

Model View Presenter



Jacek Modrakowski, Android Developer at  **scalac**

[illegible]


```
100
707 private class Download extends AsyncTask < Void, Void, Void > {
708
```

```
914
915 private class SendUserWeeklyUpdate extends AsyncTask < ArrayList < BasicNameValuePair > , Void, Void > {
916
```

```
832 private class SendPostBuzzComment extends AsyncTask < ArrayList < BasicNameValuePair > , Void, Void > {
833
```

```
875 private class SendUserDailyUpdate extends AsyncTask < ArrayList < BasicNameValuePair > , Void, Void > {
876
```


MVP

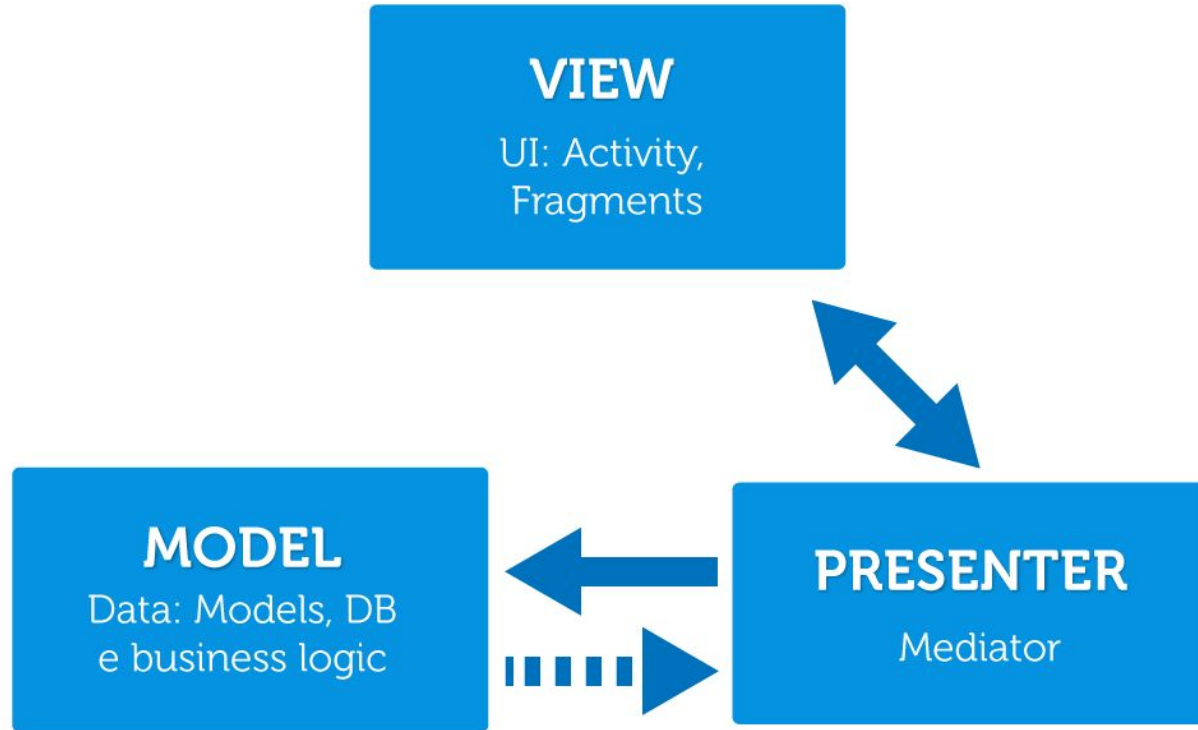
“A, komu to potrzebne?”

- Zorganizowany kod, znane podejście.
- Łatwiejszy rozwój projektu, nowe funkcjonalności, refactoring.
- Testowalny kod.
- Mniej bugów.

...MVC, MVVM, VIPER, Clean Architecture są lepsze od chaotycznego rozwijania aplikacji.

MVP

Model View Presenter



Source: (TinMegali, Medium)

MVP

Model View Presenter

VIEW

UI: Activity,
Fragments

MODEL

Data: Models, DB
e business logic

PRESENTER

Mediator



Source: (TinMegali, Medium)

```
public interface LoginView extends BaseView {  
  
    void continueAsNormalUser();  
  
    void continueAsPremiumUser();  
  
    void clearUserNameError();  
  
    void showUserNameError(String wrongNameMessage);  
  
    void clearUserPasswordError();  
  
    void showUserPasswordError(String wrongPasswordMessage);  
  
    void showProgress();  
  
    void hideProgress();  
  
    LoginView EMPTY = new LoginView() {...};  
}
```


MVP

Model View Presenter

VIEW

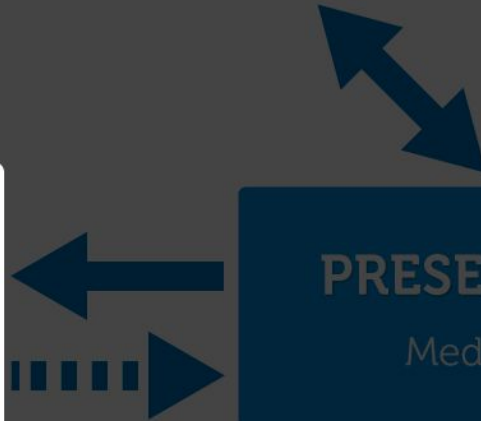
UI: Activity,
Fragments

MODEL

Data: Models, DB
e business logic

PRESENTER

Mediator



Source: (TinMegali, Medium)

```
public interface UserRepository {
```

```
    interface UserTypeAvailableListener {  
        void onUserTypeAvailable(boolean isPremium);  
    }
```

```
    void isPremiumUser(  
        @NonNull String userName,  
        @Nullable UserTypeAvailableListener userTypeAvailableListener);  
}
```

MVP

Model View Presenter

VIEW

UI: Activity,
Fragments

MODEL

Data: Models, DB
e business logic

PRESENTER

Mediator

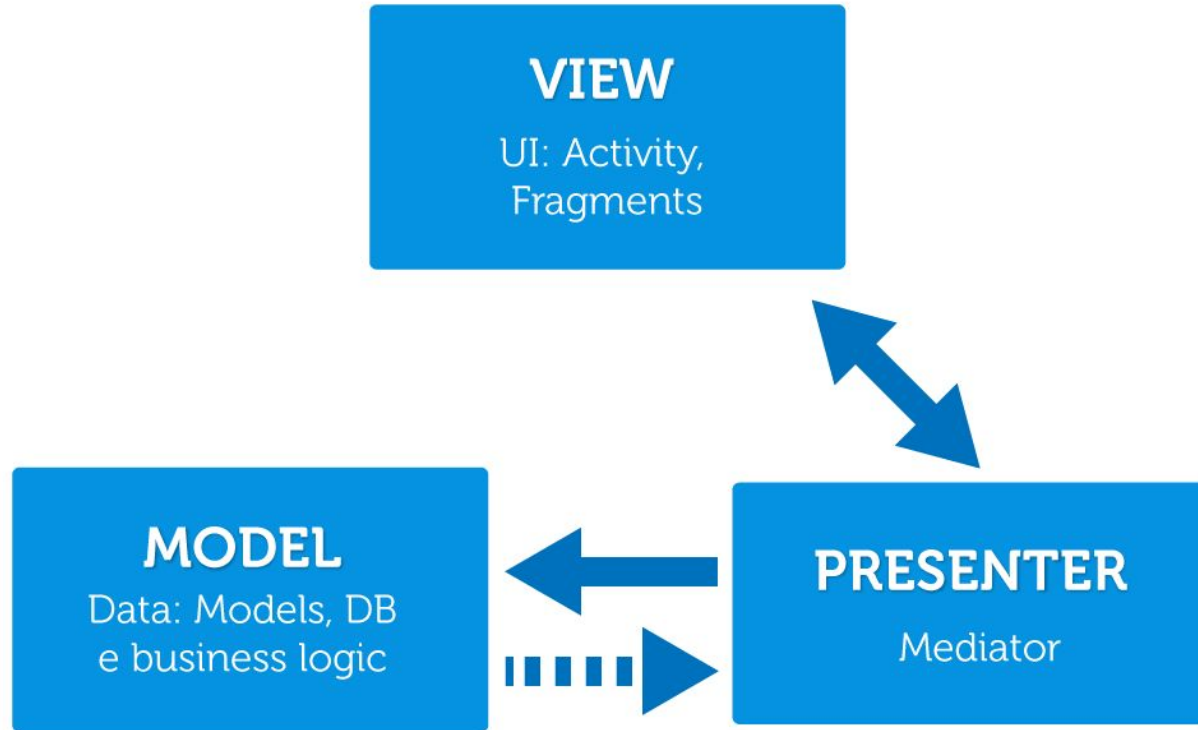


Source: (TinMegali, Medium)


```
public abstract class BasePresenter<ViewImplementation> {  
  
    private ViewImplementation view;  
  
    @NonNull protected ViewImplementation view() { return view; }  
  
    @CallSuper public void attach(@NonNull ViewImplementation view) { this.view = view; }  
  
    @CallSuper public void detach() { view = provideEmptyView(); }  
  
    @NonNull protected abstract ViewImplementation provideEmptyView();  
}
```

MVP

Model View Presenter



Source: (TinMegali, Medium)

View

{

```
loginPresenter.onLoginButtonClicked(userName);
```

Presenter

{

```
public void onLoginButtonClicked(@NonNull String userName) {  
    view().showProgress();  
  
    final UserTypeAvailableListener listener = (isPremium) -> {  
        view().hideProgress();  
  
        if (isPremium) {  
            view().continueAsPremiumUser();  
        } else {  
            view().continueAsNormalUser();  
        }  
    };  
  
    if (networkChecker.isOnline()) {  
        remoteUserRepository.isPremiumUser(userName, listener);  
    } else {  
        localUserRepository.isPremiumUser(userName, listener);  
    }  
}
```

Model

{

```
@Override  
public void isPremiumUser(  
    @NonNull final String userName,  
    @Nullable UserTypeAvailableListener userTypeAvailableListener) {  
  
    final boolean isPremium = userName.contains("premium");  
  
    if (userTypeAvailableListener != null) {  
        userTypeAvailableListener.onUserTypeAvailable(isPremium);  
    }  
}
```


Pora na odrobinę kodu.

jmodrako / jug_bydgoszcz_android_mvp



GitHub

Model

Warstwa **danych** (Repository, Dao, REST Api),
czasami logika biznesowa (rich model).

View

Rendering, przekazywanie akcji do Presentera, schowany za *interface'em*.
Implementowany najczęściej przez **Activity**, **Fragment** lub **View**.

Presenter

Kierownik imprezy, rozkazuje widokowi, pobiera
dane z modelu, **mediator między M <-> V**

PROBLEMY

- Podobnie jak kontroler w MVC tak samo prezenter lubi puchnąć (Massive View Controller).
- Threading, czy używać listenerów/callbacków, a może Observable z Rx, a może EventBus, ...
- Czy i jak zapisywać stan widoku?

Dziękuję za uwagę!

Q&A

More info

- <https://realm.io/news/eric-maxwell-mvc-mvp-and-mvvm-on-android/>
- <https://antonioleiva.com/mvp-android/>