MODUL 9

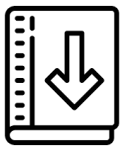
View Model

CAPAIAN PEMBELAJARAN

Mahasiswa dapat membuat Aplikasi dengan database.

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



DASAR TEORI

Pengantar

Lebih dari dua tahun yang lalu, saya bekerja di Android untuk Pemula; kelas yang membawa siswa dari nol pemrograman ke aplikasi Android pertama mereka. Sebagai bagian dari kursus, siswa membuat aplikasi satu layar yang sangat sederhana bernama Court-Counter.

Court-Counter adalah aplikasi yang sangat mudah dengan tombol yang mengubah skor bola basket. Aplikasi yang sudah selesai memiliki bug; jika Anda memutar telepon, skor Anda saat ini akan hilang secara misterius.

Apa yang sedang terjadi? Memutar perangkat adalah salah satu dari sedikit perubahan konfigurasi yang dapat dilalui aplikasi selama masa pakainya, termasuk ketersediaan keyboard dan mengubah bahasa perangkat. Semua perubahan konfigurasi ini menyebabkan Kegiatan dirobohkan dan diciptakan kembali.

Perilaku ini memungkinkan kami untuk melakukan hal-hal seperti menggunakan tata letak khusus orientasi lanskap ketika perangkat diputar pada sisi nya. Sayangnya itu bisa menjadi sakit kepala bagi insinyur baru (dan kadang-kadang tidak begitu baru) untuk membungkus kepala mereka.

Di Google I / O 2017, tim Kerangka Android memperkenalkan seperangkat Komponen Arsitektur baru, yang salah satunya berkaitan dengan masalah rotasi yang tepat ini.

Kelas ViewModel dirancang untuk menampung dan mengelola data terkait UI dengan cara yang sadar siklus hidup. Ini memungkinkan data untuk selamat dari perubahan konfigurasi seperti rotasi layar.

Posting ini adalah yang pertama dalam seri yang mengeksplorasi seluk beluk ViewModel. Dalam pos ini saya akan:

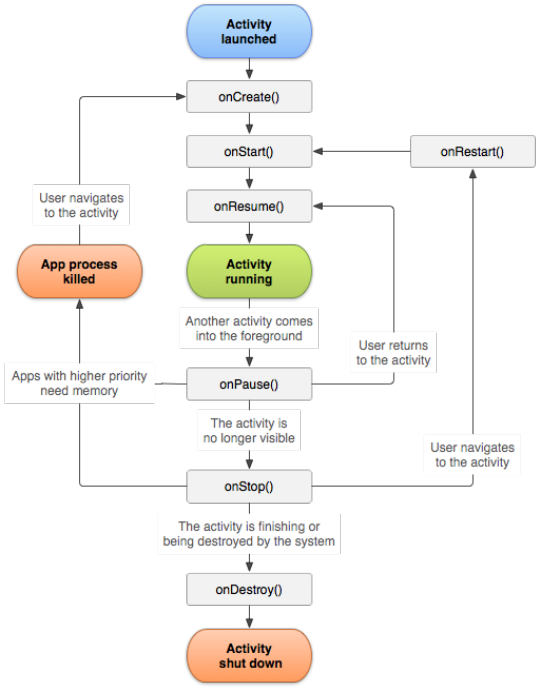
Jelaskan kebutuhan dasar yang dipenuhi oleh ViewModels

Selesaikan masalah rotasi dengan mengubah kode Court-Counter untuk menggunakan ViewModel

Lihatlah lebih dekat pada ViewModel dan asosiasi Komponen UI

Masalah yang mendasarinya

Tantangan yang mendasarinya adalah bahwa siklus hidup Aktivitas Android memiliki banyak status dan Kegiatan tunggal mungkin berulang kali melalui berbagai status berbeda ini karena perubahan konfigurasi.



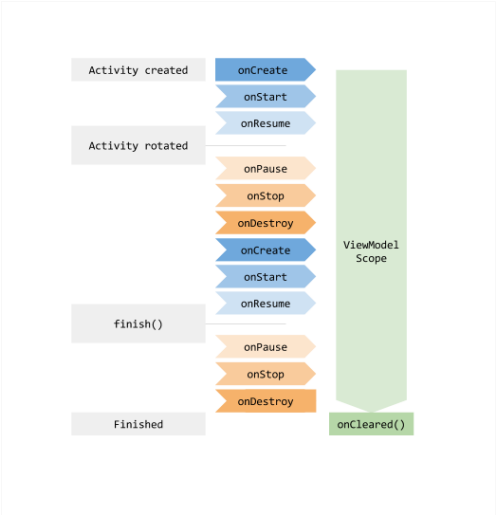
Saat sebuah Aktivitas sedang melalui semua status ini, Anda juga mungkin memiliki data UI sementara yang perlu Anda simpan dalam memori. Saya akan mendefinisikan data UI sementara sebagai data yang diperlukan untuk UI. Contohnya termasuk data yang dimasukkan pengguna, data yang dihasilkan selama runtime, atau data yang dimuat dari database. Data ini bisa berupa gambar bitmap, daftar objek yang diperlukan untuk RecyclerView atau, dalam hal ini, skor bola basket.

Sebelumnya, Anda mungkin telah menggunakan onRetainNonConfigurationInstance untuk menyimpan data ini selama perubahan konfigurasi dan membongkarnya di ujung yang lain.

Tetapi apakah itu tidak akan membengkak jika data Anda tidak perlu tahu atau mengelola status siklus hidup yang ada di dalam Aktivitas? Alih-alih memiliki variabel seperti scoreTeamA dalam Kegiatan, dan oleh karena itu terkait dengan semua tingkah laku siklus Kegiatan, bagaimana jika data itu disimpan di tempat lain, di luar Kegiatan? Ini adalah tujuan dari kelas ViewModel.

Pada diagram di bawah ini, Anda dapat melihat siklus hidup suatu Aktivitas yang mengalami rotasi dan akhirnya selesai. Masa pakai ViewModel ditampilkan di sebelah siklus hidup Aktivitas terkait. Perhatikan bahwa ViewModels dapat dengan mudah digunakan dengan Fragmen dan Aktivitas, yang saya sebut pengontrol UI. Contoh ini berfokus pada Kegiatan.

ViewModel muncul sejak pertama kali Anda meminta ViewModel (biasanya di onCreate the Activity) hingga Activity selesai dan dihancurkan. onCreate dapat dipanggil beberapa kali selama kehidupan suatu Kegiatan, seperti ketika aplikasi diputar, tetapi ViewModel bertahan sampai sekarang.



Beberapa penjelasan akan disampaikan di langkah praktik.

PRAKTIK

Langkah ke Implementasi ViewModel

Saya akan menunjukkan kepada Anda bagaimana cara menggunakan ViewModel. Sangat mudah Anda harus mengikuti langkah di bawah ini

Tambahkan dependensi di build.gradle

Buat subkelas ViewModel

Ekspos metode untuk data seperti getInitialCount () dan getCurrentCount () untuk kedua pemain.

Tulis kode dalam Aktivitas

1. Tambahkan dependensi di build.gradle

Untuk menerapkan contoh ViewModel Android, Anda harus membuka menu file dan membuat proyek baru. buka build.gradle dan tambahkan baris kode di bawah ini

```
    def lifecycle_version = "1.1.1"
    implementation "android.arch.lifecycle:extensions:$lifecycle_versi
on"
    annotationProcessor "android.arch.lifecycle:compiler:$lifecycle_ve
rsion"
```

2. Buat subkelas dari ViewModel

Sekarang buat kelas baru dengan nama MainViewModel yang memperluas kelas ViewModel dan memaparkan metode di bawah ini.

```
class MainViewModel : ViewModel() {
    var initialCountA: Int = 0
        private set
    var initialCountB = 0
        private set

    val currentCountA: Int
        get() {
            initialCountA++
            return initialCountA
        }

    val currentCountB: Int
        get() {
            initialCountB++
            return initialCountB
        }
}
```

3. Metode Paparan

Sekarang buka MainActivity dan tambahkan baris kode di bawah ini

```
class MainActivity : AppCompatActivity(), View.OnClickListener {
    lateinit  var tvScoreA: TextView
    lateinit  var tvScoreB: TextView
    lateinit  var btnPlayerA: Button
    lateinit  var btnPlayerB: Button
    private lateinit var mainViewModel: MainViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        initView()
        mainViewModel =
ViewModelProviders.of(this).get(MainViewModel::class.java)

        tvScoreA.text = mainViewModel.initialCountA.toString()
        tvScoreB.text = mainViewModel.initialCountB.toString()
    }
```

```
    private fun initView() {
        tvScoreA = findViewById(R.id.tvScorePlayerA)
        tvScoreB = findViewById(R.id.tvScorePlayerB)
        btnPlayerA = findViewById(R.id.btnPlayerA)
        btnPlayerB = findViewById(R.id.btnPlayerB)
        btnPlayerA.setOnClickListener(this)
        btnPlayerB.setOnClickListener(this)
    }

    override fun onClick(v: View) {
        when (v.id) {
            R.id.btnPlayerA ->
                tvScoreA.text = mainViewModel.currentCountA.toString()
            R.id.btnPlayerB ->
                tvScoreB.text = mainViewModel.currentCountB.toString()
        }
    }
}
```

Akhirnya, mengikuti semua langkah-langkah ini hanya LARI proyek, Output akan seperti yang diharapkan, penghitung tidak akan mengatur ulang nol pada perubahan konfigurasi, seperti di sini.

Tutorial ViewModel

Ini adalah contoh dasar ViewModel Android. Jika Anda memiliki pertanyaan, silakan bertanya di bagian komentar di bawah. Selamat Pengodean 🙂

LATIHAN

Modifikasilah aplikasi dengan menambahkan Toast jika salah satu list dipilih.

TUGAS

Buat aplikasi baru dengan menerapkan RecycleView

REFERENSI

1. https://kotlinlang.org/docs/reference/
2. https://developer.android.com/kotlin
3. https://developer.android.com/courses/kotlin-android-fundamentals/toc

4. https://codelabs.developers.google.com/android-kotlin-fundamentals/
5. https://developer.android.com/kotlin/learn
6. https://developer.android.com/kotlin/resources