`

Strato IT

디자인 패턴에 대한 연구

(MVC, MVP, MVVM)

팀장	신찬솔
팀원	김민철
	남희주
	심종현
	이규성

목 차

1. 서론

1-1. 패턴이란?

사람은 모두 다르기 때문에 코딩 스타일부터 프로젝트 구조까지 전부 제각각이다. 이런 상황이 발생한다면 협업을 할 때 서로 같은 언어를 쓰더라도 상대방이 어떤 구조로 만들었고, 어떤 스타일로 코딩하는지 파악하는데 엄청난 시간이 소비될 것이다. 또한, 협업이 아니더라도 이전에 했던 코드가 어떤 일을 하는지 나중에 다시 보았을 때, 처음부터 구조를 파악하는 자신의 모습을 심심치 않게 볼 수 있을 것이다. 하지만 코드를 패턴화 함으로 써 한눈에 파악할 수 있는 구조를 만들어 시간 절약 및 가독성을 높일 수 있다

1-2. 디자인 패턴이란?

현재 프로그래밍에 통용되는 디자인 패턴은 여러가지이다. 우리는 많은 패턴 중 MVC, MVP, MVVM 패턴에 대해 알아볼 것이다. 이 3가지 패턴은 객체 지향 프로그래밍을 설계할 때 자주 발생하는 문제들을 피하기 위해 사용되는 패턴이고, 유지보수와 확장성 및 관리를 원활하게 하기 위해 사용하는 목적임을 인지해야 한다.

2. 본론

2-1 MVC 패턴

2-1-1 정의

MVC는 Model, View, Controller의 약자로서 소프트웨어 디자인 패턴이다.

객체를 역할별로 분리하여 효율적인 코드를 할 수 있도록 만들었다.

모델(Model)

내부적으로 사용되는 데이터를 저장, 처리하는 역할을 하는 비즈니스 로직이다.

데이터를 가공하는 곳이라고 칭한다(DB 접근, 추가, 수정 등 일을 처리)

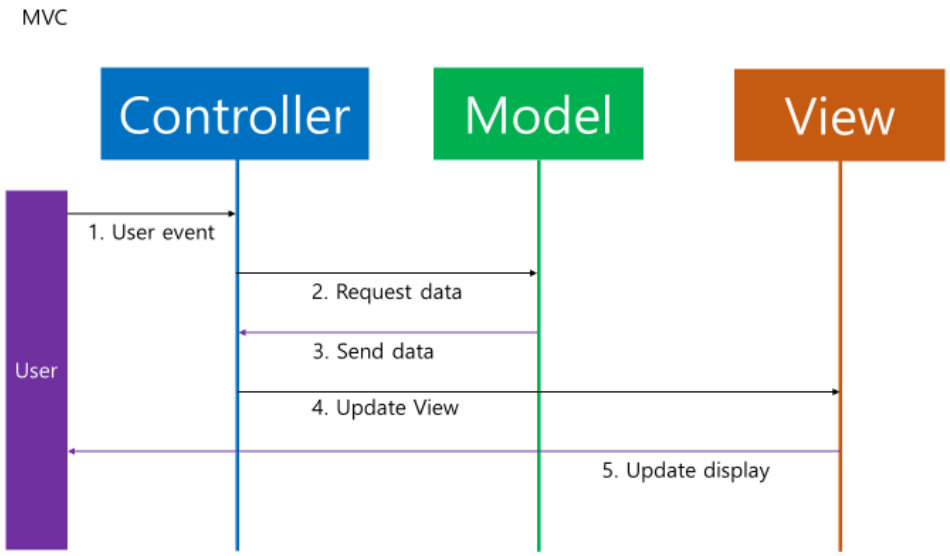
뷰(View)

주로 사용자에게 보여지는 인터페이스(UI) 영역을 뜻한다. Model에서 처리된 데이터를 받아서 사용자에게 보여주는 역할을 한다. Model과 종속 관계이다.

컨트롤러(Controller)

Model과 View의 Bridge 역할을 하기 때문에 Model이랑 View의 존재에 대해 알고 있어야 하며, 모델과 View의 변화를 감지하여 처리해주는 역할을 한다.

2-1-2 동작 순서(MVC)



[MVC - 동작원리]

1. User의 Action이 Controller에 Input

2. 관련 데이터를 Model에 요청

3. Model에서 관련 데이터를 Controller로 전송

4. Controller가 처리한 데이터를 View로 전송

5. View에서 User에게 화면을 출력해줌

2-1-3 장・단점(MVC)

<장점>

- 각각 패턴들을 구분하여 개발하므로 유지보수가 용이하고 유연성과 확정성이 높다.(유지보수 비용 절감)

- View Class가 분할되어 있어 디자이너와 개발자 간의 협업이 용이하다.

- 소규모 프로젝트에 적합하다.

<단점>

- model과 view의 의존성이 높아 어플리케이션이 커질수록 복잡하고 유지보수가 어렵다.

- Controller를 통해 View와 Model이 연결되어 프로젝트가 커진다면 수정 시 테스트가 힘들고 파악이 어렵다. (side-effect의 문제발생)

- 기본기능 설계를 위해 클래스들이 많이 필요하기 때문에 복잡하다.

2-2 MVP 패턴

2-2-1 정의

MVP 패턴은 Model, Veiw, Presenter의 앞글자를 따서 이름이 지어졌으며, MVC 패턴에서 파생된 디자인 패턴이다. 이 패턴의 핵심은 MVC 패턴의 단점인 View와 Model의 의존성을 해결하기 위하여 서로간의 상호작용을 Presenter에 위임해 서로 영향을 미치게 하지 않는 것이다.

MVP 패턴은 앞서 말했듯이 뷰, 프레젠터, 모델 세 부분으로 나뉜다.

모델(Model)

내부적으로 사용되는 데이터를 저장, 처리하는 역할을 하는 비즈니스 로직이다. View, Presenter 모두에게 의존적이지 않은 독립적이다.

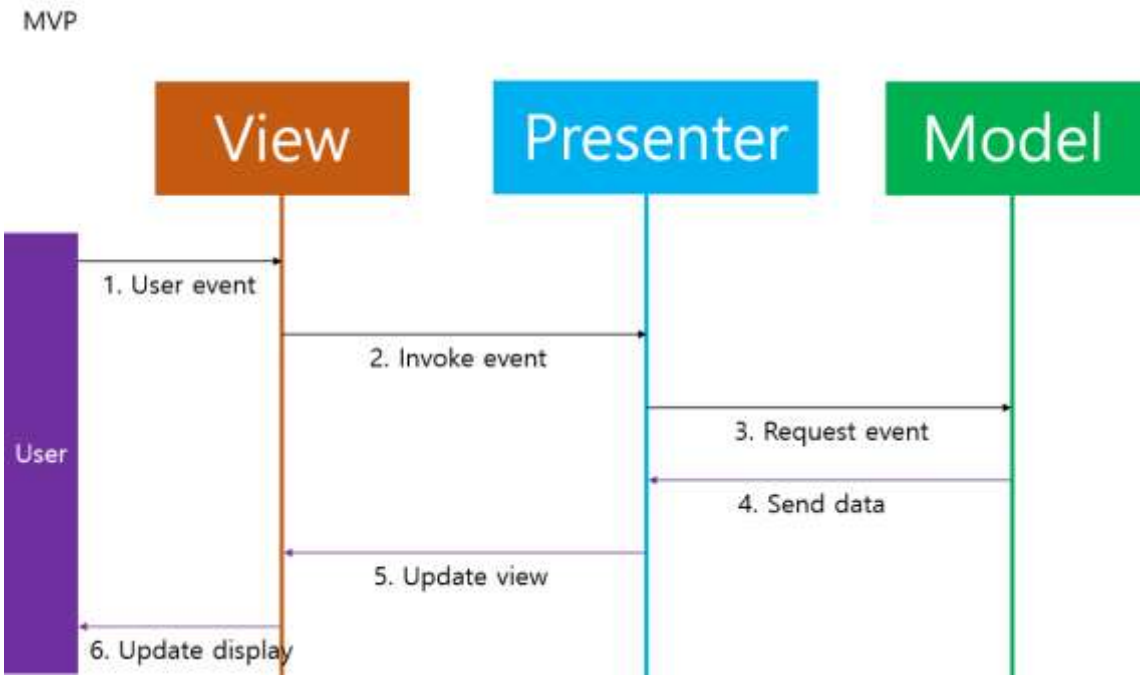
뷰(View)

주로 사용자에게 보여지는 인터페이스(UI) 영역을 뜻한다. 사용자의 Action을 감지하여 Presenter에 보내주며, Model에서 처리된 데이터를 Presenter를 통해 받아서 사용자에게 보여주는 역할을 한다. Presenter에게 의존적이다.

프레젠터(Presenter)

Model과 View 사이의 매개체 역할을 하여 View에서 발생한 Action을 받아 Model에게 전달하고, Model의 로직에서 나온 결과를 다시 View에게 전달하는 하는 매개체 역할을 한다. Model과 View 모두에게 의존적이다.

2-2-2 동작 순서(MVP)



[MVP-동작원리]

1. User의 Action이 View로 Input

2. View는 데이터를 Presenter에게 요청

3. Presenter는 Model에게 데이터 요청

4. Model은 요청받은 데이터를 Presenter로 전송

5. Presenter는 View에게 데이터 전송

6. View에서 User에게 화면을 출력해줌

***얼핏 보면 MVC 패턴과 차이가 없어보일 수 있으나, MVC에선 View만을 위한 로직이 없고 출력만을 담당한다. 그리고 Presenter(Controller)에 대한 호출의 흐름이 MVC에선 View안의 Action이 Controller를 호출한다는 점이 MVP와 차이점을 가진다.**

2-2-3 장・단점(MVP)

<장점>

- Model과 View간의 결합도를 낮춰준다.

- 새로운 기능을 추가하거나 수정이 필요할 때 관련된 부분만 수정하면 되기 때문에 확장성이 좋아진다.

- 테스트 코드를 작성하기 편리해진다.

<단점>

- View와 Presenter간의 의존성이 생긴다

- Interface를 지속적으로 작성함으로써 boilerplate 코드가 많아짐

- 소규모 프로젝트 혹은 소규모 팀에서는 부담이 될 수 있음

2-3 MVVM 패턴

2-3-1 정의

MVVM은 Model - View - View Model의 약자로서 이전의 MVP 패턴에서 Presenter를 View Model로 바꾼 디자인 패턴이다. Model은 데이터를 가지고 있고, View는 화면을 보여주는 부분을 담당하고 View Model은 View에서 발생하는 이벤트를 처리하는 부분이다. MVVM은 WPF, MS Sliverlight의 아키텍처이기도 하기 때문에 WPF에 최적화되어있으며 MVP 패턴과 비교했을 때 의존성이 없기 때문에 Model, View, View Model을 각각 개별로 모듈화하여 구현할 수 있다.

모델(Model)

내부적으로 사용되는 데이터를 저장, 처리하는 역할을 하는 비즈니스 로직이다. View, ViewModel 모두에게 의존적이지 않은 독립적이다.

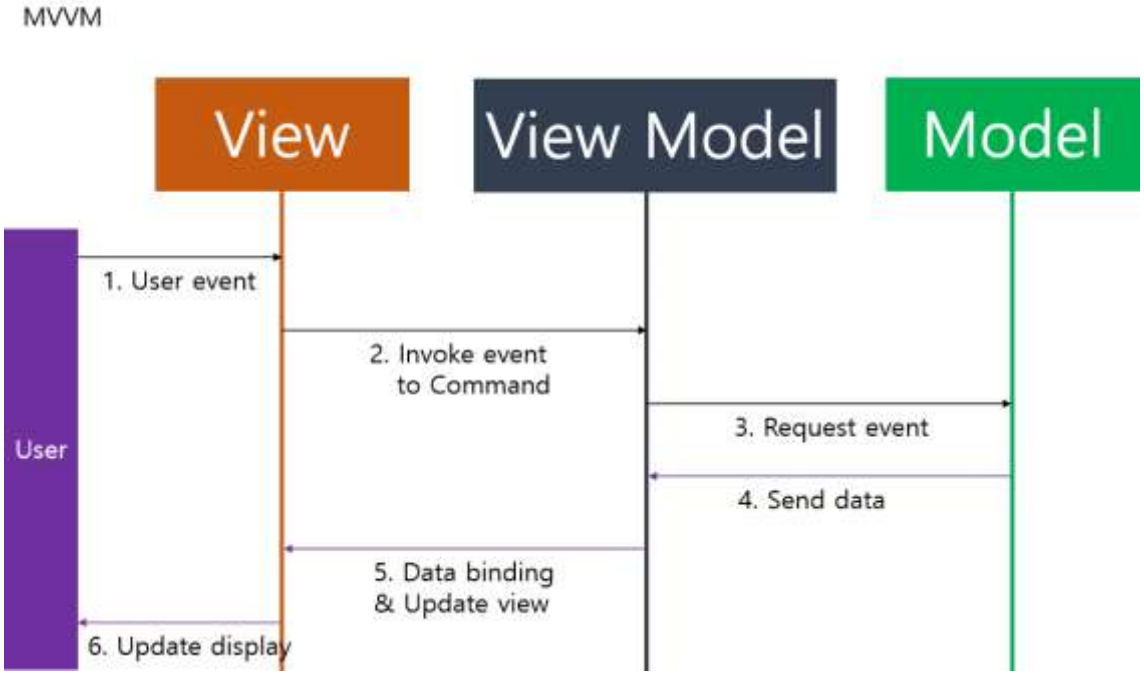
뷰(View)

주로 사용자에게 보여지는 인터페이스(UI) 영역을 뜻한다. 사용자의 Action을 감지하여 ViewModel로 보내주며, Model에서 처리된 데이터를 ViewModel을 통해 받아서 사용자에게 보여주는 역할을 한다

뷰모델(View Model)

Model과 View 사이의 매개체 역할을 하고 기존의 MVP 패턴의 Presenter의 단점을 개선하였다. View에서 발생한 Action을 Command 패턴으로 받아 Model에게 전달하고, Model에서 나온 결과를 Data Binding하여 View에 전달하는 역할을 한다.

2-3-2 동작 순서(MVVM)



[MVVM-동작원리]

1. User의 Action이 View에 Input

2. Action이 들어오면 Command 패턴으로 View Model에 Action을 전송

3. View Model은 Model에게 데이터를 요청

4. Model은 View Model에게 요청받은 데이터를 전송

5. View Model은 받은 데이터를 가공하여 저장

6. View는 View Model과 Data Binding하여 User에게 화면을 출력해줌

2-3-3 장・단점(MVVM)

<장점>

- 기존에 MVP와 비교했을 때, Model, View, View Model간 의존성이 없기 때문에 개별적인 모듈화가 가능

- EventHandler, Business Logic을 반복구현하는 번거로움을 줄임

- 대규모 프로젝트에 적합하다.

<단점>

- 다른 디자인 패턴들(MVC, MVP)에 비해 설계가 어렵다.

- 소규모 프로젝트에서는 오히려 다른 패턴이 더 효율적이다.

- 소규모 프로젝트에서는 Logic이 Main Contents보다 더 길어진다.

3.결론

위 3가지 모델에 대한 학습을 진행한 후 WPF를 사용하여 '매장관리 시스템'을 구현하면서 MVC, MVP, MVVM 각각의 패턴으로 프로젝트를 진행할 것이다. 그리고 이번 프로젝트를 통해 각 패턴에 대한 장점과 단점을 직접 경험해보고 특징에 대해 더 깊게 이해하면서 본 문서를 수정해나갈 예정이다. 이번 프로젝트가 가지는 기대효과로는, 앞으로 프로젝트를 진행하면서 어떤 패턴이 적합한지 알 수 있고, 능동적으로 패턴을 적용해 개발을 진행할 수 있을 것이라 생각한다.