adalah salah satu kemudahan yang ditawarkan oleh Room database library.

Membuat RecyclerView Adapter

Kemudian, kita akan membuat RecyclerView adapter supaya data yang kita ambil dari database bisa dimapping dan ditampilkan pada aplikasi. Untuk source code :

AdapterBarangRecyclerView.java

```
package co.twoh.roomtutorial.adapter;

import android.app.Activity;
import android.app.Dialog;
import android.arch.persistence.room.Room;
import android.content.Context;
import android.support.v7.widget.CardView;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import java.util.ArrayList;

import co.twoh.roomtutorial.R;
import co.twoh.roomtutorial.RoomCreateActivity;
import co.twoh.roomtutorial.RoomReadSingleActivity;
import co.twoh.roomtutorial.data.factory.AppDatabase;
import co.twoh.roomtutorial.model.Barang;

public class AdapterBarangRecyclerView extends
RecyclerView.Adapter<AdapterBarangRecyclerView.ViewHolder> {

    private ArrayList<Barang> daftarBarang;
    private Context context;
    private AppDatabase db;

    public AdapterBarangRecyclerView(ArrayList<Barang> barangs, Context
ctx){
        /**
         * Inisiasi data dan variabel yang akan digunakan
         */
        daftarBarang = barangs;
        context = ctx;

        db = Room.databaseBuilder(context.getApplicationContext(),
                AppDatabase.class,
"barangdb").allowMainThreadQueries().build();
    }

    class ViewHolder extends RecyclerView.ViewHolder {

        /**
         * Inisiasi View
         * Di tutorial ini kita hanya menggunakan data String untuk tiap
item
         * dan juga view nya hanyalah satu TextView
         */
        TextView tvTitle;
        CardView cvMain;
```

```
        ViewHolder(View v) {
            super(v);
            tvTitle = v.findViewById(R.id.tv_namabarang);
            cvMain = v.findViewById(R.id.cv_main);
        }
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        /**
         *  Inisiasi ViewHolder
         */
        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_barang,
parent, false);
        // mengeset ukuran view, margin, padding, dan parameter layout
lainnya
        ViewHolder vh = new ViewHolder(v);
        return vh;
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, final int position) {
        /**
         *  Menampilkan data pada view
         */
        final String name = daftarBarang.get(position).getNamaBarang();

        holder.cvMain.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                /**
                 *  Kodingan untuk tutorial Selanjutnya Read detail data
                 */
                Barang barang =
db.barangDAO().selectBarangDetail(daftarBarang.get(position).getBarangId())
;

context.startActivity(RoomReadSingleActivity.getActIntent((Activity)
context).putExtra("data", barang));
            }
        });
        holder.cvMain.setOnLongClickListener(new View.OnLongClickListener()
{
            @Override
            public boolean onLongClick(View view) {
                /**
                 *  Kodingan untuk tutorial Selanjutnya :p Delete dan
update data
                 */
                final Dialog dialog = new Dialog(context);
                dialog.setContentView(R.layout.view_dialog);
                dialog.setTitle("Pilih Aksi");
                dialog.show();

                Button editButton = dialog.findViewById(R.id.bt_edit_data);
                Button delButton =
dialog.findViewById(R.id.bt_delete_data);

                //apabila tombol edit diklik
```

```
                editButton.setOnClickListener(
                        new View.OnClickListener() {
                            @Override
                            public void onClick(View view) {
                                dialog.dismiss();
                                onEditBarang(position);
                            }
                        }
                );

                //apabila tombol delete diklik
                delButton.setOnClickListener(
                        new View.OnClickListener() {
                            @Override
                            public void onClick(View view) {
                                dialog.dismiss();
                                onDeteleBarang(position);
                            }
                        }
                );
                return true;
            }
        });
        holder.tvTitle.setText(name);
    }

    private void onDeteleBarang(int position){
        db.barangDAO().deleteBarang(daftarBarang.get(position));
        daftarBarang.remove(position);
        notifyItemRemoved(position);
        notifyItemRangeRemoved(position, daftarBarang.size());
    }

    private void onEditBarang(int position){
        context.startActivity(RoomCreateActivity.getActIntent((Activity)
context).putExtra("data", daftarBarang.get(position)));
    }

    @Override
    public int getItemCount() {
        /**
         * Mengembalikan jumlah item pada barang
         */
        return daftarBarang.size();
    }
}
```

Membuat Activity Read All Data

Jika sudah, maka terakhir kita akan membuat activity read all data, dimana pada activity inilah kita akan mengakses Data Access Object (*BarangDAO*), mengambil semua data barang, dan setelah itu ditampilkan pada aplikasi.

Isi kode *RoomReadActivity.java* seperti di bawah :

```
package co.twoh.roomtutorial;

import android.app.Activity;
```

```
import android.arch.persistence.room.Room;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;

import java.util.ArrayList;
import java.util.Arrays;

import co.twoh.roomtutorial.adapter.AdapterBarangRecyclerView;
import co.twoh.roomtutorial.data.factory.AppDatabase;
import co.twoh.roomtutorial.model.Barang;

public class RoomReadActivity extends AppCompatActivity {

    private AppDatabase db;
    private RecyclerView rvView;
    private RecyclerView.Adapter adapter;
    private RecyclerView.LayoutManager layoutManager;
    private ArrayList<Barang> daftarBarang;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {

        /**
         * Initialize layout dan sebagainya
         */
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_read);

        /**
         * Initialize ArrayList untuk data barang
         */
        daftarBarang = new ArrayList<>();

        /**
         * Initialize database
         * allow main thread queries
         */
        db = Room.databaseBuilder(getApplicationContext(),
                AppDatabase.class,
"barangdb").allowMainThreadQueries().build();

        /**
         * Initialize recyclerview dan layout manager
         */
        rvView = findViewById(R.id.rv_main);
        rvView.setHasFixedSize(true);
        layoutManager = new LinearLayoutManager(this);
        rvView.setLayoutManager(layoutManager);

        /**
         * Add all data to arraylist
         */
        daftarBarang.addAll(Arrays.asList(db.barangDAO().selectAllBarangs()
));
```

```
        /**
         * Set all data ke adapter, dan menampilkannya
         */
        adapter = new AdapterBarangRecyclerView(daftarBarang, this);
        rvView.setAdapter(adapter);
    }

    public static Intent getActIntent(Activity activity) {
        // kode untuk pengambilan Intent
        return new Intent(activity, RoomReadActivity.class);
    }
}
```

Pada source code di atas bisa kita lihat bagaimana cara mengakses BarangDAO, sebenarnya sama caranya dengan di tutorial insert data sebelumnya. Perlu diketahui di sini kita menggunakan method *allowMainThreadQueries()* supaya kita bisa menjalankan method read all data di *main thread*. Namun cara ini tidak disarankan, karena kita sebenarnya harus menjalankannya di background thread menggunakan AsyncTask.

CRUD Room Database di Android : Membuat Fungsi Edit Data

Membuat Edit Data Interface

Untuk membuat fungsi Edit Data, pertama kali kita harus membuat interface-nya terlebih dahulu. Pada materi ini interface-nya menggunakan Activity yang sama seperti pada saat kita melakukan Insert Data, yaitu *RoomCreateActivity.java*. Kita hanya perlu melakukan modifikasi pada class tersebut untuk bisa menerima data dan melakukan fungsi Edit, full source code dari class tersebut bisa dilihat di sini.

RoomCreateActivity.java

```
package co.twoh.roomtutorial;

import android.app.Activity;
import android.arch.persistence.room.Room;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import co.twoh.roomtutorial.data.factory.AppDatabase;
import co.twoh.roomtutorial.model.Barang;

public class RoomCreateActivity extends AppCompatActivity {

    private AppDatabase db;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create);

        db = Room.databaseBuilder(getApplicationContext(),
                AppDatabase.class, "barangdb").build();

        final EditText etNamaBarang   = findViewById(R.id.et_namabarang);
        final EditText etMerkBarang   = findViewById(R.id.et_merkbarang);
        final EditText etHargaBarang  = findViewById(R.id.et_hargabarang);
        Button btSubmit         = findViewById(R.id.bt_submit);

        final Barang barang = (Barang)
getIntent().getSerializableExtra("data");

        if(barang!=null){
            etNamaBarang.setText(barang.getNamaBarang());
```

```
            etMerkBarang.setText(barang.getMerkBarang());
            etHargaBarang.setText(barang.getHargaBarang());
            btSubmit.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {

barang.setNamaBarang(etNamaBarang.getText().toString());

barang.setMerkBarang(etMerkBarang.getText().toString());

barang.setHargaBarang(etHargaBarang.getText().toString());

                    updateBarang(barang);
                }
            });
        }else{
            btSubmit.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Barang b = new Barang();
                    b.setHargaBarang(etHargaBarang.getText().toString());
                    b.setMerkBarang(etMerkBarang.getText().toString());
                    b.setNamaBarang(etNamaBarang.getText().toString());
                    insertData(b);
                }
            });
        }
    }

    private void updateBarang(final Barang barang){
        new AsyncTask<Void, Void, Long>(){
            @Override
            protected Long doInBackground(Void... voids) {
                long status = db.barangDAO().updateBarang(barang);
                return status;
            }

            @Override
            protected void onPostExecute(Long status) {
                Toast.makeText(RoomCreateActivity.this, "status row
"+status, Toast.LENGTH_SHORT).show();
            }
        }.execute();
    }

    private void insertData(final Barang barang){

        new AsyncTask<Void, Void, Long>(){
            @Override
            protected Long doInBackground(Void... voids) {
                long status = db.barangDAO().insertBarang(barang);
                return status;
            }

            @Override
            protected void onPostExecute(Long status) {
                Toast.makeText(RoomCreateActivity.this, "status row
"+status, Toast.LENGTH_SHORT).show();
            }
        }.execute();
    }
```

```
    public static Intent getActIntent(Activity activity) {
        // kode untuk pengambilan Intent
        return new Intent(activity, RoomCreateActivity.class);
    }
}
```

Menambahkan Fungsi Edit Data pada Data Access Object

Kemudan kita akan menambahkan fungsi baru untuk Edit data pada Data Access Object. Buka file *BarangDAO.java* kalian, dan copy paste kan kode di bawah ini :

```
package co.twoh.roomtutorial.data;

import android.arch.persistence.room.Dao;
import android.arch.persistence.room.Insert;
import android.arch.persistence.room.OnConflictStrategy;
import android.arch.persistence.room.Query;
import android.arch.persistence.room.Update;

import co.twoh.roomtutorial.model.Barang;

@Dao
public interface BarangDAO {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    long insertBarang(Barang barang);

    @Update
    int updateBarang(Barang barang);

    @Query("SELECT * FROM tbarang")
    Barang[] selectAllBarangs();

}
```

Pada source code di atas, kita menambahkan fungsi *updateBarang()* pada class Data Access Object. Method tersebut mempunyai annotation *@Update*, yang berarti method itu adalah method bawaan dari Room Database Library, dan yang perlu kita lakukan hanyalah menambahkan annotation dan method tersebut pada DAO, logic untuk update datanya akan dihandle langsung oleh Room database library.

Membuat Edit Data Flow

Selanjutnya kita membuat flow untuk Edit data. Jadi untuk melakukan Edit Data pada aplikasi, kita akan membuka activity *Read All data* terlebih dahulu. Kemudian pada list RecyclerView kita akan menambahkan *OnLongClickClickListener* pada RecyclerView Adapter, yang apabila kita melakukan long click pada RecyclerView item akan muncul Dialog. Pada Dialog itulah kita akan bisa memilih opsi Edit Data. Setelah opsi Edit Data dipilih, maka kita akan dibawa ke Activity berbeda dengan menyertakan Bundle data *Barang* pada Intent, untuk melakukan Edit Data.

Untuk source code yang diperlukan di flow ini, bisa dilihat di sini (*AdapterBarangRecyclerView.java*).

AdapterBarangRecyclerView.java

```
package co.twoh.roomtutorial.adapter;

import android.app.Activity;
import android.app.Dialog;
import android.arch.persistence.room.Room;
import android.content.Context;
import android.support.v7.widget.CardView;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import java.util.ArrayList;

import co.twoh.roomtutorial.R;
import co.twoh.roomtutorial.RoomCreateActivity;
import co.twoh.roomtutorial.RoomReadSingleActivity;
import co.twoh.roomtutorial.data.factory.AppDatabase;
import co.twoh.roomtutorial.model.Barang;

public class AdapterBarangRecyclerView extends
RecyclerView.Adapter<AdapterBarangRecyclerView.ViewHolder> {

    private ArrayList<Barang> daftarBarang;
    private Context context;
    private AppDatabase db;

    public AdapterBarangRecyclerView(ArrayList<Barang> barangs, Context
ctx){
        /**
         * Inisiasi data dan variabel yang akan digunakan
         */
        daftarBarang = barangs;
        context = ctx;

        db = Room.databaseBuilder(context.getApplicationContext(),
                AppDatabase.class,
"barangdb").allowMainThreadQueries().build();
    }

    class ViewHolder extends RecyclerView.ViewHolder {

        /**
         * Inisiasi View
         * Di tutorial ini kita hanya menggunakan data String untuk tiap
item
         * dan juga view nya hanyalah satu TextView
         */
        TextView tvTitle;
        CardView cvMain;

        ViewHolder(View v) {
```

```
            super(v);
            tvTitle = v.findViewById(R.id.tv_namabarang);
            cvMain = v.findViewById(R.id.cv_main);
        }
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        /**
         *  Inisiasi ViewHolder
         */
        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_barang,
parent, false);
        // mengeset ukuran view, margin, padding, dan parameter layout
lainnya
        ViewHolder vh = new ViewHolder(v);
        return vh;
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, final int position) {
        /**
         *  Menampilkan data pada view
         */
        final String name = daftarBarang.get(position).getNamaBarang();

        holder.cvMain.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                /**
                 *  Kodingan untuk tutorial Selanjutnya :p Read detail data
                 */
                Barang barang =
db.barangDAO().selectBarangDetail(daftarBarang.get(position).getBarangId())
;

context.startActivity(RoomReadSingleActivity.getActIntent((Activity)
context).putExtra("data", barang));
            }
        });
        holder.cvMain.setOnLongClickListener(new View.OnLongClickListener()
{
            @Override
            public boolean onLongClick(View view) {
                /**
                 *  Kodingan untuk tutorial Selanjutnya :p Delete dan
update data
                 */
                final Dialog dialog = new Dialog(context);
                dialog.setContentView(R.layout.view_dialog);
                dialog.setTitle("Pilih Aksi");
                dialog.show();

                Button editButton = dialog.findViewById(R.id.bt_edit_data);
                Button delButton =
dialog.findViewById(R.id.bt_delete_data);

                //apabila tombol edit diklik
                editButton.setOnClickListener(
                        new View.OnClickListener() {
```

```
                            @Override
                            public void onClick(View view) {
                                dialog.dismiss();
                                onEditBarang(position);
                            }
                        }
                );

                //apabila tombol delete diklik
                delButton.setOnClickListener(
                        new View.OnClickListener() {
                            @Override
                            public void onClick(View view) {
                                dialog.dismiss();
                                onDeteleBarang(position);
                            }
                        }
                );
                return true;
            }
        });
        holder.tvTitle.setText(name);
    }

    private void onDeteleBarang(int position){
        db.barangDAO().deleteBarang(daftarBarang.get(position));
        daftarBarang.remove(position);
        notifyItemRemoved(position);
        notifyItemRangeRemoved(position, daftarBarang.size());
    }

    private void onEditBarang(int position){
        context.startActivity(RoomCreateActivity.getActIntent((Activity)
context).putExtra("data", daftarBarang.get(position)));
    }

    @Override
    public int getItemCount() {
        /**
         * Mengembalikan jumlah item pada barang
         */
        return daftarBarang.size();
    }
}
```

Source code XML layout dari dialog yang muncul ketika kita melakukan OnLongClick, bisa dilihat di sini.

view_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="10dp"
    android:gravity="center"
    >
```

```
    <Button
        android:layout_margin="5dp"
        android:id="@+id/bt_edit_data"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Edit Data"
        />

    <Button
        android:layout_margin="5dp"
        android:id="@+id/bt_delete_data"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Delete"
        />
</LinearLayout>
```

Memanggil Fungsi Edit Data

Apabila langkah-langkah di atas sudah selesai dilakukan maka yang terakhir yang perlu dilakukan adalah mengintegrasikan step-step di atas dan melakukan pemanggilan dari fungsi Edit data dari DAO. Dengan terlebih dahulu menginisialisasi class AppDatabase.java yang sebelumnya sudah kita buat.

Codingan utamanya seperti di bawah ini, kita tambahkan pada class *RoomCreateActivity.java* :
Inisialisasi AppDatabase :

```
private AppDatabase db;

db = Room.databaseBuilder(getApplicationContext(),
            AppDatabase.class, "barangdb").build();
```

Execute fungsi Edit Data dengan memasukkan object Barang baru yang value nya sudah diubah :

```
private void updateBarang(final Barang barang){
    new AsyncTask<Void, Void, Long>(){
        @Override
        protected Long doInBackground(Void... voids) {
            long status = db.barangDAO().updateBarang(barang);
            return status;
        }

        @Override
        protected void onPostExecute(Long status) {
            Toast.makeText(RoomCreateActivity.this, "status row "+status,
Toast.LENGTH_SHORT).show();
        }
    }.execute();
}
```

CRUD Database di Android Menggunakan Room Library : Membuat Fungsi Delete Data

Membuat Fungsi Delete Data pada Room Database Library di Android

Menambahkan Fungsi Delete Data pada Data Access Object

Setelah kalian selesai membaca dan mempraktekkan tutorial-tutorial sebelumnya tentang Room database library ini. Buka project aplikasi inventaris sederhana kalian di Android Studio, dan pertama-tama, kita harus menambahkan method untuk delete data pada Data Access Object. Data Access Object (DAO) adalah sebuah class interface yang berisi method-method yang digunakan untuk mengakses database secara langsung. Buka interface *BarangDAO.java* dan copy pastekan kode di bawah ini :

```
package co.twoh.roomtutorial.data;

import android.arch.persistence.room.Dao;
import android.arch.persistence.room.Delete;
import android.arch.persistence.room.Insert;
import android.arch.persistence.room.OnConflictStrategy;
import android.arch.persistence.room.Query;
import android.arch.persistence.room.Update;

import co.twoh.roomtutorial.model.Barang;

@Dao
public interface BarangDAO {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    long insertBarang(Barang barang);

    @Update
    int updateBarang(Barang barang);

    @Delete
    int deleteBarang(Barang barang);

    @Query("SELECT * FROM tbarang")
    Barang[] selectAllBarangs();

}
```

Bisa dilihat pada codingan di atas, kita menambahkan method *deleteBarang()* pada DAO dan melemparkan object Barang berupa data Barang yang ingin kita hapus. Method delete barang menggunakan annotation *@Delete* yang berarti itu adalah method bawaan dari Room database library dan kita hanya perlu menggunakan annotation itu saja untuk membuat method delete data.

Membuat Delete Data Flow

Pada dasarnya flow untuk Delete data sama seperti flow untuk Update data. Yaitu pada aplikasi kita akan masuk ke halaman lihat semua data dan di situ kita akan melakukan Long Click pada item data yang ingin kita Delete. Setelah kita melakukan Long Click makan akan muncul dialog menu yang berisi opsi untuk "Edit data" dan "Delete data". Kemudian kita pilih opsi "Delete data" untuk menghapus data yang kita inginkan. Untuk source code yang diperlukan di flow ini, bisa dilihat di sini *AdapterBarangRecyclerView.java*. Source code XML layout dari dialog yang muncul ketika kita melakukan OnLongClick, bisa view_dialog.xml

Memanggil Fungsi Delete Data

Apabila langkah-langkah di atas sudah selesai dilakukan, maka next step nya adalah memanggil fungsi delete data pada DAO saat kita memilih opsi "Delete data" pada pop-up dialog. Dengan terlebih dahulu menginisialisasi class AppDatabase.java yang sebelumnya sudah kita buat.

Codingan utamanya adalah seperti di bawah, kita tambahkan pada class *AdapterBarangRecyclerView.java* :
Initiate AppDatabase :

```
private AppDatabase db;

db = Room.databaseBuilder(getApplicationContext(),
            AppDatabase.class, "barangdb").build();
```

Melakukan pemanggilan onDeleteBarang() dengan posisi barang sebagai parameter. Menggunakan parameter posisi itu kita akan mengambil data barang yang ingin dihapus:

```
private void onDeteleBarang(int position){
        db.barangDAO().deleteBarang(daftarBarang.get(position));
        daftarBarang.remove(position);
        notifyItemRemoved(position);
        notifyItemRangeRemoved(position, daftarBarang.size());
    }
```

CRUD Database Android Menggunakan Room : Membuat Fungsi Read Detail Data

Kapan-kapan….