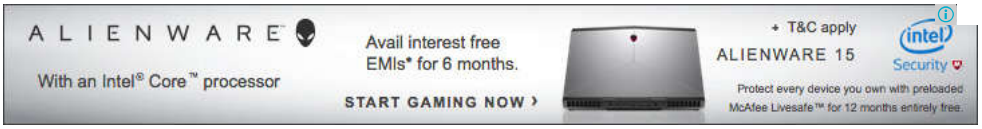
Tin Megali

MODEL VIEW PRESENTER (MVP) IN ANDROID, PART 1

4 DE MARCH DE 2016 | TINMEGALI@GMAIL.COM | 16 COMMENTS



Architecture Pattern is a fundamental part of computer science. It's the only way to maintain a project clean, expansible and testable. The patterns are recognized solutions that have been developed over the years and are considered industry standards. They're constantly evolving, and in the Android SDK, the reliable **Model View Controller** pattern is steadily being dropped for the **Model View Presenter.**

In the first part of this article we'll discuss the **main differences between MVC** (Model View Controller) **and MVP** (Model View Presenter)**, why MVC is being dropped**, **how MVP adjusts to Android** SDK and its **greatest advantages**.

- Model View Presenter (MVP) in Android, part 2
- Model View Presenter (MVP) in Android, part 3

The Android SDK

When we briefly analise the Android SDK, specially the relations between *layout x activity x data*, we have the impression that the *pattern* that best suits Android is the Model View Controller (MVC). However, when the project gains complexity, the separation of concerns offered by MVC isn't enough, specially to realize **unit tests**.

Yet, the way that the Android SDK was planned, allows us to use other types of architecture patterns, including no pattern whatsoever, the so called anti-patterns. Even though MVC is a reliable and well known solution, it's losing ground for its younger brother, the Model View Presenter (MVP), that offers some advantages, like a more well defined separation of concerns.

Should I use MVC or MVP in my project?

There isn't exactly a correct answer for this question. Some people will believe that MVC is the solution, others will stand with MVP and some will tend towards another solution, like MVVM for example. Each one of these approaches have advantages and disadvantages. Which means that the only way to answer the question is to understand the pros and cons of each solution, so you can make your choice wisely.

[expand title="Model-View-Controller (MVC) definition" tag="h5"]

Model–view–controller (MVC) is a software architectural pattern mostly (but not exclusively) for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

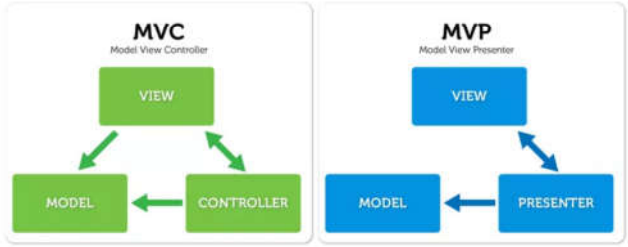
wikipedia

[/expand]

Model–view–presenter (MVP) is a derivation of the model–view–controller (MVC) architectural pattern, and is used mostly for building user interfaces.

In MVP the presenter assumes the functionality of the "middle-man". In MVP, all presentation logic is pushed to the presenter.

wikipedia



MVC vs MVP

Differences between MVC and MVP

Model View Presenter

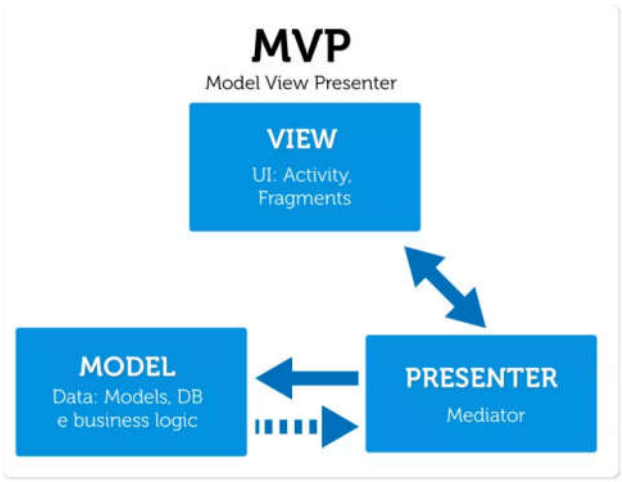
- View more separated from Model. The Presenter is the mediator between Model and View.
- Easier to create unit tests
- Generally there is a one to one mapping between View and Presenter, with the possibility to use multiple Presenters for complex Views

Model View Controller

- Controllers are behavior based and can share multiple views.
- View can communicate directly with Model

Model View Presenter (MVP) on Android

Android's separation of concerns isn't well defined. The Activities for example, have a too close relationship with data mechanisms. Although, for an application to become expandable, maintainable and testable, it's extremely important to create a deep separation of concerns and this is probably the biggest advantage that we'll get with the MVP adoption.



Layer relations in a Android's Model View Presenter pattern

Best way to implement the MVP pattern

This is actually a blurry topic. There are many interesting approaches towards the MVP and also a lot of different solutions to adapt it to Android. The way which the pattern is implemented, varies according to the role assumed by the Presenter. But independently of the choice, the MVP core should be preserved:

Presenter

The Presenter is responsible to act as ***the middle man between View and Model****. It retrieves data from the Model and returns it formatted to the View. But unlike the typical MVC, it also decides what happens when you interact with the View.*

View

The View, usually implemented by an Activity, will contain a reference to the presenter. The only thing that the view will do is to call a method from the Presenter every time there is an interface action.

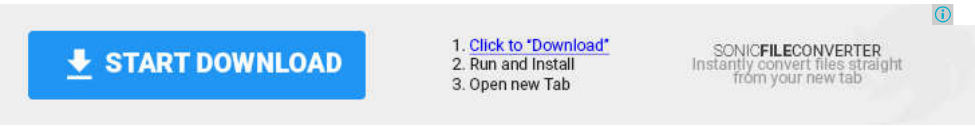
Model

In an application with a good layered architecture, this model would only be the gateway to the domain layer or business logic. See it as the provider of the data we want to display in the view.

These excellent definitions above were extracted from Antonio Leiva's article.

On the next article of this series, we'll implement the Model View Presenter pattern in Android. We'll follow a more conservative path, using only canonical code, without any libraries from outside the Android SDK. This approach will help to understand the different relations between the MVP's layers.

If you want to check the final **MVP library**, that was based on Dr. Douglas C. Schmidt vision, by all means do.



See you soon!

References:

- *http://antonioleiva.com/mvp-android/*
- *http://hannesdorfmann.com/android/mosby*
- *http://www.codeproject.com/Articles/288928/Differences-between-MVC-and-MVP-for-Beginners*
- *https://github.com/konmik/konmik.github.io/wiki/Introduction-to-Model-View-Presenter-on-Android*
- *https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter*

Also published on Medium.

Share

Related

Model View Presenter (MVP) in Android, Part 2
6 de March de 2016
In "Android"

Model View Presenter (MVP) in Android, Part 3
12 de March de 2016
In "Android"

Testing on Android part 1: Unit testing
8 de April de 2016
In "Android"

ANDROID ARCHITECTURE PATTERNS MODEL VIEW CONTROLLER MODEL VIEW PRESENTER

16 THOUGHTS ON "MODEL VIEW PRESENTER (MVP) IN ANDROID, PART 1"

Gopalan R C
9 DE MARCH DE 2016 AT 01:02
Good article. Eagerly waiting to read the next article.

★ **tinmegali@gmail.com**

9 DE MARCH DE 2016 AT 17:27

Hi Gopalan, the second part was already. You can check it out here: http://www.tinmegali.com/en/model-view-presenter-mvp-in-android-part-2/

mr_dsw

9 DE MARCH DE 2016 AT 04:51

thank you

vhsoni

9 DE MARCH DE 2016 AT 11:40

Good article. Waiting for the MVP implementation.

★ tinmegali@gmail.com

9 DE MARCH DE 2016 AT 18:07

Hi vhsoni, the second part was already. You can check it out here: http://www.tinmegali.com/en/model-view-presenter-mvp-in-android-part-2/

Esmores

10 DE MARCH DE 2016 AT 00:18

Great article and love the diagrams. Hoping to adopt this into my projects.

youngchan

13 DE MARCH DE 2016 AT 22:12

it's very good article.
can i tanslate your post to korean in my blog?

★ tinmegali@gmail.com

13 DE MARCH DE 2016 AT 23:15

Hello younchan!
You can translate the article, no problem. Could you be so kind to add a link to the original post?
Tks

youngchan

13 DE MARCH DE 2016 AT 23:27

sure, of course!

Ted

14 DE MARCH DE 2016 AT 07:16

Hello TINMEGALI, Thank you for your nice article.
i have translated part 1 in my blog.
this is the link:
http://tedcoder.com/2016/03/14/model-view-presenter-android-part-1/

★ tinmegali@gmail.com

14 DE MARCH DE 2016 AT 09:10

Nice job Ted. tks

Pingback: Model View Presenter (MVP) In Android | Android Bookmarks

DanV

22 DE MARCH DE 2016 AT 03:02

Hi,
Can you please add testing for the MainActivity and the GenericMVPActivity?

Thank you!

★ **tinmegali@gmail.com**

1 DE APRIL DE 2016 AT 11:18

Hey DanV. I'm writing a new article about testing. I'll talk about all kinds of tests, including Activity tests. It will be published this weekend.
best regards

---

youngchan

22 DE MARCH DE 2016 AT 07:27

Translated to Korean.

link : http://tiii.tistory.com/24
Thank you.

Pingback: Testing on Android part 1: Unit Testing