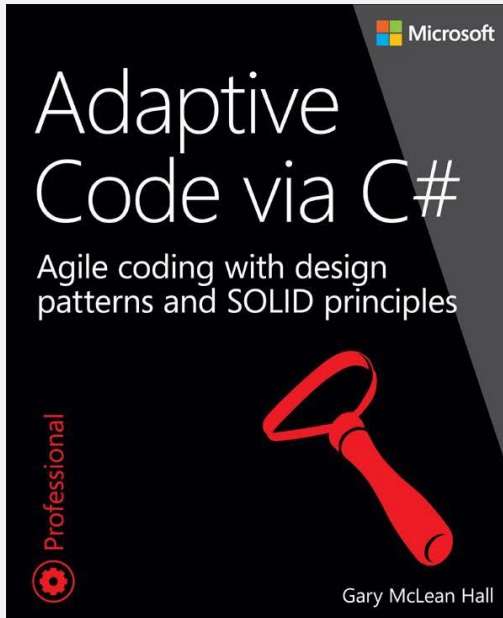


MVx in Android

How I Became an Android Developer



Official documentation



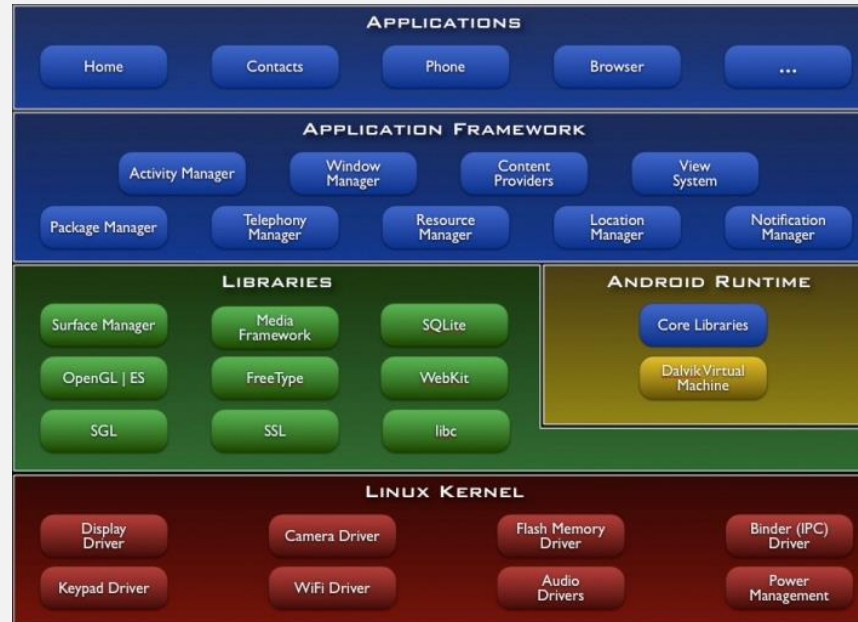
Blogs

After a Short While



Activities and Fragments of 500+ lines of code

Google for “Android Application Architecture”

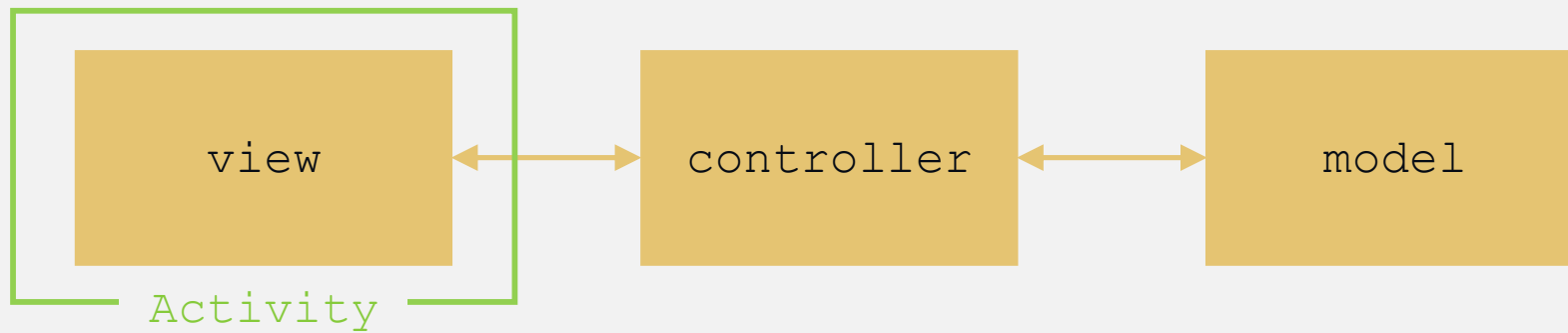
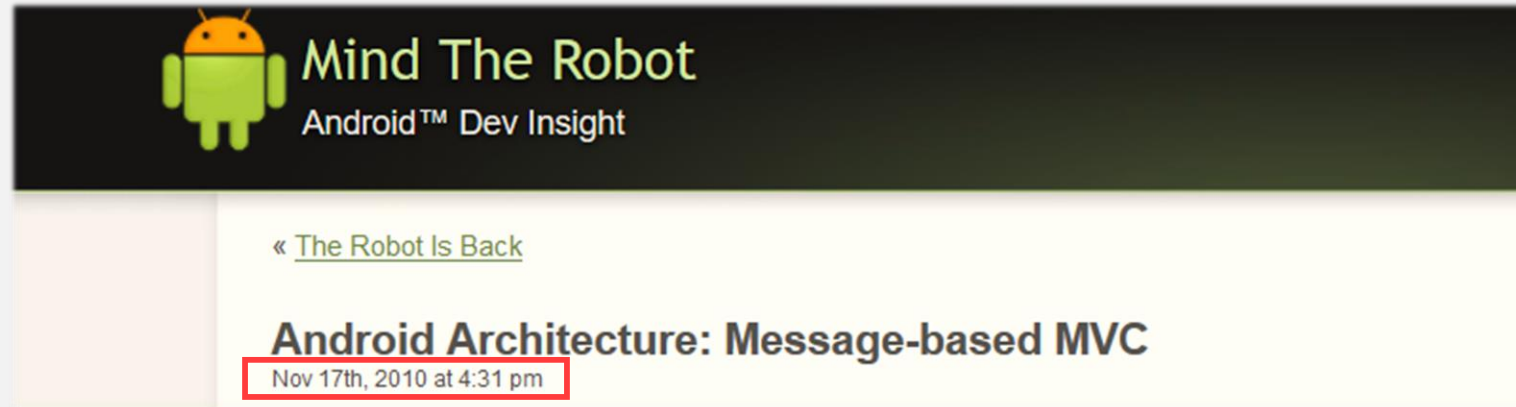


Exactly what I looked for... said no one ever.

**The best place to hide a dead body is page 2 of
Google search results**

I saw pages 2, 3, 4...

Android Architecture: Message-based MVC




Tremendous improvement...


... but still didn't feel quite right.

Android Architecture [in 10 parts]

Android Developer

Los Angeles Freelance Developer

[ABOUT](#) [CONTACT](#) [BLOG](#) [RSS](#) 



[← Not Everything is Polymorphic](#)
[Android Architecture: Structuring Network Calls, Part 1](#) →

To search, type and hit enter

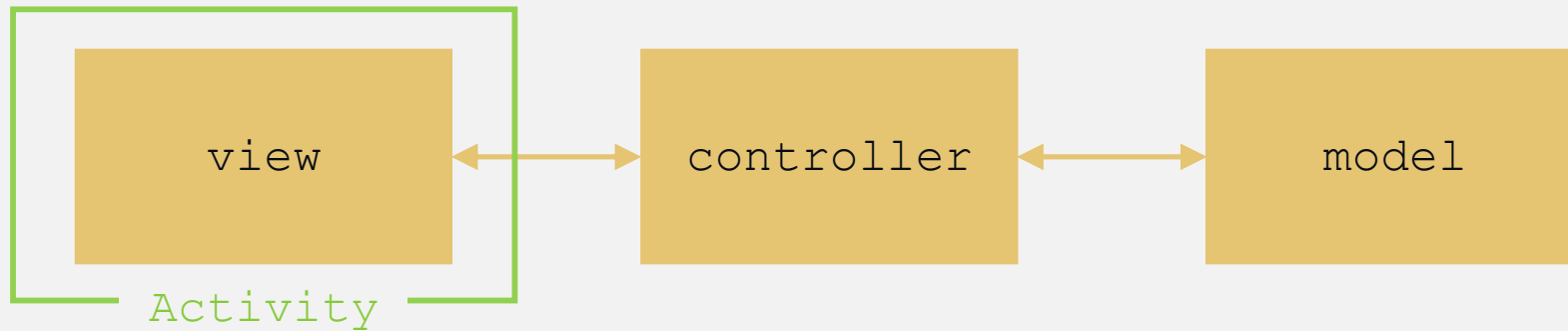
Android Architecture: Part 10, The Activity Revisited

July 9th, 2012 · 44 Comments · Android, OOP

RECENT POSTS

- [Introducing EssentialsLoader](#)

Joshua Musselwhite's Breakthrough



**The most important architectural breakthrough
in **Android** development EVER!**

The Main Benefit of MVx

Testability... ?

No need for MVx if you don't unit test then?

**Better testability is just a by-product of the
main MVx benefit**

**The main benefit of MVx is
decoupled UI logic**

UI logic in views seems to be the only common denominator of all MVx implementations

UI Logic Responsibilities

Render system output

Capture user interactions with UI elements and route them into the system
(no handling)

UI logic must be decoupled due to its special characteristics

UI Logic Characteristics

Detailed and accurate requirements (UI mockups)

Much higher rate of change in most cases

Unreadable - verbose, messy, hacky, etc.

Easiest to test manually

Hardest to test automatically

Why not decouple UI logic inside Activity?

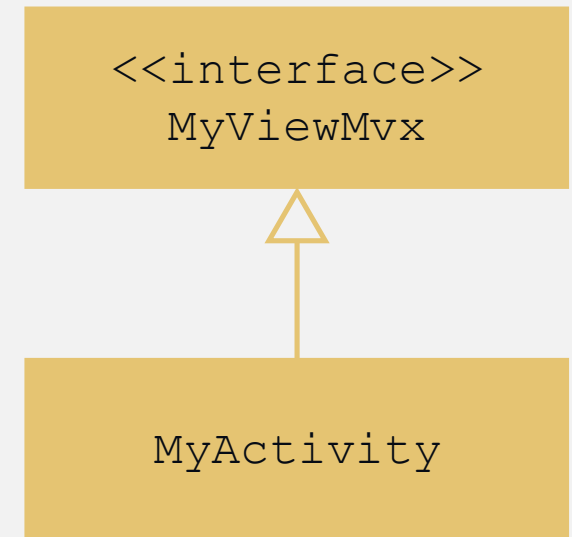
Example: Outsourcing

Detailed UI specification



+

UI logic should be
decoupled from the rest
of the application



**Would you integrate this class
into your app “as is”?**

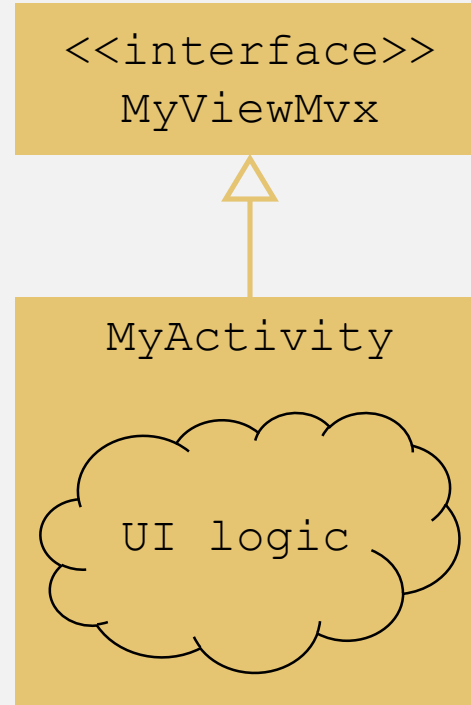
The Whole Picture

Life-cycle

Screens navigation

Runtime permissions

Loaders



Fragments

Dialogs

Dependency injection

More...

**Activity is God Object with many
responsibilities...**

**...therefore, it's impossible to decouple UI logic if it
resides inside Activity!**

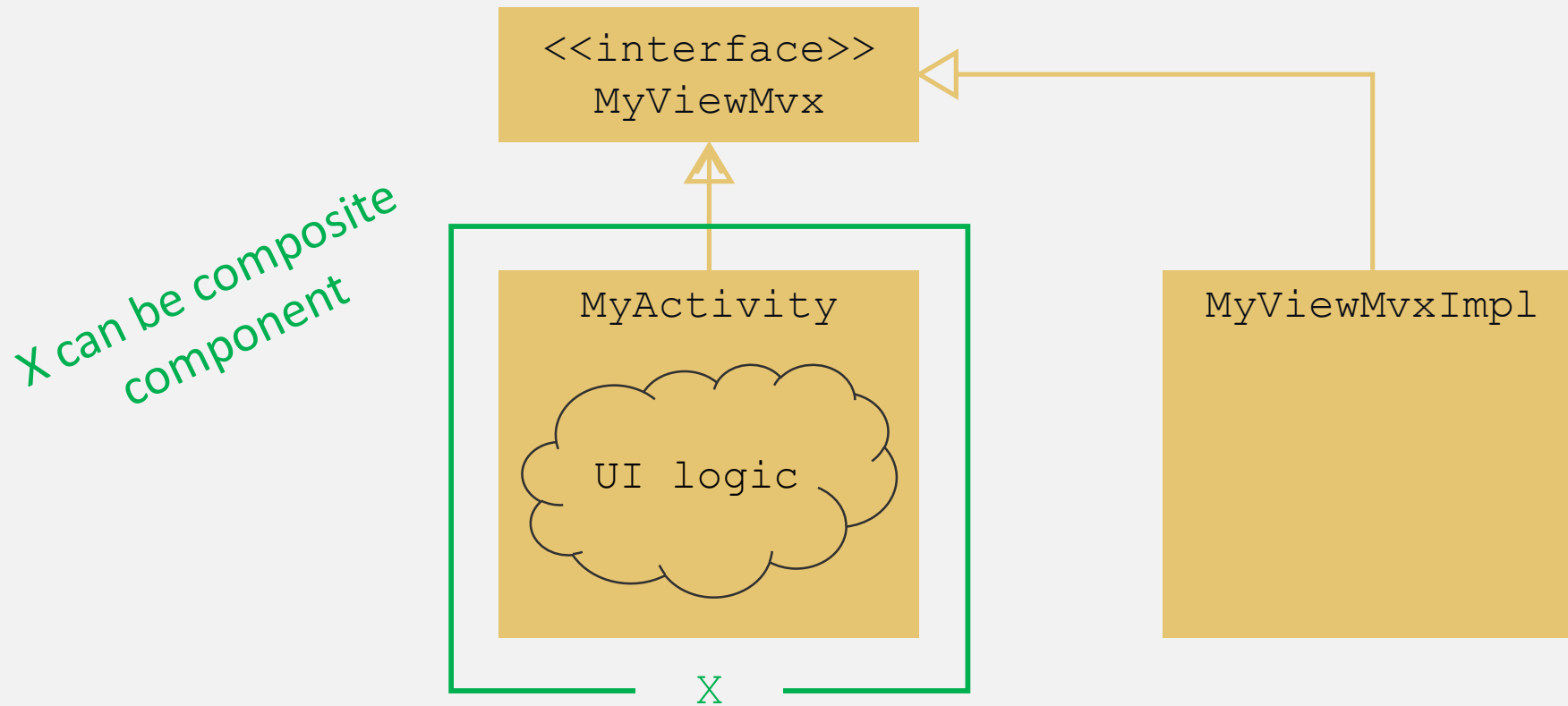
Fragments are God Objects too...

...therefore, it's impossible to decouple UI logic if it resides inside Fragments either!

**Can UI logic be decoupled outside of Activities
and Fragments?**

Yes!

Extract UI Logic Into Standalone Class



Composition instead of inheritance

From Theory to Practice

MVC or MVP or MVVM or ... ?

Letters don't matter!

I'm going to call it MVC!

**Is this another “New and Shiny Android
Architecture of the Month”?**

**2010:
Ivan proposed MVC on
Android**

**2014:
Vasiliy improved view
implementation and extended to
Fragments**



**2012:
Josh realized that Activity is
controller**

**The most mature and battle-tested architectural pattern for Android
development!**

MVx in Android

The main benefit of MVx is decoupled UI logic

Extracting UI logic from Activities and Fragments is the only way to make it truly decoupled

Activities and Fragments take on their natural role – controllers

MVC implementation you're going to see is the most mature and battle-tested architectural pattern for Android development