XML, JSON, YAML

컴퓨터는 데이터를 전송할 때, 줄 바꿈이나 띄어쓰기 없이 한 줄로 된 데이터 만을 주고받음.

사람이 작성하고 전송할 때는 엔터와 탭이 모두 삭제된 한 줄의 문자열로 전송됨. (Minify)

테이블 형태와 같이 어떠한 구조를 갖는 데이터를 전송할 수 있도록 해주는 역할이 가장 크다.

XML, JSON, YAML은 모두 데이터를 구조화하고 저장하는 포맷이다.

XML

xml은 HTML과 같이 "태그"라는 형식을 사용

각각의 태그명이 각각의 항목명이 되고 그 사이에 내용이 들어가는 형식 ex) <age>44</age>

스키마 라는 것을 통해 검증 가능.

단점

작성하기 번거러움.

태그들을 열고 닫아야 하니 글자수가 늘어나 장황해 보임.

```xml
<person>
    <name>홍길동</name>
    <age>30</age>
    <city>서울</city>
</person>
```

JSON

자바스크립트의 객체 표기법인 JSON은 보다 간결한 형태로 구조화된 정도를 표시.

눈에 잘들어오고 코드 양도 xml에 비해 적음.

간결하고 작성하기 쉬워 xml을 대체해 나가고 있다.

단점

문법 오류에 취약함. 콤마 하나 잘못치면 문서 전체가 해석 불가가 됨.

주석을 달 수 없음.

```json
{
    "person": {
        "name": "홍길동",
        "age": 30,
        "city": "서울"
    }
}
```

이러한 특징들을 고려해서 안정성이 요구되는 곳에는 XML이 가벼움을 중시하는 곳에는 JSON이 활용된다.

YAML

yaml은 데이터를 한 줄로 실어 보내는 것이 아니라, 사람이 보기 좋게 작성하는 데 목적을 둔다.

파이썬처럼 줄 바꿈과 태그가 필수 요소이다. 이를 어기면 정보가 파괴되기 때문에 yaml 문서는 minify를 하지 않는다.

주석 사용 가능하고, 특히 편리한 점은 상속을 사용해서 여러 데이터들을 효율적으로 작성 가능.

이처럼 쓰고 있는 사람의 편의를 우선시하는 형식이기 때문에 도커컴포즈, 스프링 등의 설정 파일에 많이 사용된다.

```yaml
person:
  name: 홍길동
  age: 30
  city: 서울
```

MVC, VMP, MVVM

MVC, MVP, MVVM은 소프트웨어 개발에서 사용되는 세 가지 주요 아키텍처 패턴이다.

MVC: (Model View Controller), MVP: (Model View Presenter), MVVM: (Model View ViewModel)

MVC (Model View Controller)

구성요소

모델(Model): 데이터와 비즈니스 로직을 관리.

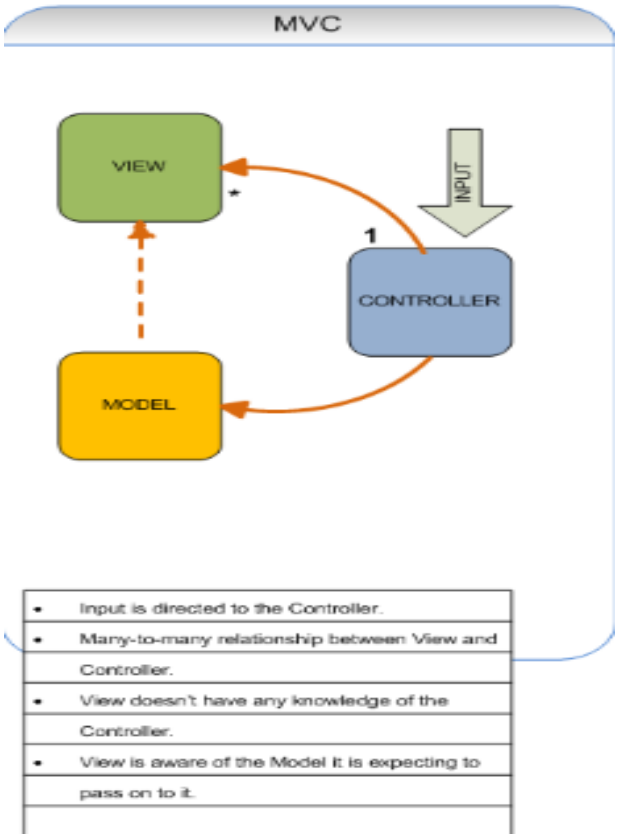
뷰(View): 사용자 인터페이스를 나타냄

컨트롤러(Controller): 모델과 뷰 사이의 상호 작용을 관리하고 비즈니스 로직을 처리함.

특징

사용자 인터페이스와 비즈니스 로직을 분리하여 유지보수와 테스트를 용이하게 함.

클래식한 웹 및 애플리케이션 개발에서 널리 사용됨.



모든 입력은 Controller 에서 처리된다. 입력이 Controller 로 들어오면 Controller 는 입력에 해당하는 Model 을 조작(업데이트) 하고, Model 을 나타내어줄 View 를 선택함.

이때 Controller 는 View 를 선택만 하고 업데이트를 시켜주지 않기 때문에 View 는 Model 을 이용하여 업데이트함.

View 는 Model 을 이용하기 때문에 서로 간의 의존성을 완벽히 피할 수 없다는 단점이 있고, 좋은 MVC 패턴이라 함은 View 와 Model 간의 의존성을 최대한 낮게 한 패턴이 좋은 패턴이라 할 수 있다

MVP (Model View Presenter)

구성요소

모델(Model): 데이터와 비즈니스 로직을 관리.

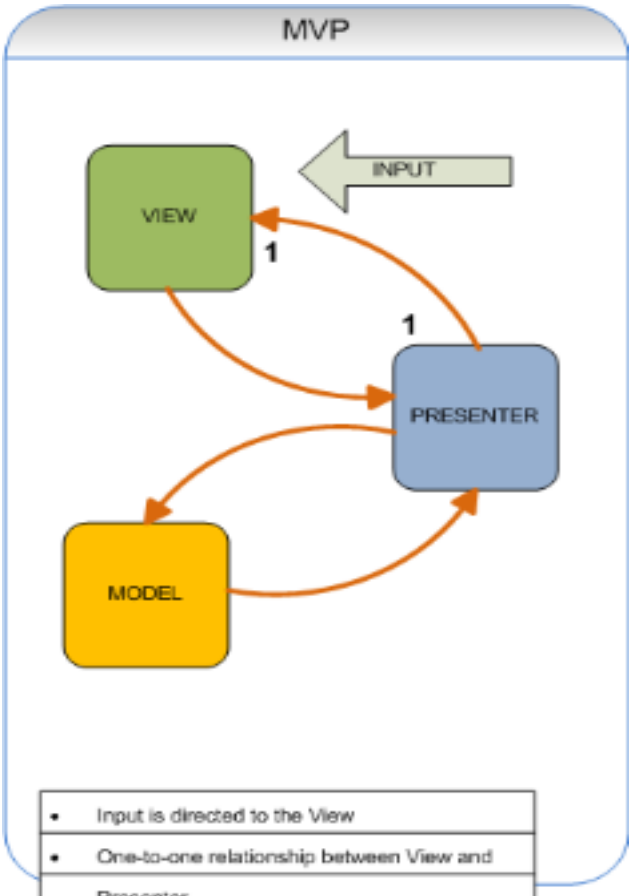
뷰(View): 사용자 인터페이스를 나타냄

프리젠터(Presenter): 뷰와 모델 사이의 중재자 역할을 하며 비즈니스 로직을 처리하고 뷰의 상태를 업데이트 함.

특징

뷰와 모델 간의 직접적인 의존성을 없애고 테스트 가능성을 높임.

안드로이드 개발에서 많이 사용되며, 사용자 인터페이스 로직을 분리하여 유연하고 확장 가능한 구조를 제공함



• Input is directed to the View
• One-to-one relationship between View and Presenter.
• View holds the reference to its Presenter and Presenter is aware of its View.
• View is not aware of the Model. Presenter updates the Model.

MVC 패턴과 다르게 입력이 View 에서 처리됨. Presenter 는 View 의 인스턴스를 갖고 있으며 View 와 1 대 1 관계이고, 그에 해당하는 Model 의 인스턴스 또한 갖고 있기 때문에 View 와 Model 사이에서 다리와 같은 역할을 한다.

View 에서 이벤트가 발생하면 Presenter 에게 전달해주고 Presenter 는 해당 이벤트에 따른 Model 을 조작하고 그 결과를 바인딩을 통해 View 에게 통보를 하여 View 를 업데이트 시켜줌. MVC 패턴과는 다르게 Presenter 를 통해 Model 과 View 를 완벽히 분리해 주기 때문에 View 는 Model 을 따로 알고 있지 않아도 된다는 장점이 있다.
단점으로는 View 와 1 대 1 관계이기 때문에 View 와의 의존성이 매우 강하다.

MVVM (Model View ViewModel)

구성요소

모델(Model): 데이터와 비즈니스 로직을 관리.

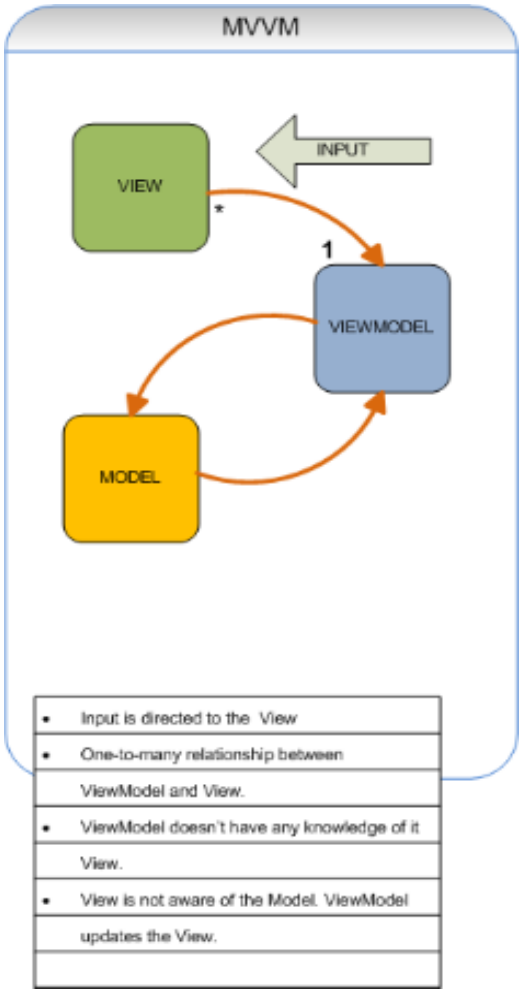
뷰(View): 사용자 인터페이스를 나타냄

뷰모델(ViewModel): 뷰와 모델 사이에서 중간 계층으로 동작하며, 뷰에 표시할 데이터와 명령을 노출시킴.

특징

데이터 바인딩을 통해 뷰와 모델을 연결하고, 비즈니스 로직을 포함해 뷰의 상태를 관리함.

리액티브 프로그래밍 및 데이터 바인딩 프레임워크와 함께 사용됨 ex) Angular, Vue.js, React 등



ViewModel 뷰 모델 말그대로 View 를 나타내 주기 위한 Model 이라고 생각하면 된다. View 보다는 Model 과 유사하게 디자인되며, View 의 바인딩 될 때 가장 강력하다. MVP 와 같이 View 에서 입력이 처리된다. MVVM 패턴의 가장 큰 장점이라 함은 Command 와 Data Binding 으로 MVP 패턴과 달리 View 와의 의존성을 완벽히 분리할 수 있다는 장점이 있다. Command 를 통하여 Behavior 를 View 의 특정한 ViewAction(Event)와 연결할 수 있으며, ViewModel 의 속성과 특정 View 의 속성을 Binding 시켜 줌으로써 ViewModel 속성이 변경될 때마다 View 를 업데이트 시켜줄 수 있다.

각 패턴은 다른 목적과 사용 사례에 맞게 선택되며, 각각의 장단점이 있음.

프로젝트의 요구사항, 개발 환경, 개발자의 경험과 선호도를 고려하여 선택 해야함.