How to MVVM

0. 개요

0. 개요

Model

데이터 모델, 내부 DB 접근 등 데이터 소스를 담당

View

Activity, Fragment 등 전반적인 View를 담당

ViewModel을 관측하며, 변화된 ViewModel을 View에 반영

ViewModel

View에 대한 Model

Model에서 제공받은 데이터를 가지고 있고, 상태 변경을 View에 알림

1 : N 구조로 사용 가능

DataBinding으로 View와 소통

1. 구현해보기

하기 전에

1. How to DataBinding

DataBinding

1. How to DataBinding

블로그로 봅시다

1. 구현해보기

화면을 돌리면 죽어요

2. AAC

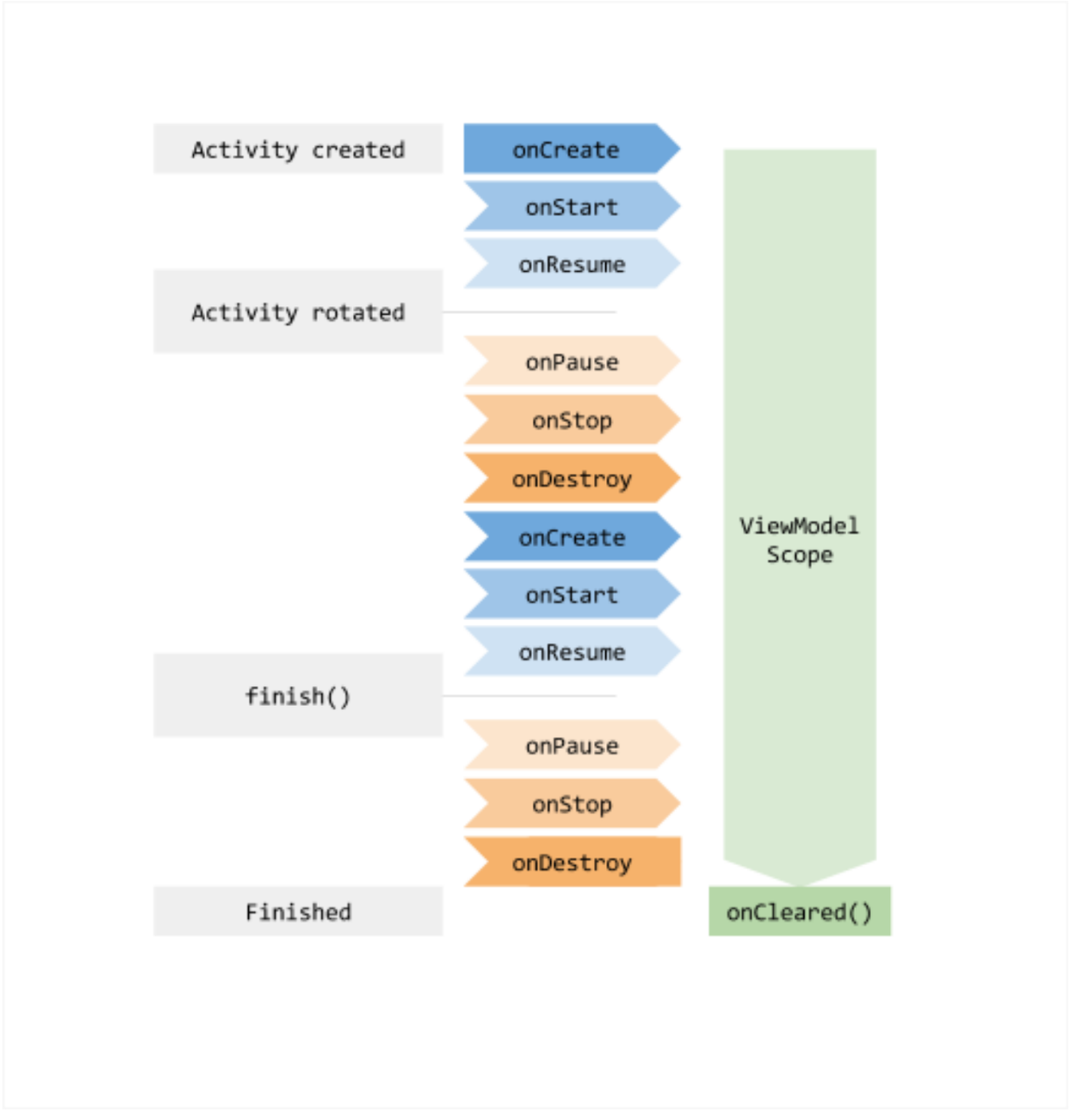
Android Architecture Components

2. AAC

LiveData

ViewModel

2-1. ViewModel



2-1. ViewModel

ViewModelProviders.of(this).get( )

2-2. LiveData

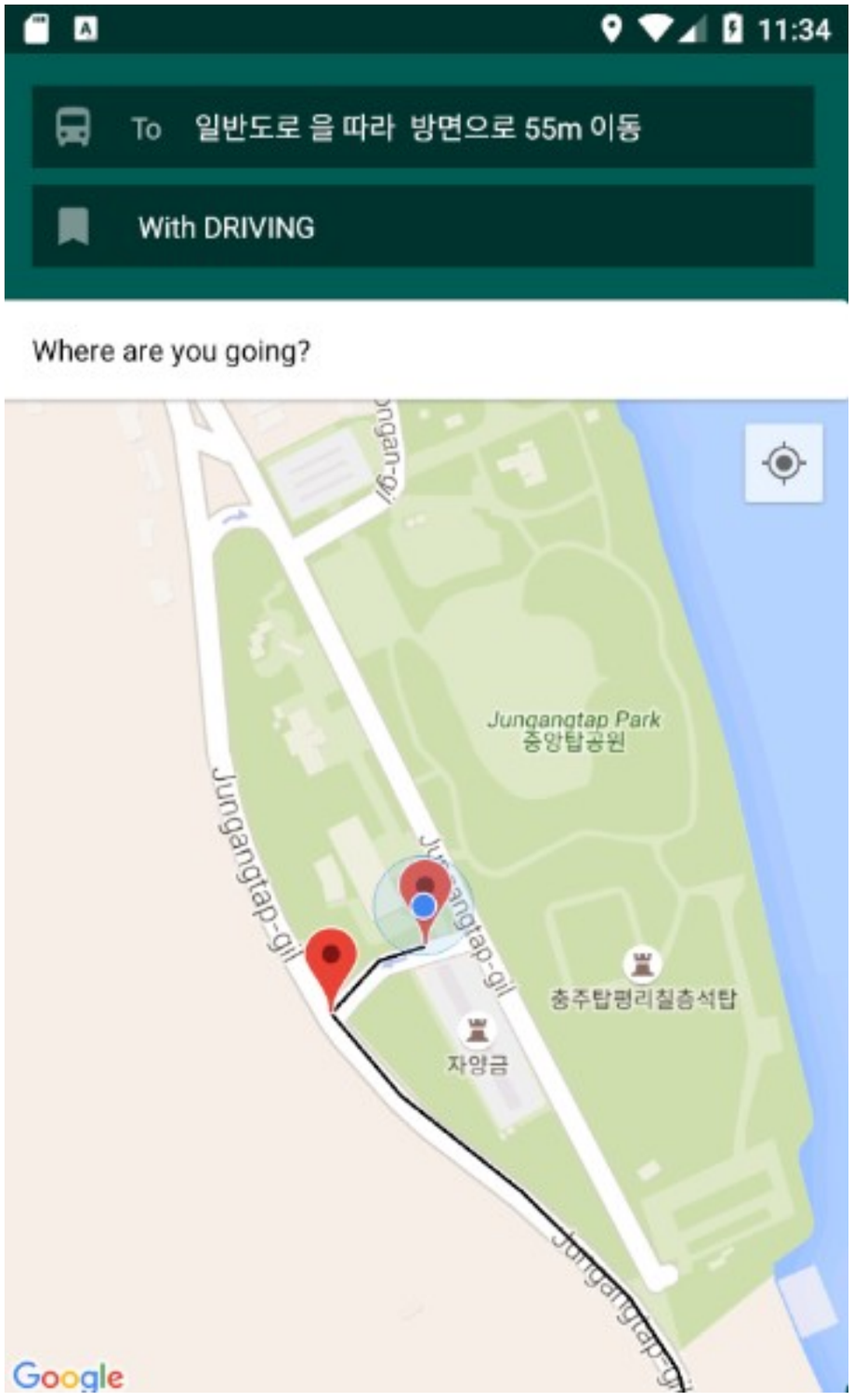
UI와 데이터의 상태 일치 가능

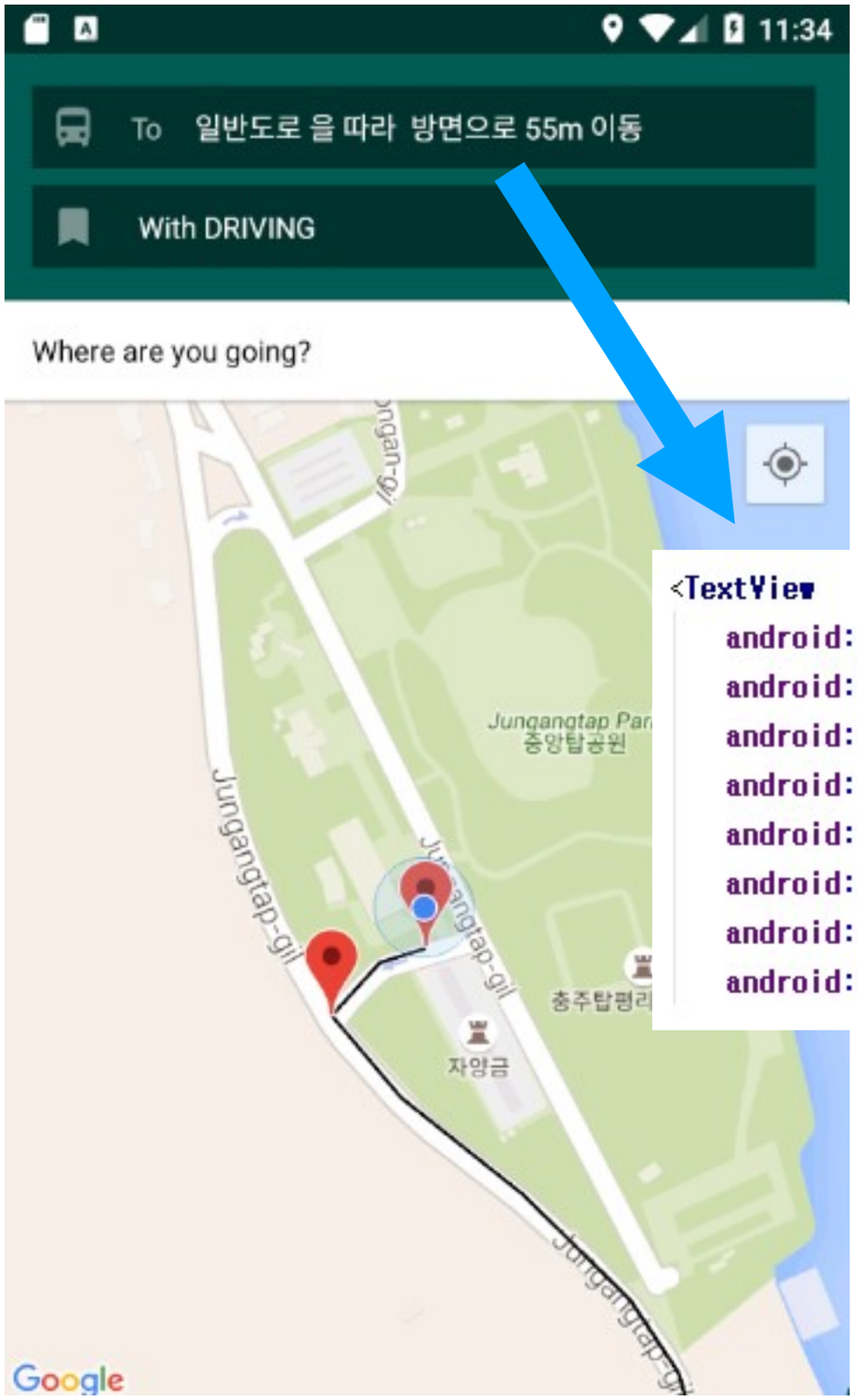
직접 생명주기를 핸들링할 필요 X

ViewModel 안에서 사용

3. MVVM with AAC

코드로 볼게요


11:34
To 일반도로 을 따라 방면으로 55m 이동
With DRIVING
Where are you going?
Jungangtap Park
중앙탑공원
Jungangtap-gil
충주탑평리칠층석탑
자양금
Google




```
<TextView
    android:id="@+id/navigation_trans_location_tv"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical"
    android:layout_marginEnd="16dp"
    android:layout_marginStart="16dp"
    android:textColor="#FFF"
    android:text="@{viewModel.transit}" />
```

```kotlin
class NavigationActivity : DataBindingActivity<ActivityNavigationBinding>(), OnMapReadyCallback {
    override fun getLayoutId(): Int = R.layout.activity_navigation

    val viewModel by lazy { ViewModelProviders.of(this)[NavigationViewModel::class.java] }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding.viewModel = viewModel

        viewModel.getNavigation(transType.toString(), lat, lng, desLat, desLng)
        val mapFragment = supportFragmentManager.findFragmentById(R.id.main_startTravel_mapView) as SupportMapFragment
        mapFragment.getMapAsync(this)

        viewModel.travelFinishEvent.observe(this, Observer {
            toast("Travel Is Finish!")
            val arriveIntent = Intent(this, ArriveActivity::class.java)
            arriveIntent.putExtra("placeid", placeId)
            startActivity(arriveIntent)
            finish()
        })
    }

    override fun onMapReady(map: GoogleMap) {

        val locationManager = getSystemService(Context.LOCATION_SERVICE) as LocationManager

        viewModel.polyLineEvent.observe(this, Observer {
            val d = PolyUtil.decode(viewModel.polyLine)
            map.addPolyline(PolylineOptions().addAll(d)
                    .width(5F))
            viewModel.direction.forEach {
                LatLng(it.lat, it.lng).let {
                    map.addMarker(MarkerOptions().position(it))
                }
            }
        })

        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 3000, 1F, object : LocationListener {
            override fun onLocationChanged(location: Location) {
                CameraUpdateFactory.newLatLngZoom(LatLng(location.latitude, location.longitude), 17F).let {
                    map.animateCamera(it)
                    viewModel.compareLocation(location.latitude, location.longitude)
                    }
                }
            })
        }
    }
}
```

```
class NavigationViewModel : ViewModel() {
    var direction = ArrayList<DirectionModel.Point>()
    var index = 0
    val changeTextLiveEvent = SingleLiveEvent<Any>()
    val travelFinishEvent = SingleLiveEvent<Any>()
    val transit = MutableLiveData<String>()
    val type = MutableLiveData<String>()
    val polyLineEvent = SingleLiveEvent<String>()
    var polyLine = ""

    fun getNavigation(transport: String, lat: Double, lng: Double, desLat: Double, desLng: Double) {
        getDirection(transport, lat, lng, desLat, desLng) {
            onSuccess = {
                direction = body()!!.points
                polyLine = body()!!.polyline.replace("\\", """\""")
                polyLineEvent.call()
                transit.value = direction[index].instruction
                type.value = direction[index + 1].let { " With ${it.mode}" }
            }
        }

    }

    fun compareLocation(lat: Double, lng: Double) {
        if (index < direction.size - 1) {
            val direction = direction[index + 1]
            direction.lat = String.format("%.5f", direction.lat).toDouble()
            direction.lng = String.format("%.5f", direction.lng).toDouble()

            if (direction.lat - 0.001 < lat.toFive() && lat.toFive() < direction.lat + 0.001) {
                if (direction.lng - 0.001 < lng.toFive() && lng.toFive() < direction.lng + 0.001) {
                    index++
                    transit.value = this.direction[index].instruction
                    type.value = this.direction[index + 1].let { " With ${it.mode}" }
                }
            }
        } else if (direction.size != 0) {
            travelFinishEvent.call()
        }
    }

    fun Double.toFive() = String.format("%.5f", this).toDouble()
}
```