Android Room

Praktikum Pemrograman Mobile - 05

Wanda Gusdya

2023

Menyimpan Data di Database Lokal

Aplikasi yang menangani data terstruktur dalam jumlah sangat banyak akan sangat terbantu jika data tersebut disimpan secara lokal. Kasus penggunaan yang paling umum adalah menyimpan bagian data yang relevan ke dalam cache sehingga jika perangkat tidak dapat mengakses jaringan, pengguna masih dapat menjelajahi konten tersebut meskipun offline.

Library persistensi Room menyediakan layer abstraksi atas SQLite untuk memungkinkan akses database yang lancar sambil memanfaatkan kemampuan penuh SQLite. Secara khusus, Room memberikan manfaat berikut:

- Verifikasi waktu kompilasi kueri SQL.
- Anotasi praktis yang meminimalkan kode boilerplate muncul berulang kali dan rentan error.
- Jalur migrasi database sederhana.

Karena pertimbangan ini, Anda disarankan menggunakan Room dan jangan menggunakan SQLite API secara langsung.

Latihan

Ubahlah beberapa bagian dari aplikasi yang dibangun di modul 4 untuk dapat menangani penyimpanan data ke database lokal.

Pertama-tama, bukalah file build.gredle untuk module app kemudian tambahkan plugin kotlin-kapt seperti berikut.

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'kotlin-kapt'
}
```

Kemudian, ubah versi kotlinCompilerExtensionVersion menjadi 1.4.3

```
composeOptions {
    kotlinCompilerExtensionVersion '1.4.3'
}
```

Di bagian dependencies, tambahkan deklarasi dependensi untuk Android Room

```
implementation "androidx.room:room-runtime:2.5.0"
implementation "androidx.room:room-ktx:2.5.0"
kapt "androidx.room:room-compiler:2.5.0"
```

Selanjutnya, tambahkan dependensi uuid untuk membuat primary key berbasis UUID

```
implementation "com.benasher44:uuid:0.4.0"
```

Universally unique identifier (UUID) adalah pengenal dengan panjang 128-bit yang biasa digunakan pada sistem informasi berbasis komputer. UUID yang dibuat sesuai dengan metode standar merupakan

pengenal unik. Keunikannya tidak bergantung pada sebuah sistem registrasi terpusat atau koordinasi antara pihak-pihak yang membuatnya, seperti kebanyakan skema penomoran lainnya. Probabilitas bahwa UUID dapat diduplikasi memang tidak nol, namun dianggap cukup mendekati nol untuk diabaikan.

Selanjutnya, tambahkan dependensi untuk penggunaan Live Data di aplikasi.

```
implementation "androidx.lifecycle:lifecycle-runtime-compose:2.6.0-rc01"
implementation "androidx.compose.runtime:runtime-
livedata:$compose_ui_version"
```

Update juga versi dependensi lain seperti berkut.

```
implementation 'androidx.core:core-ktx:1.9.0'
implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.5.1'
implementation 'androidx.activity:activity-compose:1.6.1'
implementation "androidx.compose.ui:ui:$compose_ui_version"
implementation "androidx.compose.ui:ui-tooling-preview:$compose_ui_version"
implementation 'androidx.compose.material:material:1.3.1'
testImplementation 'junit:junit:4.13.2'
androidTestImplementation 'androidx.test.ext:junit:1.1.5'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
androidTestImplementation "androidx.compose.ui:ui-test-
junit4:$compose_ui_version"
debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
debugImplementation "androidx.compose.ui:ui-test-
manifest:$compose_ui_version"
```

Sehingga, sekarang file build.gradle untuk module app terlihat seperti berikut.

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'kotlin-kapt'
}

android {
    namespace 'id.ac.unpas.functionalcompose'
    compileSdk 33

    defaultConfig {
        applicationId "id.ac.unpas.functionalcompose"
        minSdk 24
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
```

```
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion '1.4.3'
    }
    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {
    implementation 'androidx.core:core-ktx:1.9.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.5.1'
    implementation 'androidx.activity:activity-compose:1.6.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-
preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.3.1'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation "androidx.compose.ui:ui-test-
junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-
manifest:$compose_ui_version"


    implementation "androidx.room:room-runtime:2.5.0"
    implementation "androidx.room:room-ktx:2.5.0"
    kapt "androidx.room:room-compiler:2.5.0"
    implementation "com.benasher44:uuid:0.4.0"
    implementation "androidx.lifecycle:lifecycle-runtime-compose:2.6.0-rc01"
    implementation "androidx.compose.runtime:runtime-
livedata:$compose_ui_version"

}
```

Selanjutnya, bukalah file build.gradle level project, kemudian update versi compose_ui_version menjadi versi 1.3.3 dan org.jetbrains.kotlin.android menjadi versi 1.8.10 seperti berikut.

```
buildscript {
    ext {
        compose_ui_version = '1.3.3'
    }
}// Top-level build file where you can add configuration options common to
all sub-projects/modules.
plugins {
    id 'com.android.application' version '7.4.1' apply false
    id 'com.android.library' version '7.4.1' apply false
    id 'org.jetbrains.kotlin.android' version '1.8.10' apply false
}
```

Lakukan Gradle Sync untuk project, lalu bukalah kelas SetoranSampah pada package model. Tambahkan anotasi Entity untuk kelas tersebut.

```
@Entity
data class SetoranSampah(
```

Tambahkan import untuk anotasi tersebut

```
import androidx.room.Entity
```

Kemudian, tambahkan atribut id yang akan kita gunakan sebagai primary key. Kita perlu menambahkan anotasi PrimaryKey untuk atribut tersebut.

```
@PrimaryKey val id: String,
```

Import yang digunakan untuk anotasi PrimaryKey adalah

```
import androidx.room.PrimaryKey
```

Sekarang, kelas SetoranSampah akan terlihat seperti berikut.

```
package id.ac.unpas.functionalcompose.model

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity
data class SetoranSampah(
    @PrimaryKey val id: String,
    val tanggal: String,
    val nama: String,
    val berat: String
)
```

Selanjutnya, buat package baru bernama persistences lalu buat interface kotlin baru bernama SetoranSampahDao dengan kode sebagai berikut.

```
package id.ac.unpas.functionalcompose.persistences

import androidx.lifecycle.LiveData
```

```
import androidx.room.*
import id.ac.unpas.functionalcompose.model.SetoranSampah

@Dao
interface SetoranSampahDao {
    @Query("SELECT * FROM SetoranSampah")
    fun loadAll(): LiveData<List<SetoranSampah>>

    @Query("SELECT * FROM SetoranSampah WHERE id = :id")
    fun find(id: String): SetoranSampah?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(vararg items: SetoranSampah)

    @Delete
    fun delete(item: SetoranSampah)
}
```

Masih di package persistences, buat sebuah kelas abstrak bernama AppDatabase dengan kode sebagai berikut.

```
package id.ac.unpas.functionalcompose.persistences

import androidx.room.Database
import androidx.room.RoomDatabase
import id.ac.unpas.functionalcompose.model.SetoranSampah

@Database(entities = [SetoranSampah::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun setoranSampahDao(): SetoranSampahDao
}
```

Selanjutnya, bukalah file PengelolaanSampahScreen.kt kemudian tambahkan beberapa deklarasi variabel untuk mengambil context, inisialisasi database, dan dao.

```
@Composable
fun PengelolaanSampahScreen() {
    //TODO 5. Inisiasi kelas db lalu ambil dao, ubah list jadi livedata, ubah
parameter items di baris 48
    val context = LocalContext.current

    val db = Room.databaseBuilder(
        context,
        AppDatabase::class.java, "pengelolaan-sampah"
    ).build()

    val setoranSampahDao = db.setoranSampahDao()
```

Kemudian, ubah dua baris kode berikut

```
val _list = remember { MutableStateFlow(listOf<SetoranSampah>()) }
val list by remember { _list }.collectAsState()
```

Dengan kode berikut

```
val list : LiveData<List<SetoranSampah>> = setoranSampahDao.loadAll()
val items: List<SetoranSampah> by list.observeAsState(initial = listOf())
```

Ubah juga kode berikut

```
items(items = list, itemContent = { item ->
```

menjadi

```
items(items = items, itemContent = { item ->
```

Kemudian, buka file FormPencatatanSampah.kt lalu ubah parameter onSimpan mennadi SetoranSampahDao seperti berikut.

```
fun FormPencatatanSampah(setoranSampahDao: SetoranSampahDao) {
```

Tambahkan juga variabel scope untuk referensi coroutine scope yang akan digunakan untuk menyimpan data di background thread.

```
val scope = rememberCoroutineScope()
```

Di bagian penangan onClick button simpan, ubah kode menjadi seperti berikut.

```
val id = uuid4().toString()
val item = SetoranSampah(id, tanggal.value.text, nama.value.text,
berat.value.text)
scope.launch {
    setoranSampahDao.insertAll(item)
}
tanggal.value = TextFieldValue("")
nama.value = TextFieldValue("")
berat.value = TextFieldValue("")
```

Import fungsi uuid4() menggunakan statement berikut

```
import com.benasher44.uuid.uuid4
```

Sekarang, file FormPencatatanSampah.kt akan terlihat seperti berikut.

```
package id.ac.unpas.functionalcompose.screens

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.Button
import androidx.compose.material.ButtonDefaults
import androidx.compose.material.OutlinedTextField
import androidx.compose.material.Text
```

```
import androidx.compose.runtime.Composable
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.rememberCoroutineScope
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.input.KeyboardCapitalization
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.TextFieldValue
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.benasher44.uuid.uuid4
import id.ac.unpas.functionalcompose.model.SetoranSampah
import id.ac.unpas.functionalcompose.persistences.SetoranSampahDao
import id.ac.unpas.functionalcompose.ui.theme.Purple700
import id.ac.unpas.functionalcompose.ui.theme.Teal200
import kotlinx.coroutines.launch

@Composable
fun FormPencatatanSampah(setoranSampahDao: SetoranSampahDao) {
    val tanggal = remember { mutableStateOf(TextFieldValue("")) }
    val nama = remember { mutableStateOf(TextFieldValue("")) }
    val berat = remember { mutableStateOf(TextFieldValue("")) }

    val scope = rememberCoroutineScope()

    Column(modifier = Modifier
        .padding(10.dp)
        .fillMaxWidth()) {

        OutlinedTextField(
            label = { Text(text = "Tanggal") },
            value = tanggal.value,
            onValueChange = {
                tanggal.value = it
            },
            modifier = Modifier
                .padding(4.dp)
                .fillMaxWidth(),
            placeholder = { Text(text = "yyyy-mm-dd") }
        )

        OutlinedTextField(
            label = { Text(text = "Nama") },
            value = nama.value,
            onValueChange = {
                nama.value = it
            },
            modifier = Modifier
                .padding(4.dp)
                .fillMaxWidth(),
            keyboardOptions = KeyboardOptions(capitalization =
KeyboardCapitalization.Characters, keyboardType = KeyboardType.Text),
            placeholder = { Text(text = "XXXXX") }
        )
```

```
        OutlinedTextField(
            label = { Text(text = "Berat") },
            value = berat.value,
            onValueChange = {
                berat.value = it
            },
            modifier = Modifier
                .padding(4.dp)
                .fillMaxWidth(),
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Decimal),
            placeholder = { Text(text = "5") }
        )

        val loginButtonColors = ButtonDefaults.buttonColors(
            backgroundColor = Purple700,
            contentColor = Teal200
        )

        val resetButtonColors = ButtonDefaults.buttonColors(
            backgroundColor = Teal200,
            contentColor = Purple700
        )

        Row (modifier = Modifier
            .padding(4.dp)
            .fillMaxWidth()) {
            Button(modifier = Modifier.weight(5f), onClick = {
                val id = uuid4().toString()
                val item = SetoranSampah(id, tanggal.value.text,
nama.value.text, berat.value.text)
                scope.launch {
                    setoranSampahDao.insertAll(item)
                }
                tanggal.value = TextFieldValue("")
                nama.value = TextFieldValue("")
                berat.value = TextFieldValue("")
            }, colors = loginButtonColors) {
                Text(
                    text = "Simpan",
                    style = TextStyle(
                        color = Color.White,
                        fontSize = 18.sp
                    ), modifier = Modifier.padding(8.dp)
                )
            }

            Button(modifier = Modifier.weight(5f), onClick = {
                tanggal.value = TextFieldValue("")
                nama.value = TextFieldValue("")
                berat.value = TextFieldValue("")
            }, colors = resetButtonColors) {
                Text(
                    text = "Reset",
                    style = TextStyle(
                        color = Color.White,
                        fontSize = 18.sp
```

```
                    ), modifier = Modifier.padding(8.dp)
                )
            }
        }
    }
}
```

Kembali ke file PengelolaanSampahScreen.kt, ubah pemanggilan fungsi FormPencatatanSampah menjadi seperti berikut.

```
FormPencatatanSampah(setoranSampahDao)
```

Jalankan aplikasi, kemudian amati hasilnya. Kita tidak perlu mengaitkan data secara langsung seperti di modul 4, karena sudah ditangani oleh LiveData. LiveData akan selalu mempertahankan data terbaru dari database lokal.