MVC, MVP and MVVM Architecture Summary

Version:

- S.White – 15-08-2017

Sources:

- https://android.jlelse.eu/android-architecture-2f12e1c7d4db
- http://www.tachenov.name/2016/09/30/208/

Model View Controller

MVC design pattern splits an application into three main aspects: Model, View and Controller. It forces a separation of concerns, it means domain model and controller logic are decoupled from user interface (view). As a result maintenance and testing of the application become simpler and easier.

Model

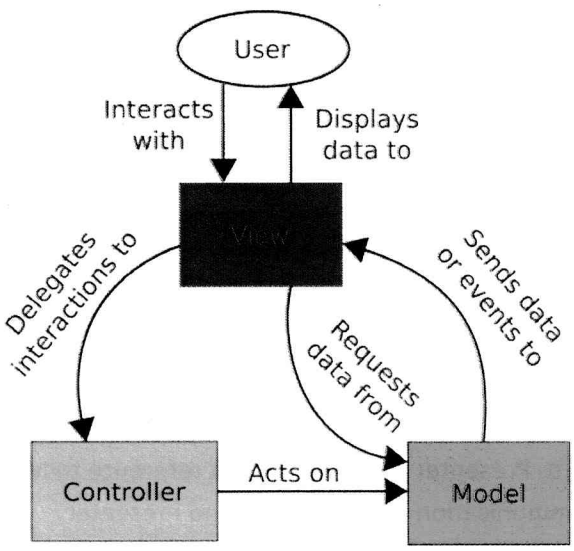
The Model represents a set of classes that describe the business logic i.e. business model as well as data access operations i.e. data model. It also defines business rules for data means how the data can be changed and manipulated.

View

The View represents the UI components. It is only responsible for displaying the data that is received from the controller as the result. This also transforms the model(s) into UI.

Controller

The Controller is responsible to process incoming requests. It receives input from users via the View, then process the user's data with the help of Model and passing the results back to the View. Typically, it acts as the coordinator between the View and the Model.



Model View Presenter

This pattern is similar to MVC pattern in which controller has been replaced by the presenter. This design pattern splits an application into three main aspects: Model, View and Presenter.

Model

The Model represents a set of classes that describes the business logic and data. It also defines business rules for data means how the data can be changed and manipulated.

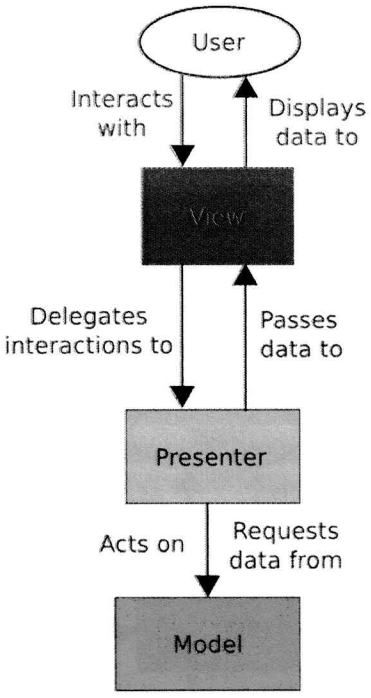
View

The View represents the UI components. It is only responsible for displaying the data that is received from the presenter as the result. This also transforms the model(s) into UI.

Presenter

The Presenter is responsible for handling all UI events on behalf of the view. This receive input from users via the View, then process the user's data with the help of Model and passing the results back to the View. Unlike view and controller, view and presenter are completely decoupled from each other's and communicate to each other's by an interface.

Also, presenter does not manage the incoming request traffic as controller.



Key Points about MVP Pattern

- User interacts with the View
- View has a reference to Presenter but View has not reference to Model
- Provides two way communication between View and Presenter
- There is 1-to-1 relationship between View and Presenter
  It means one View is mapped to only one Presenter

Model View ViewModel

MVVM stands for Model-View-ViewModel. This pattern supports two-way data binding between view and View model. This enables automatic propagation of changes, within the state of view model to the View. Typically, the view model uses the observer pattern to notify changes in the view model to model.

Model

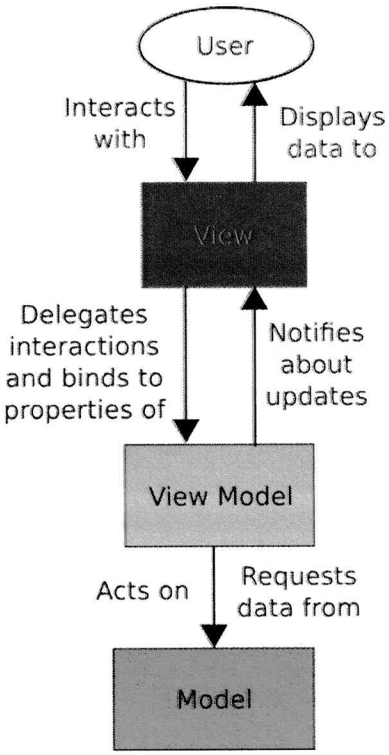
The Model represents a set of classes that describes the business logic and data. It also defines business rules for data means how the data can be changed and manipulated.

View

The View represents the UI components. It is only responsible for displaying the data that is received from the controller as the result. This also transforms the model(s) into UI.

View Model

The ViewModel is responsible for exposing methods, commands, and other properties that helps to maintain the state of the view, manipulate the model as the result of actions on the view, and trigger events in the view itself.



Key Points about MVVM Pattern:

- User interacts with the View
- View has a reference to ViewModel but ViewModel has no information about the View
- Supports two-way data binding between View and ViewModel
- There is many-to-1 relationship between View and ViewModel
  It means many View can be mapped to one ViewModel