Android Jetpack : ViewModel & LiveData

Semangat pagi!

Pada pertemuan kali ini akan dibahas tentang ViewModel yang merupakan bagian dari Architecture Android Jetpack dan dalam artikel ini akan belajar beberapa point seperti berikut ini :

1. What's ViewModel ?
2. What's LiveData ?
3. How to Implementation it ?

Silahkan cari referensi lebih lanjut tentang ViewModel, di sini akan belajar juga bagaimana mengimplementasikan ke dalam kode pengembangan Aplikasi Android. Dalam studi kasus nya nanti akan dibuat Aplikasi Matematika dengan menggunakan ViewModel & LiveData.

1. What's ViewModel?

ViewModel adalah sebuah object yang menyediakan data untuk komponen yang ada di UI dan akan mempertahankan data jika ada perubahan state (misalnya konfigurasi layar).
Pernahkah mengalam kesulitan untuk menjaga tampilan data pada UI ? Contohnya seperti menampilkan data dalam bentuk tipe data string dan ketika terjadi perubahan konfigurasi layar akan kehilangan data string tersebut menjadi kosong.
Bagaimana cara menjaga datanya supaya tidak hilang? Pendekatan kode seperti ini dapat digunakan untuk menjaga datanya :

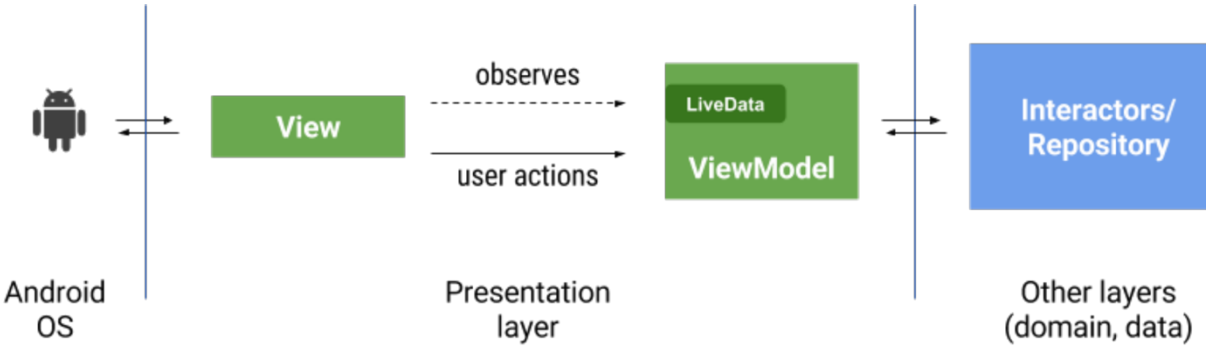
```
// menggunakan savedInstanceState untuk menjaga data
if (savedInstanceState != null) {
    ...
}
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    ...
}
```

Bagaimana datanya dalam jumlah banyak yang harus di jaga pada state UI komponennya? Jika menggunakan pendekatan kode diatas akan tidak efisien, mengalami memory leaks dan bahkan crash pada aplikasi, sehingga ViewModel adalah jawabannya.

2. What's LiveData?

LiveData adalah sebuah class data holder yang artinya LiveData memegang peranan penting untuk menyampaikan data ke UI dalam ViewModel. Jika ada ada perubahan data, LiveData akan memberikan notifikasi kepada UI bahwa ada update data terbaru yang secara bersamaan bekerja dengan class ViewModel. LiveData mengetahui aktifitas lifecycle dari UI pada aplikasi android. Jadi dia paham betul kapan harus menyampaikan datanya.

Perhatikan pada gambar berikut ini :



Gambar diatas menunjukkan bahwa LiveData tidak akan menyampaikan data ke UI jika dari UI tersebut tidak active dan sebaliknya, jika UI active LiveData akan secara otomatis menyampaikan data nya ke UI melalui class ViewModel yang sebelumnya sudah berkomunikasi dengan UI.

3. How to Implementation it?

Sekarang akan mengimplementasikan bagaimana ViewModel dan LiveData bekerja secara bersamaan.

Class ViewModel :
Buat satu class ViewModel yang extending ke class ViewModel seperti contoh dibawah ini :

```
class MainViewModel : ViewModel() {
}
```

Satu buah class dikatakan sebagai ViewModel apabila class tersebut extending class ViewModel seperti approach code diatas. Dan dalam class MainViewModel tambahkan approach code berikut ini :

```
val liveData = MutableLiveData<Result>() // => can change value
// fungsi ini akan melakukan kalkulasi perhitungan
fun setHitung(panjang: String, lebar: String): LiveData<Result> {
    val p = panjang.toDouble()
    val l = lebar.toDouble()
    val tampung = p * l
    val result = Result(tampung.toString())
    liveData.postValue(result)
    return liveData
}
// fungsi ini akan mengembalikan data hasil kalkulasi pada fungsi diatas
dalam livedata
fun getHitung(): LiveData<Result> {
    return liveData
}
```

Catatan :
LiveData bersifat read-only
MutableLiveData dapat merubah value dari data nya (setValue||postValue)

Selanjutnya adalah pada class MainActivity buat approach code nya seperti ini :

```
class MainActivity : AppCompatActivity() {

    // deklarasi ViewModel
    lateinit var viewModel: MainViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        /**
         * communication your activity/fragment with MainViewModel
         * */
        viewModel =
ViewModelProviders.of(this).get(MainViewModel::class.java)
        /**
         * get value from your live data into MainViewModel
         * */
        viewModel.getHitung().observe(this, getHitung) // => asynchronous

        btn_hitung.setOnClickListener {
            hitung(edt_panjang.text.toString().trim(),
edt_panjang.text.toString().trim())
        }
    }
    fun hitung(panjang: String, lebar: String) {
        if (panjang.isEmpty() || lebar.isEmpty()) {
            toast("kosong cuy")
        } else {
            viewModel.setHitung(edt_panjang.text.toString().trim(),
edt_lebar.text.toString().trim())
        }
    }
    /**
     * SUBSCRIBE
     * call this if you want get your data when function gethitung() to do
observe
     * */
    val getHitung = Observer<Result> {
        it.let {
            tv_result.text = it?.result
        }
    }
}
```

Jika sudah silahkan run approach code diatas. Dan jika dijalankan coba lakukan kalkulasi perhitungan nya. Setelah itu coba rubah konfigurasi layar pada aplikasi dan lihat datanya akan terjaga pada state UI, sehingga tidak akan kehilangan pada kalkulasi perhitungan pada data di UI.

Kesimpulan

1. Memahami apa itu ViewModel.
2. Memahami api itu LiveData.
3. Memahami ViewModel & LiveData works.
4. Memahami bagaimana ViewModel menjaga data jika ada perubahan konfigurasi pada layar.
5. Memahami cara mengimplementasikan ViewModel & LiveData.

Untuk full project-nya silahkan cek repository berikut ini:
https://github.com/PiusAnggoro/Belajar-ViewModel