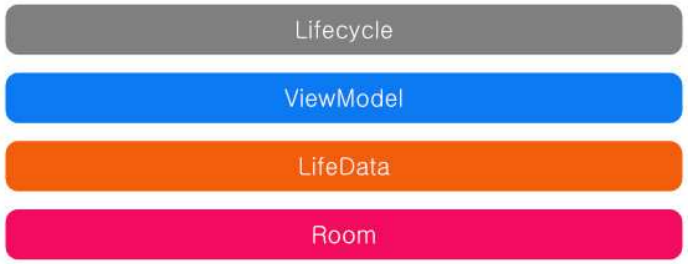
03-1. ViewModel

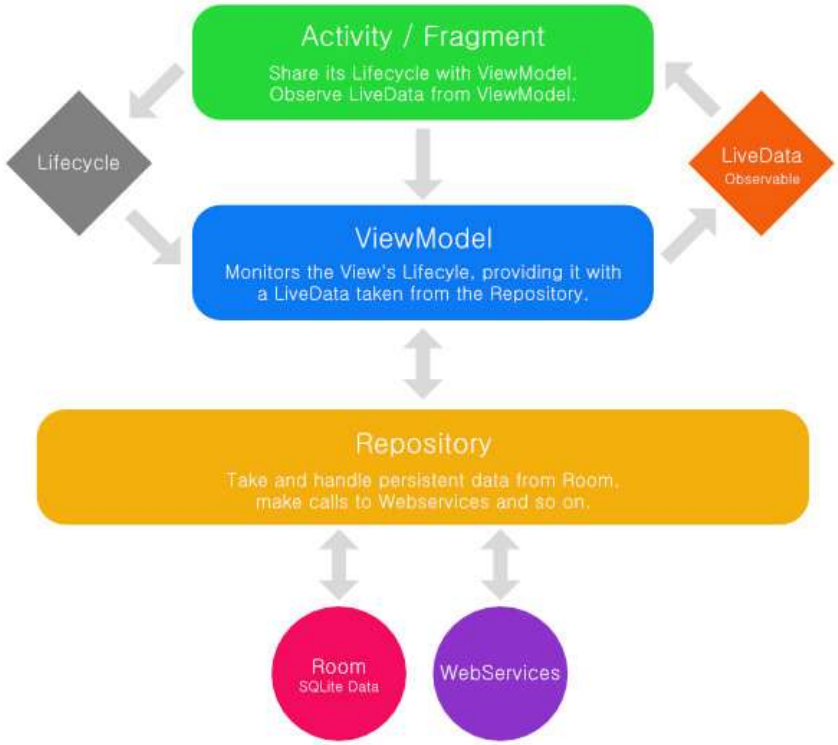
Android Architecture Component

Android Architecture Component

- 2017 Google I/O 에서 발표
- 2018 Google I/O 에서 발표된 JetPack 으로 통합
- Architecture Components 가 공개되기 전까지는 안드로이드 앱에서 특정 Architecture가 권장되지 않았다.
- MVP, MVC, MVVM, MVPP 등 다양한 아키텍처가 사용
- 안드로이드 앱을 위한 아키텍처를 정의하고 이를 구현하기 위한 라이브러리를 제공하고자 architecture components 을 공개
- Separation of concerns 을 목적으로 함.
- 그로인한 UI 와 모델이 분리



Android Architecture Component



Android Architecture Component

Activity/ Fragment : View
- UI를 구성하고 유저와 상호 작용. LifeCycle 변경을 감지.
- B/L 처리를 위해 ViewModel 이용.
- ViewModel에서 전달한 LiveData 의 내용을 화면에 출력.

ViewModel
- View와 Model을 분리시키기 위한 가교 역할.
- View의 요청에 의한 Repository 실행.
- Repository에서 발생한 데이터를 LiveData로 View에 전달.

Repository
- 데이터의 저장 및 획득이 주 목적
- Room : 데이터 영속화

ViewModel

- MVVM 의 핵심은 화면에서 B/L 을 분리해서 개발
- Activity, Fragment 에서 UI 를 처리하고 ViewModel 에서 B/L 을 처리
- Activity에서 onSaveInstanceState() 함수을 이용해 작성하던 Bundle 데이터 저장도 ViewModel로 대체가 가능

```
implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version'
```

ViewModel

- ViewModel 상속으로 작성

```
class MyViewModel1: ViewModel() {

}
```

ViewModel

- Activity 에서 ViewModel 이용
- ViewModelProvider 를 이용한 객체생성

```
val viewModel = ViewModelProvider(this).get(MyViewModel1::class.java)
```

- 직접 객체생성도 가능

```
val viewModel = MyViewModel1()
```

- ViewModelProvider 을 이용한 객체생성해야 Activity 상태 변화에 따른 ViewModel 이 재 생성되는 것을 막을 수 있다.

- Kotlin property delegate 이용

```
val viewModel: MyViewModel1 by viewModels()
```

Saving UI States

- Activity 의 Lifecycle 함수 이용
- 사용자 목록이나 비트맵과 같은 대용량일 가능성이 높은 데이터가 아니라, 직렬화했다가 다시 역직렬화할 수 있는 소량의 데이터에만 적합
- 성능의 문제점

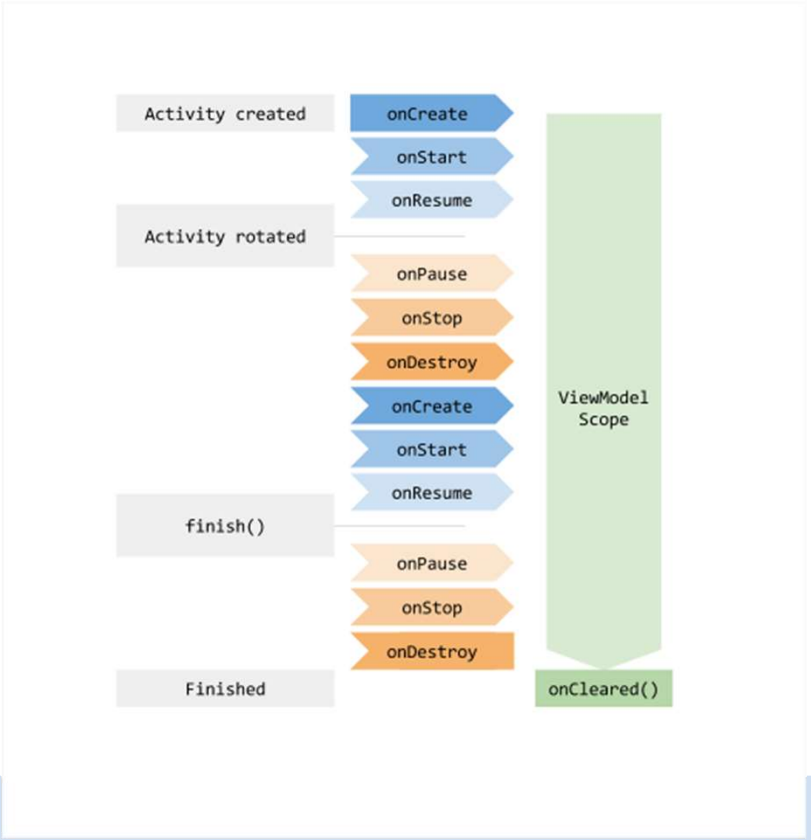
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val data1 = savedInstanceState?.getString("data1")
        val data2 = savedInstanceState?.getInt("data2")
    }

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putString("data1", "hello")
        outState.putInt("data2", 10)
    }
}
```

ViewModel Lifecycle

- Activity Scope 내에서 Singleton
- ViewModel 객체가 종료되는 것은 Activity가 finished 되거나 Fragment 가 detach 되었을 때



AndroidViewModel

- AndroidViewModel 은 ViewModel 의 서브 클래스

ViewModel()
AndroidViewModel(Application application)

- Application 객체를 이용해 Context 객체 획득

```
class MyApplicationViewModel(application: Application): AndroidViewModel(application) {
}
```

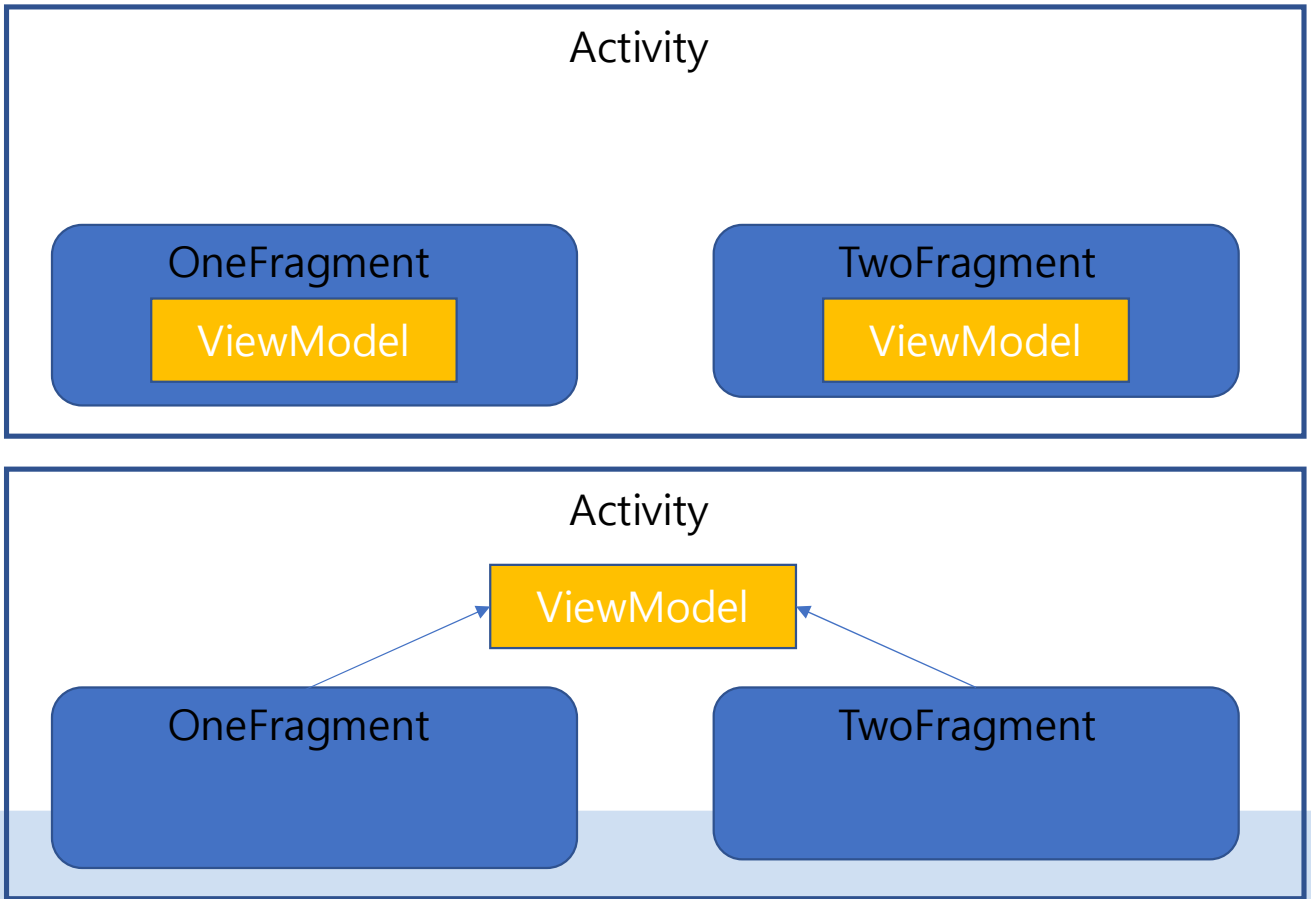
Fragment 에서 ViewModel 이용

- ViewModel 의 Owner 가 Fragment 가 된다.

```
class OneFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val model = ViewModelProvider(this).get(MyFragmentViewModel::class.java)
        //……
    }
}
```

Fragment 에서 ViewModel 이용

•Activity 내에 Fragment 가 여러 개 있는 경우 Fragment 간 ViewModel 공유 필요



Fragment 에서 ViewModel 이용

- ViewModel 의 Owner 를 Activity 로 변경

```
val model = ViewModelProvider(requireActivity()).get(MyFragmentViewModel::class.java)
```

- Kotlin property delegate 이용

```
val model: MyFragmentViewModel by activityViewModels()
```

```
implementation "androidx.fragment:fragment-ktx:$fragment_version"
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare