Hilt dan ViewModel

Praktikum Pemrograman Mobile - 07

Wanda Gusdya

2023

Dependency Injection

Pada modul ini, kita akan mempelajari cara menginjeksi aplikasi dengan Injeksi Dependensi menggunakan Dagger-Hilt di Jetpack Compose. Injeksi dependensi adalah pola desain yang bertujuan untuk memisahkan perhatian membangun objek dan menggunakannya dengan membentuk inversi kontrol, yang mengarah ke program yang digabungkan secara longgar. Di Android, ada dua metode populer untuk Injeksi Dependensi. Koin dan Dagger-Hilt. Kita akan menggunakan Dagger-Hilt pada modul ini.

Latihan

Ubah project dari modul 5 untuk dapat menggunakan Dagger Hilt dan menerapkan pola rancangan MVVM (model-view-viewmodel). Pertama-tama, tambahkan dependensi plugin pada file build.gradle level Project.

```
id 'com.google.dagger.hilt.android' version '2.45' apply false
```

Sehingga, file akan terlihat 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
    id 'com.google.dagger.hilt.android' version '2.45' apply false
}
```

Selanjutnya, tambahkan dependensi berikut pada file build.gradle level module app.

```
implementation "com.google.dagger:hilt-android:2.45"
implementation "androidx.hilt:hilt-navigation-compose:1.0.0"
kapt "com.google.dagger:hilt-compiler:2.45"
kapt "androidx.hilt:hilt-compiler:1.0.0"
androidTestImplementation "com.google.dagger:hilt-android-testing:2.45"
kaptAndroidTest "com.google.dagger:hilt-compiler:2.45"
```

Kemudian, tambahkan plugin hilt di bagian atas file build.gradle tersebut.

```
id 'dagger.hilt.android.plugin'
```

Sehingga, file build.gradle akan terlihat seperti berikut.

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

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"

    implementation "com.google.dagger:hilt-android:2.45"
    implementation "androidx.hilt:hilt-navigation-compose:1.0.0"
    kapt "com.google.dagger:hilt-compiler:2.45"
    kapt "androidx.hilt:hilt-compiler:1.0.0"
    androidTestImplementation "com.google.dagger:hilt-android-testing:2.45"
    kaptAndroidTest "com.google.dagger:hilt-compiler:2.45"

}
```

Kemudian, buatlah kelas kotlin baru di package utama (misal id.ac.unpas.functionalcompose) dengan nama BankSampahApp. Isi kelas tersebut dengan kode berikut.

```
package id.ac.unpas.functionalcompose

import android.app.Application
import dagger.hilt.android.HiltAndroidApp

@HiltAndroidApp
class BankSampahApp : Application()
```

Lalu, buka file manifest, tambahkan atribut name pada tag application seperti berikut.

```
android:name=".BankSampahApp"
```

Sehingga, file manifest terlihat seperti berikut.

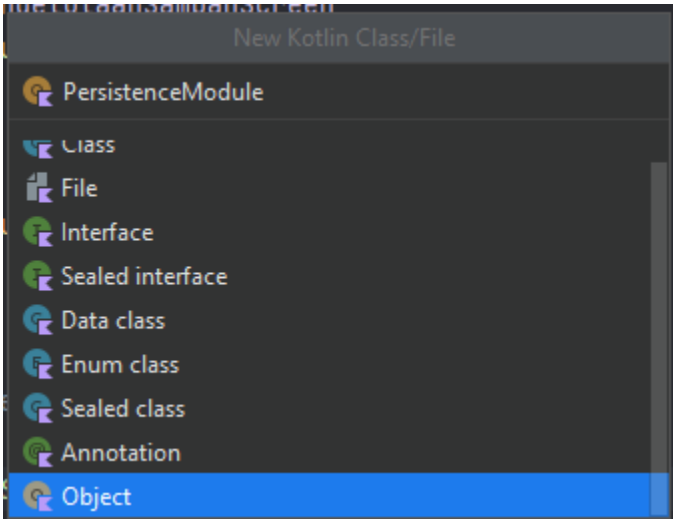
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.FunctionalCompose"
        android:name=".BankSampahApp"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.FunctionalCompose">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Selanjutnya, tambahkan anotasi @AndroidEntryPoint ke kelas MainActivity, seperti berikut.

```
package id.ac.unpas.functionalcompose

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import dagger.hilt.android.AndroidEntryPoint
import id.ac.unpas.functionalcompose.screens.PengelolaanSampahScreen
import id.ac.unpas.functionalcompose.ui.theme.FunctionalComposeTheme

@AndroidEntryPoint
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        setContent {
            FunctionalComposeTheme {
                // A surface container using the 'background' color from the
theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    PengelolaanSampahScreen()
                }
            }
        }
    }
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    FunctionalComposeTheme {
        PengelolaanSampahScreen()
    }
}
```

Kemudian, buat package baru dengan nama di (singkatan dari dependency injection). Buat sebuah file kotlin bernama PersistenceModule dengan tipe Object (paling bawah pada pilihan New Kotlin Class/File) seperti pada gambar berikut.



Beri kode seperti berikut.

```
package id.ac.unpas.functionalcompose.di

import android.app.Application
import androidx.room.Room
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import id.ac.unpas.functionalcompose.persistences.AppDatabase
import id.ac.unpas.functionalcompose.persistences.SetoranSampahDao
import javax.inject.Singleton

@Module
@InstallIn(SingletonComponent::class)
object PersistenceModule {
    @Provides
    @Singleton
    fun provideAppDatabase(application: Application): AppDatabase {
        return Room
            .databaseBuilder(
                application,
                AppDatabase::class.java,
                "pengelolaan-sampah"
            )
            .fallbackToDestructiveMigration()
            .build()
    }

    @Provides
    @Singleton
    fun provideSetoranSampahDao(appDatabase: AppDatabase): SetoranSampahDao {
        return appDatabase.setoranSampahDao()
    }
}
```

File object di atas adalah sebuah modul yang berisi objek-objek yang dibutuhkan (dependensi) oleh kelas/fungsi lain. Dagger Hilt akan menyediakan instan dari spesifikasi yang sudah ditentukan di file modul ini.

Selanjutnya, buatlah kelas PengelolaanSampahViewModel di package screens lalu tambahkan kode berikut.

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

import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
import dagger.hilt.android.lifecycle.HiltViewModel
import id.ac.unpas.functionalcompose.model.SetoranSampah
import id.ac.unpas.functionalcompose.persistences.SetoranSampahDao
import javax.inject.Inject

@HiltViewModel
```

```
class PengelolaanSampahViewModel @Inject constructor(private val
setoranSampahDao: SetoranSampahDao) : ViewModel() {
    val list : LiveData<List<SetoranSampah>> = setoranSampahDao.loadAll()

    suspend fun insert(id: String,
                       tanggal: String,
                       nama: String,
                       berat: String){
        val item = SetoranSampah(id, tanggal, nama, berat)
        setoranSampahDao.insertAll(item)
    }
}
```

Lalu, ubah file PengelolaanSampahScreen, hapus kode di bawah ini

```
    val context = LocalContext.current

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

    val setoranSampahDao = db.setoranSampahDao()

    val list : LiveData<List<SetoranSampah>> = setoranSampahDao.loadAll()
```

Tambahkan kode berikut di bawah header fungsi.

```
val viewModel = hiltViewModel<PengelolaanSampahViewModel>()
```

Lalu, ubah kode berikut

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

dengan

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

Lalu, buka file FormPencatatanSampah lalu hapus parameter fungsi menjadi seperti berikut.

```
fun FormPencatatanSampah() {
```

Kemudian ubah penangan event onclick untuk button simpan menjadi seperti berikut.

```
val id = uuid4().toString()
scope.launch {
    viewModel.insert(id, tanggal.value.text, nama.value.text,
berat.value.text)
    tanggal.value = TextFieldValue("")
    nama.value = TextFieldValue("")
    berat.value = TextFieldValue("")
}
```

Sekarang, file PengelolaanSampahScreen 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.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.Divider
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.livedata.observeAsState
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import id.ac.unpas.functionalcompose.model.SetoranSampah

@Composable
fun PengelolaanSampahScreen() {
    val viewModel = hiltViewModel<PengelolaanSampahViewModel>()

    val items: List<SetoranSampah> by viewModel.list.observeAsState(initial =
listOf())

    Column(modifier = Modifier.fillMaxWidth()) {
        FormPencatatanSampah()

        LazyColumn(modifier = Modifier.fillMaxWidth()) {
            items(items = items, itemContent = { item ->

                Row(modifier = Modifier
                    .padding(15.dp)
                    .fillMaxWidth()) {
                    Column(modifier = Modifier.weight(3f)) {
                        Text(text = "Tanggal", fontSize = 14.sp)
                        Text(text = item.tanggal, fontSize = 16.sp,
```

```
fontWeight = FontWeight.Bold)
                    }

                    Column(modifier = Modifier.weight(3f)) {
                        Text(text = "Nama", fontSize = 14.sp)
                        Text(text = item.nama, fontSize = 16.sp, fontWeight =
FontWeight.Bold)
                    }

                    Column(modifier = Modifier.weight(3f)) {
                        Text(text = "Berat", fontSize = 14.sp)
                        Text(text = "${item.berat} Kg", fontSize = 16.sp,
fontWeight = FontWeight.Bold)
                    }
                }

                Divider(modifier = Modifier.fillMaxWidth())

            })
        }
    }

}
```

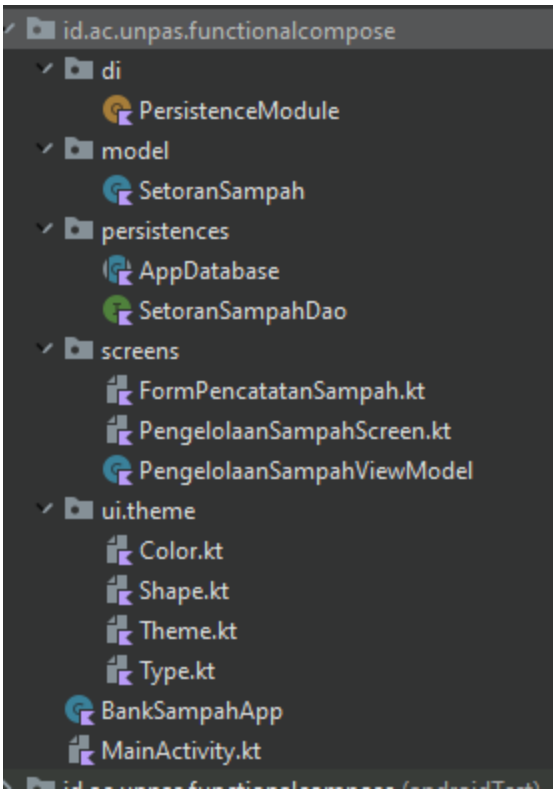
File FormPencatatanSampah akan terlihat seperti berikut.

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

import android.util.Log
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 androidx.hilt.navigation.compose.hiltViewModel
import com.benasher44.uuid.uuid4
import id.ac.unpas.functionalcompose.ui.theme.Purple700
import id.ac.unpas.functionalcompose.ui.theme.Teal200
```

```
import kotlinx.coroutines.launch

@Composable
fun FormPencatatanSampah() {
    val viewModel = hiltViewModel<PengelolaanSampahViewModel>()

    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()
                scope.launch {
                    viewModel.insert(id, tanggal.value.text, nama.value.text,
berat.value.text)
                    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)
                )
            }
        }
    }
}
```

Sekarang, struktur proyek terlihat seperti ini.



Ketika aplikasi dijalankan, tidak akan ada perbedaan. Namun, pola ini akan menguntungkan proses development terutama dalam pengujian. Kita tidak perlu membuat instan untuk setiap penggunaan kelas Dao dan AppDatabase.