



A vibrant, circus-themed title card. The background features a sunburst pattern of yellow and orange rays. In the center, a large, ornate blue banner with a red and gold border contains the text "Charles' Android" in a white, serif font. Above the text is a small yellow tent icon. To the left, a cartoon acrobat with blonde hair and a red feather in her hair is suspended on a yellow rope, wearing a red and blue outfit. To the right, a blue cartoon bear wearing a red bow tie is balancing a yellow ball with two stars on its nose. The bear is standing on a large, decorated yellow and blue ball. Several yellow stars are scattered around the banner.

Charles' Android



MVVM

Start from the bottom



The need for architecture

- Lower maintenance costs
- Improve performance
- Improve security
- Improve stability
- Make unit test easier

principles of App Architecture



SOLID

- SRP
- OCP
- LSP
- ISP
- DIP

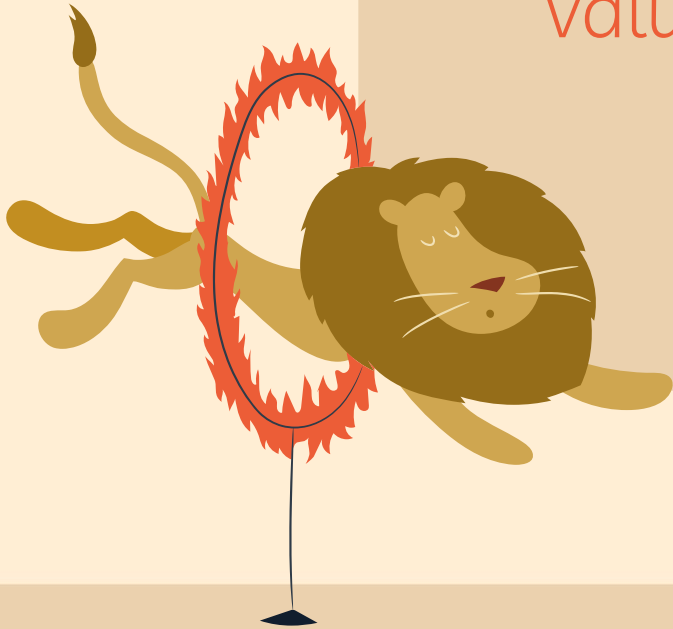


Clean Architecture

- Layering
- Separation
- Indepence

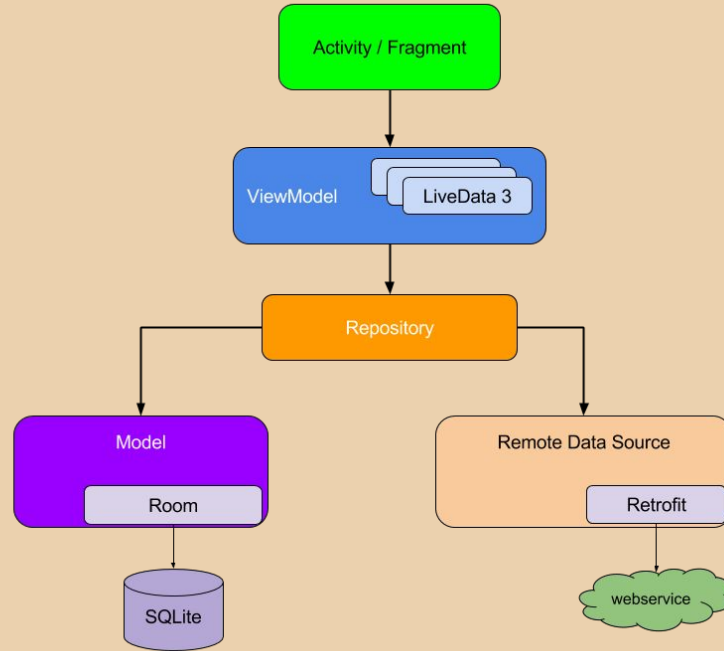
“Good design adds value faster than it adds cost.”

- Thomas C. Gale

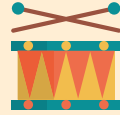


Recommended App Architecture

.....



What do you choose?



MVC

MVP

MVVM

MVC

Model

Provide data,
Everything excepting
Controller and View

Controller

Manipulate Events and Data

View

Display contents

MVC



MVP

Model

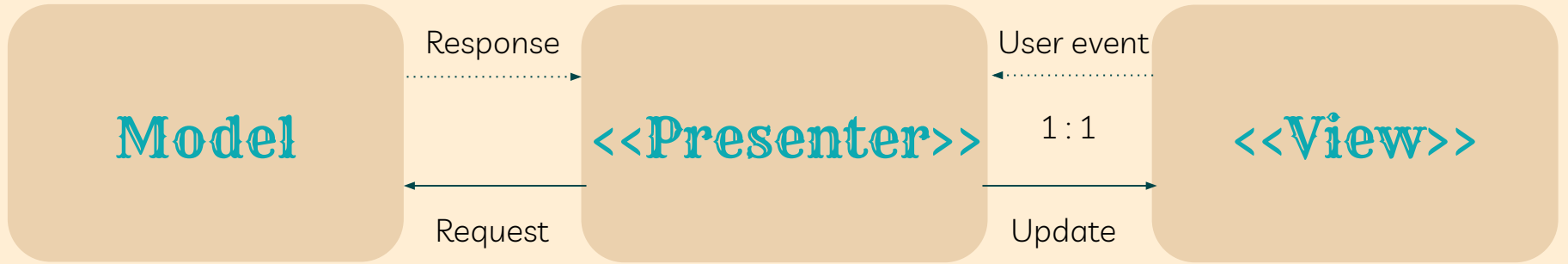
Presenter

View

Manipulate Events and Data
(a.k.a Supervising Controller or
Mediator)

Display contents
Passively

MVP



MVVM

Model

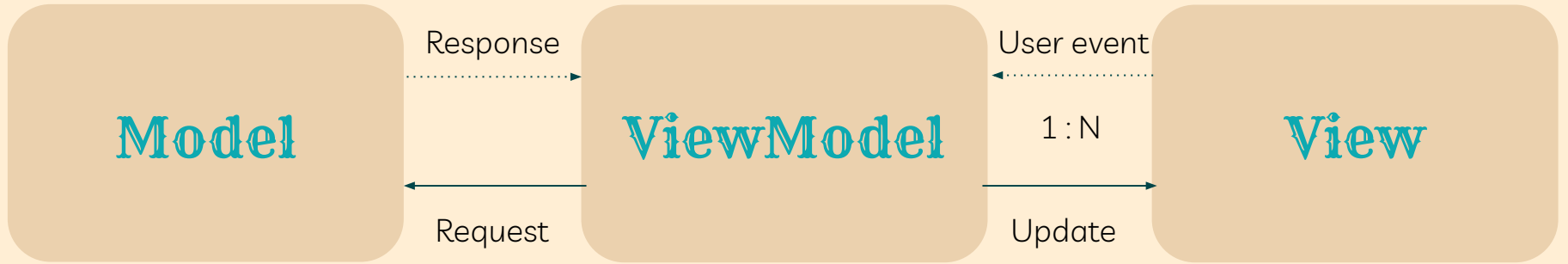
ViewModel

View

Manipulate Events and Data
**Independent from View
completely**

Display contents

MVVM



Trait of Architecture pattern

	Maintenance	Testable	Reusable
MVC	X	X	X
MVP	△	○	X
MVVM	○	○	○

Which class can be for ViewModel?

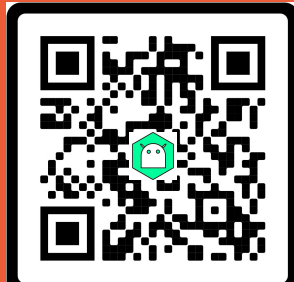
BaseObservable

- Included in data binding package.
- Can contain Context
- Used notifyChange() or notifyPropertyChanged() to update
- Updates views automatically using ObservableField

ViewModel

- Lifecycleaware
- **Prohibits to refer context but only App's context**
- Can be a problem when using Dagger2
- Has own it's scope
- Used with LiveData usually
- Provides saving UI States method
- Includes support for Kotlin coroutines

THANKS



SCAN ME

