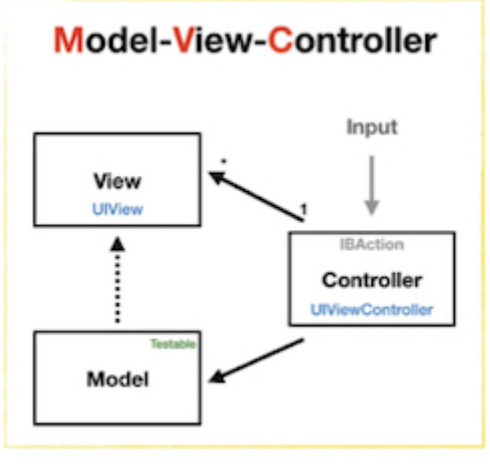
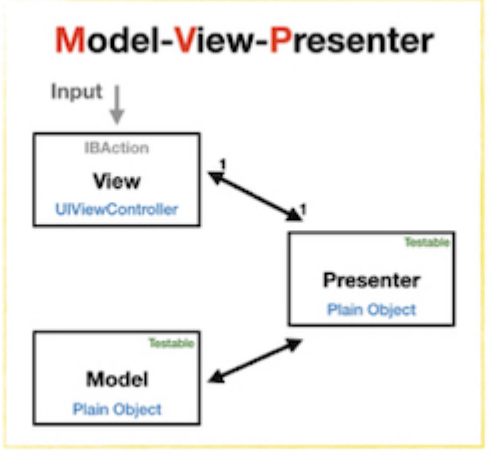
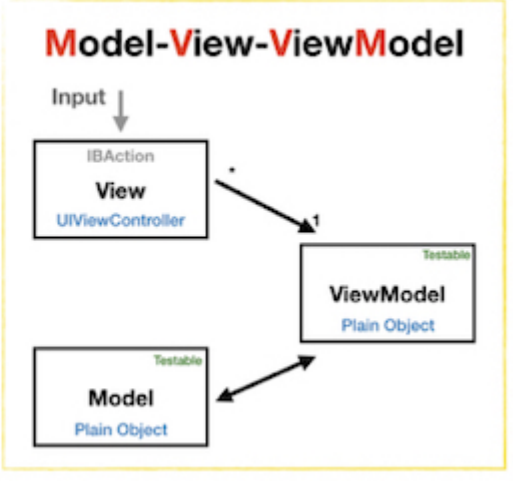
MVC, MVP 패턴, MVVM 패턴

작성 날짜	@2023년 11월 15일
구분	디자인패턴







MVC, MVP, MVVM

MVC, MVP, MVVM의 공통된 주제는 관심사에 따라 역할(레이어)를 분리하여 효율적인 동작을 수행하는 패턴을 의미한다.

역할을 분리하여 얻게되는 이점은 유지 보수와 개발 효율이 도음이 되기 때문이다.

MVC, MVP, MVVM는
Model과 View 사이의 결합도를 낮추고 기능 단위를 강화하고자 역할을 담당하는 것은 Controller, Presenter, View Model 이라는 이름을 가지고 있으며

MVC (Model + View + Controller)

Controller는 model과 view 사이의 커뮤니케이션할 수 있도록 해주는 패턴

Model
input 데이터를 처리 수행 및 비즈니스 로직을 나타내고 클래스의 집합을 가지는 역할
input 데이터가 바뀌고 조작되는지에 대한 비즈니스 규칙을 디자인하는데 동작
View
사용자에게 보여지는 화면(UI)
뷰는 컨트롤러가 폼 형태로 되돌려준 결과를 보여준다. 모델은 UI에 뷰로 변환 가능
Controller
사용자의 입력(input)을 받고 처리, model과 view 사이의 커뮤니케이션을 담당

동작 방식

1. 사용자의 INPUT이 Controller에 전달
2. Controller는 사용자의 INPUT를 확인하고, Model을 업데이트
3. Controller는 Model을 나타내줄 View를 선택
4. View는 Model을 이용하여 사용자에게 화면을 전달

** MVC에서 View가 업데이트 되는 방법*
- View가 Model을 이용하여 직접 업데이트 하는 방법
- Model에서 View에게 Notify 하여 업데이트 하는 방법
- View가 Polling으로 주기적으로 Model의 변경을 감지하여 업데이트 하는 방법

특징

Controller와 View는 1 : N의 관계

Controller는 View를 선택할 뿐 직접 업데이트 하지 않는다. (즉, View는 Controller를 알지 못한다.)

장점

단순하고 보편적으로 많이 사용되는 디자인패턴이다.

여러 개의 뷰를 모델에 빌드할 수 있다.

개발 속도를 병렬적으로 가속화 시킬 수 있다.

변경사항이 전체 모델에 영향을 주지 않는다.

데이터를 어떠한 형태의 가공 없이 리턴한다.

테스트에 용이하다.

단점

View와 Model 사이의 의존성이 높아 어플리케이션이 커질 수록 유지보수가 어렵다.

비즈니스 로직과 데이터가 분리되어 있기 때문에 코드 복제가 제한된다.

하나의 뷰에 대응되는 컨트롤러가 필요하다.

뷰가 수정되면 컨트롤러도 수정되어야 한다.

MVP (Model + View + Presenter)

Model과 View는 MVC 패턴과 동일하고, Controller 대신 Presenter가 존재

Model

input 데이터를 처리 수행 및 비즈니스 로직을 나타내고 클래스의 집합을 가지는 역할

input 데이터가 바뀌고 조작되는지에 대한 비즈니스 규칙을 디자인하는데 동작

View

UI의 입력 및 사용자에게 보여지는 화면(UI) 표시 하는 역할

Presenter

view에서 요청한 정보를 Model을 통해 가공하여 View에 전달해 주는 역할

model과 view 사이의 커뮤니케이션을 담당

MVP와 다른 점은 하나의 View에 하나의 Presenter가 매칭

*** View - Presenter의 추상적인 통신**

View - Presenter는 서로 통신하고자 인터페이스를 사용하여 추상적인 형태로 명령을 주고 받는다.

예로 들면 contract patter의 사용, Task Id 등이 있다. 그래서 View가 먼저 사용자의 입력을 통보하면 Presenter는 모델과의 통신을 통해 데이터의 갱신 및 스스로 판단 후 비즈니스 로직을 처리 후 View에게 변경 요청을 하게 된다.

View - Presenter가 추상적인 형태로 명령을 주 받기 때문에 View는 입력이나 이벤트를 넘겼을 때 어떠한 처리를 통해 들어오는지 여부는 알 수 없다. 즉, Presenter가 일련의 처리 (ex. 로딩바를 보여줘)를 요청하니 처리하는 것이라 생각하면 된다.

동작 방식

1. 사용자의 INPUT(Action)은 View를 통해 들어온다
2. view는 데이터를 Presenter에 요청
3. Presenter는 Model에게 데이터를 요청
4. Model은 Presenter에서 요청받은 데이터를 응답
5. Presenter는 View에게 데이터를 응답
6. View는 Presenter가 응답한 데이터를 이용하여 화면 표시 및 전달

특징

Presenter와 View는 1:1 관계

Presenter는 View와 Model의 인스턴스를 가지고 있어 둘을 연결하는 접착제 역할

장점

Presenter를 통해서만 데이터를 전달 받기 때문에 View와 Model의 의존성이 없다.

MVP는 세 가지의 다른 계층의 추상화하기 때문에 어플리케이션의 디버깅을 더 쉽게 만든다

코드 재사용성이 높아진다. 뷰를 컨트롤 하기 위해 여러개의 프레젠터를 가질 수 있다.

관심사 분리를 실행할 수 있다. 비즈니스 로직과 영속성 로직을 Activity와 Fragment 클래스에서 분리할 수 있다.

단점

View와 Presenter 사이의 의존성이 높아 어플리케이션이 커질 수록 유지보수가 어렵다.

MVVM (Model + View + View Model)

Model과 View은 다른 패턴과 동일

Model

input 데이터를 처리 수행 및 비즈니스 로직을 나타내고 클래스의 집합을 가지는 역할

input 데이터가 바뀌고 조작되는지에 대한 비즈니스 규칙을 디자인하는데 동작

View

UI의 입력 및 사용자에게 보여지는 화면(UI) 표시 하는 역할

View Model

View를 표현하기 위해 만든 View를 위한 Model로 View를 나타내 주기 위한 Model이자 View를 나타내기 위한 데이터 처리를 하는 부분

동작 방식

1. 사용자의 INPUT(Action)은 View를 통해 들어온다
2. View에 Action이 들어오면, Command 패턴으로 View Model에 Action을 전달
3. View Model은 Model에게 데이터를 요청
4. Model은 View Model에게 요청받은 데이터를 응답
5. View Model은 응답 받은 데이터를 가공하여 저장
6. View는 View Model과 Data Binding하여 화면 표시 및 전달

특징

View Model과 View는 1 : N 관계

MVVM 패턴은 Command 패턴과 Data Binding 두 가지 패턴을 사용하여 구현되었습니다.

Command 패턴과 Data Binding을 이용하여 View와 View Model 사이의 의존성을 없앴습니다.

장점

MVVM 패턴은 View와 Model 사이의 의존성이 없으며 Command 패턴과 Data Binding을 사용하여 View와 View Model 사이의 의존성 또한 없앤 디자인패턴이다.

각각의 부분은 독립적이기 때문에 모듈화 하여 개발할 수 있다.

단점

MVVM 패턴의 단점은 View Model의 설계가 쉽지 않다

Reference

https://devowen.com/457

https://beomy.tistory.com/43

https://gasungbilife.tistory.com/5

https://leehochang.tistory.com/24

꼬리 질문

command 패턴과 Data Binding이란?
service 계층과 mvc 패턴