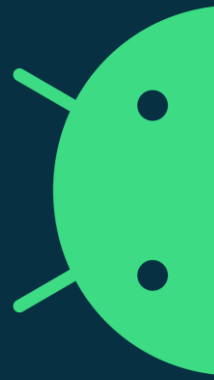
ViewModel
CMPS 312

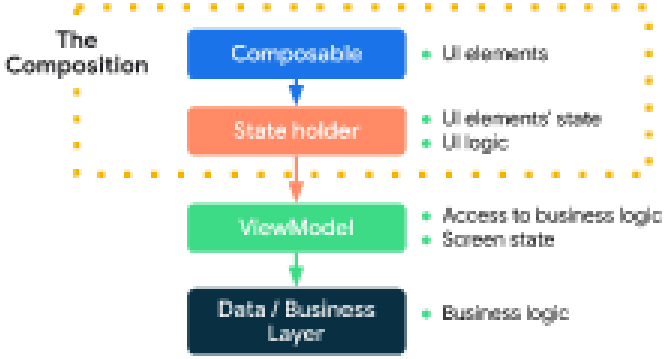


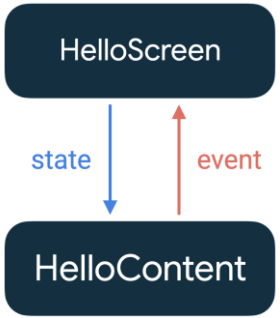
ViewModel overview | Android Developers

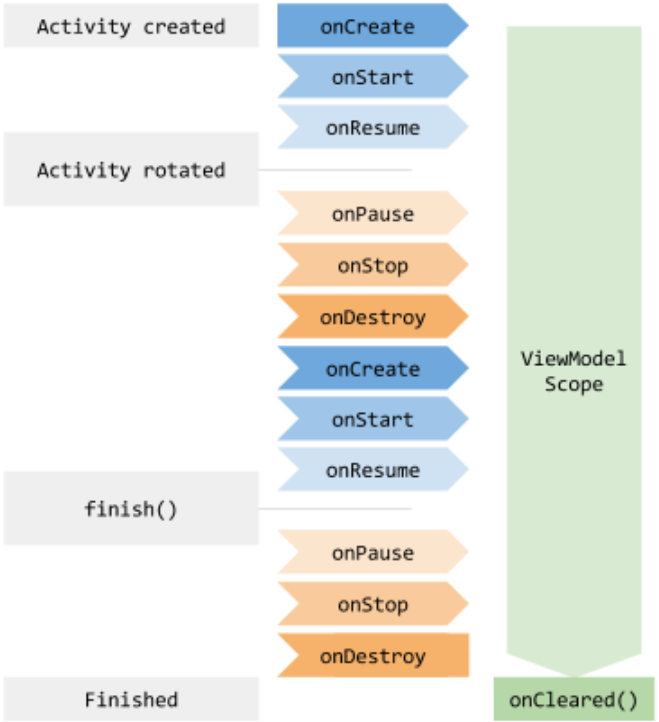
UI controllers such as activities and fragments are primarily intended to display UI data, react to user actions, or handle operating system communication, such as permission requests. Requiring UI controllers to also be responsible for loading data from a database or network adds bloat to the class.

- ViewModel is an Architecture Component.
- The purpose of the ViewModel is to acquire and keep the information that is necessary for an Activity.
- The Activity should be able to observe changes in the ViewModel.
- ViewModel's only responsibility is to manage the data for the UI. It should never access your view hierarchy or hold a reference back to the Activity.
    - A ViewModel must never reference a view, Lifecycle, or any class that may hold a reference to the activity context.
- ViewModel objects are automatically retained during configuration changes.

ViewModel overview | Android Developers







[ViewModel overview | Android Developers]

- You need to add required dependencies.
- Define your viewModel class by extending class ViewModel

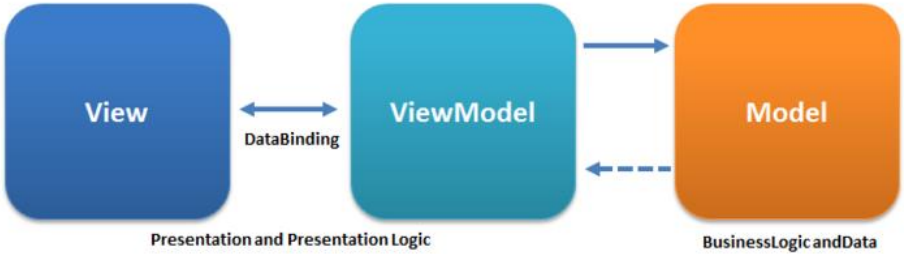
```
class MyViewModel : ViewModel() {
        /*...*/
}
```

- You usually request a ViewModel the first time the system calls an activity object's onCreate() method.
- Create a ViewModel the first time the system calls an activity's onCreate() method.
- Re-created activities receive the same MyViewModel instance created by the first activity.
- Use the 'by viewModels()' Kotlin property delegate the activity-ktx artifact

```
val model: MyViewModel by viewModels()
```

- If the activity is re-created, it receives the same MyViewModel instance that was created by the first activity.
- When the owner activity is finished, the framework calls the ViewModel objects's onCleared() method so that it can clean up resources.

MVVM (Model–view–viewmodel)



- Model–view–viewmodel (MVVM) is a software architectural pattern.
- It facilitates the separation of the development of the graphical user interface (the view) from the development of the business logic or back-end logic (the model) so that the view is not dependent on any specific model platform.
- The viewmodel of MVVM is a value converter, meaning
    - the viewmodel is responsible for exposing (converting) the data objects from the model in such a way that objects are easily managed and presented.
    - In this respect, the viewmodel is more model than view, and handles most if not all of the view's display logic.
    - viewmodel may implement a mediator pattern, organizing access to the back-end logic around the set of use cases supported by the view.

MVVM (Model–view–viewmodel)