[iOS]

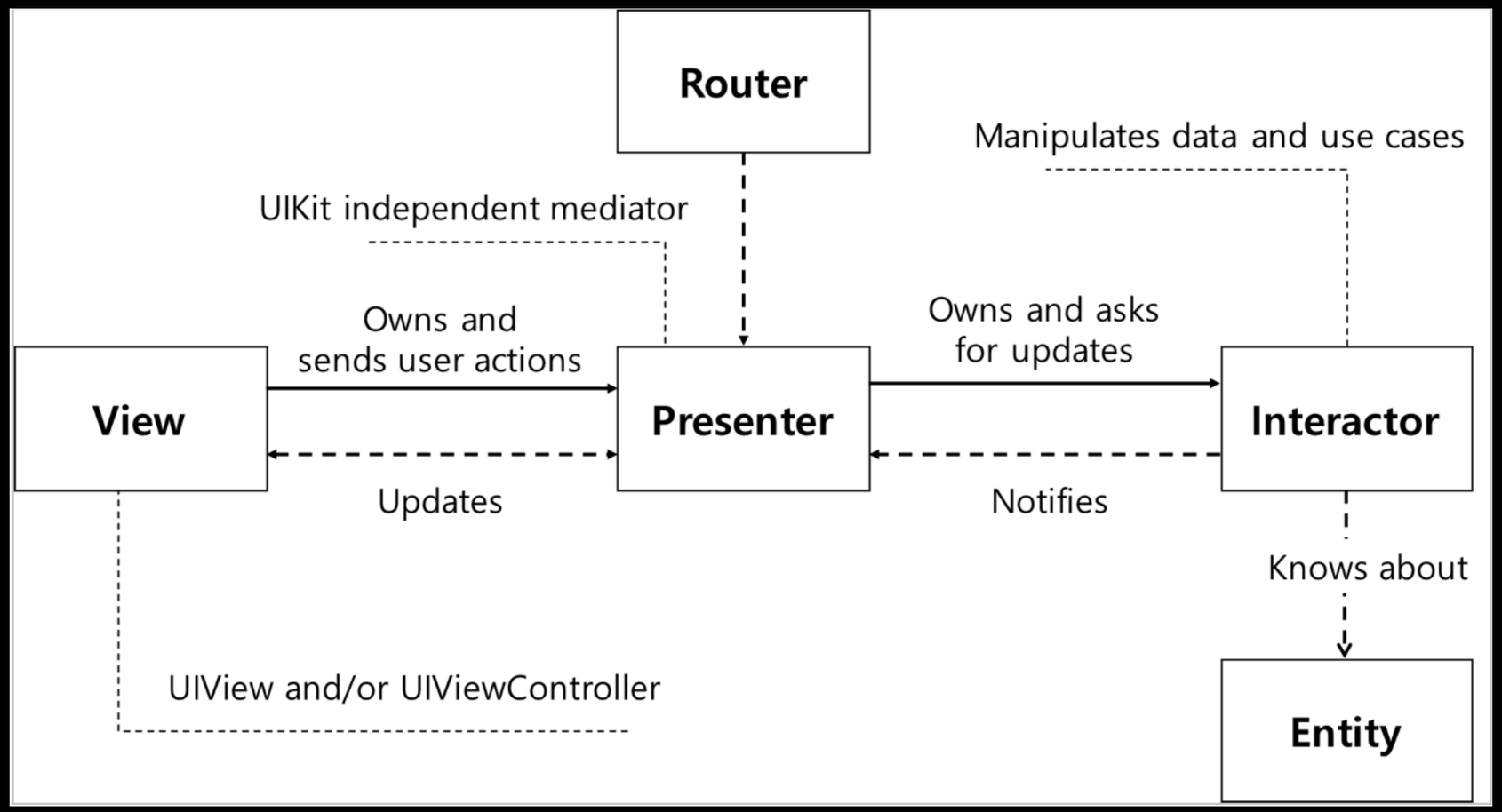
VIPER

Bill Kim(김정훈) | ibillkim@gmail.com

목차

VIPER

VIPER


Router
UIKit independent mediator
Manipulates data and use cases
Owns and
sends user actions
Owns and asks
for updates
View
Presenter
Interactor
Updates
Notifies
Knows about
UIView and/or UIViewController
Entity

VIPER

View : Presenter를 소유하여 요청을 받으며 액션을 전달하는 객체

Interactor : Entity 모델 객체를 조작하고 비지니스 로직 관리

Presenter : Interactor로부터 데이터를 가져오고 View로 데이터를 언제 보여줄지를 결정하는 객체

Entity : Model 객체

Router(Wireframe) : Persenter와 달리 View를 어떻게 보여줄지(seques)를 결정하는 객체

VIPER 장단점

장점 :

- 전형적인 MV(X)보다 역할 및 책임 분담이 명확하다
- 명확한 분리로 인한 유닛 테스트가 용이하다
- 한번 잘 설계하면 재활용이 가능하며 코드 이해가 쉽다
- 큰 프로젝트에서 여러명의 개발자가 공동 작업하기에 좋다

단점 :

- 많은 역할 분담으로 인하여 객체 및 코드가 많다
- 역활과 분리로 인하여 서로간의 인터페이스가 많아 유지보수 비용이 더 많이 든다
- 인터페이스 간의 응집도와 연관성이 매우 높다
- 초기 기획이 명확하지 않으면 적용하기가 어렵다

MV(X) 와의 비교

- Model(Data Interaction) 로직은 빈 데이터 구조로써 Enities와 함께 Interactor에 이동된다

- Controller/Presenter/View Model 이 Presenter로 이동하는 UI 표시는 책임을 지지만 데이터를 변경할 능력은 없다

- VIPER는 명시적으로 Router에 의해 결정된 네이게이션 책임을 해결한 첫 패턴이다

- VIPER는 책임에 대한 분리가 명확하기에 MV(X)와 달리 설계자에 따라서 효과가 극대화될 수 있다

VIPER 자동생성기

아래의 링크들을 통하여 VIPER 아키텍처 구조를 자동으로 생성할 수 있다

* VIPER gen :
https://github.com/pepibumur/viper-module-generator?utm_source=swifting.io&utm_medium=web&utm_campaign=blog%20post

* Generamba :
https://github.com/rambler-ios/Generamba?utm_source=swifting.io&utm_medium=web&utm_campaign=blog%20post

* Boa :
https://medium.com/@Supercharge/generating-viper-modules-with-boa-e8d9f090966b?utm_source=swifting.io&utm_medium=web&utm_campaign=blog%20post#.nry31apor

Sample Code : VIPER

```
import UIKit

struct Person { // Entity (usually more complex e.g. NSManagedObject)
    let firstName: String
    let lastName: String
}

struct GreetingData { // Transport data structure (not Entity)
    let greeting: String
    let subject: String
}

protocol GreetingProvider {
    func provideGreetingData()
}

protocol GreetingOutput: class {
    func receiveGreetingData(greetingData: GreetingData)
}

class GreetingInteractor : GreetingProvider {
    weak var output: GreetingOutput!

    func provideGreetingData() {
        let person = Person(firstName: "David", lastName: "Blaine") // usually comes from
        let subject = person.firstName + " " + person.lastName
        let greeting = GreetingData(greeting: "Hello", subject: subject)
        self.output.receiveGreetingData(greeting)
    }
}
```

Sample Code : VIPER

```
protocol GreetingViewEventHandler {
    func didTapShowGreetingButton()
}

protocol GreetingView: class {
    func setGreeting(greeting: String)
}


class GreetingPresenter : GreetingOutput, GreetingViewEventHandler {
    weak var view: GreetingView!
    var greetingProvider: GreetingProvider!

    func didTapShowGreetingButton() {
        self.greetingProvider.provideGreetingData()
    }

    func receiveGreetingData(greetingData: GreetingData) {
        let greeting = greetingData.greeting + " " + greetingData.subject
        self.view.setGreeting(greeting)
    }
}
```

Sample Code : VIPER

```
class GreetingViewController : UIViewController, GreetingView {
    var eventHandler: GreetingViewEventHandler!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
    }

    func didTapButton(button: UIButton) {
        self.eventHandler.didTapShowGreetingButton()
    }

    func setGreeting(greeting: String) {
        self.greetingLabel.text = greeting
    }

    // layout code goes here
}
// Assembling of VIPER module, without Router
let view = GreetingViewController()
let presenter = GreetingPresenter()
let interactor = GreetingInteractor()
view.eventHandler = presenter
presenter.view = view
presenter.greetingProvider = interactor
interactor.output = presenter
```

References

References

[1] iOS Architecture Patterns : http://labs.brandi.co.kr/2018/02/21/kimjh.html

[2] iOS Architecture Patterns : https://medium.com/ios-os-x-development/ios-architecture-patterns-ecba4c38de52

[3] (번역) 언제 VIPER를 사용할까? : https://beomy.tistory.com/43

[4] MVC, MVP, MVVM 비교 : https://blog.canapio.com/48

[5] From mvc to viper : https://www.slideshare.net/kprofic/from-mvc-to-viper?utm_source=swifting.io&utm_medium=web&utm_campaign=blog%20post

Thank you!