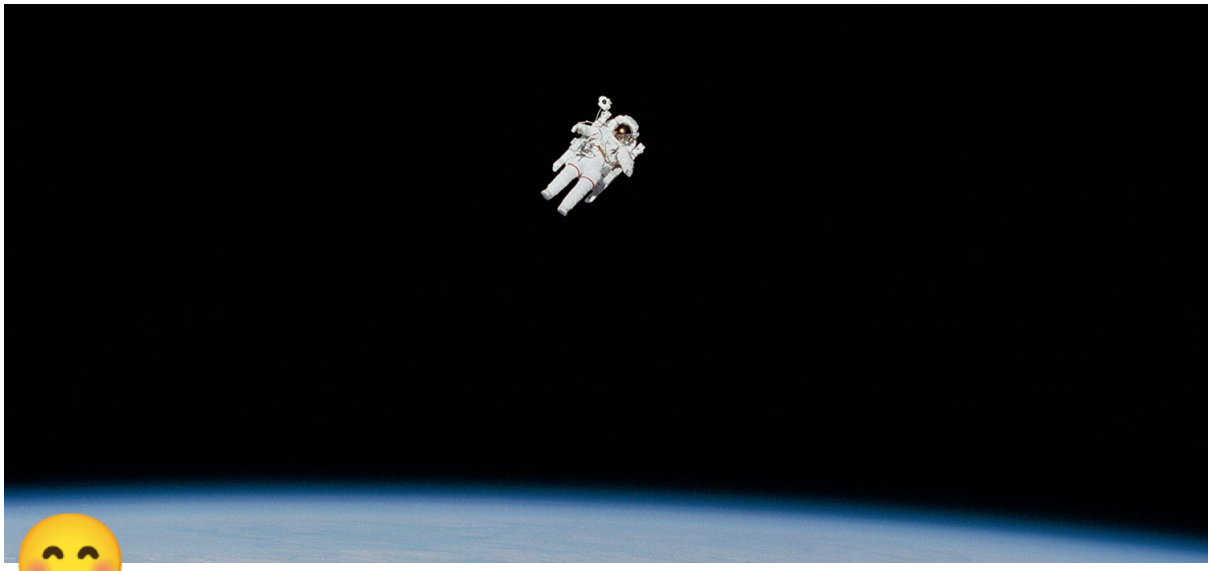


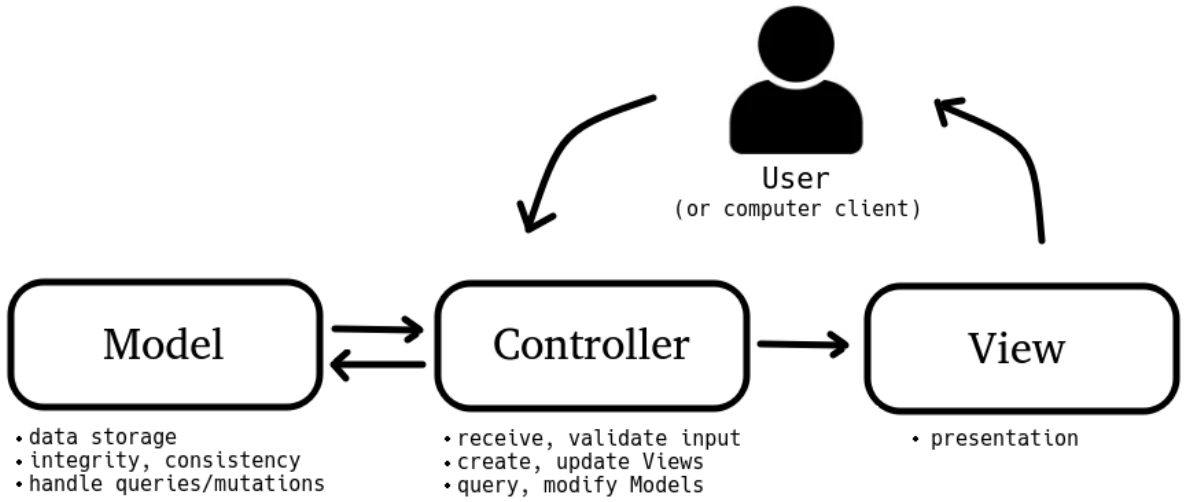
😊

MVC 모델 + 구현

☺ MVC 모델이란?

애플리케이션의 구성을 세 가지 관심사로 구분한 패턴이다.

이 패턴을 성공적으로 구현하면, 사용자에게 보여지는 UI와 Business Logic을 분리하여 서로 영향없이 쉽게 고칠 수 있는 애플리케이션을 만들 수 있게 된다.



😎 Model

: data와 data를 처리하는 부분

- 객체 혹은 Java의 **POJO**.
- View 혹은 Controller를 참조하는 내부 속성을 가지면 안된다(직접 수정 방지).
- 변경이 일어났을 때 Controller에 알려주는 부분이 구현되어야 한다.

😍 View

: 화면을 구성해주는 부분

- Model이 가지고 있는 데이터의 표현을 해주는 요소.
- 데이터를 따로 저장하지 않고, 받은 데이터를 단순 표현만 해야한다.
- Model에 질의를 해서 자신을 업데이트하는 부분이 구현되어야 한다.

🥺 Controller

: 사용자의 입력을 받고 처리하는 부분

- 사용자의 입력을 받고, validate 하는 역할.
- Model로 흘러가는 data flow를 통제(control)하고, Model에 변경이 일어날때마다 업데이트가 일어날 View를 선택 해준다.
- 이로써 Model과 View를 분리해주는 역할을 한다.
- Model과 View의 변경을 모니터링 한다.

🤓 동작순서

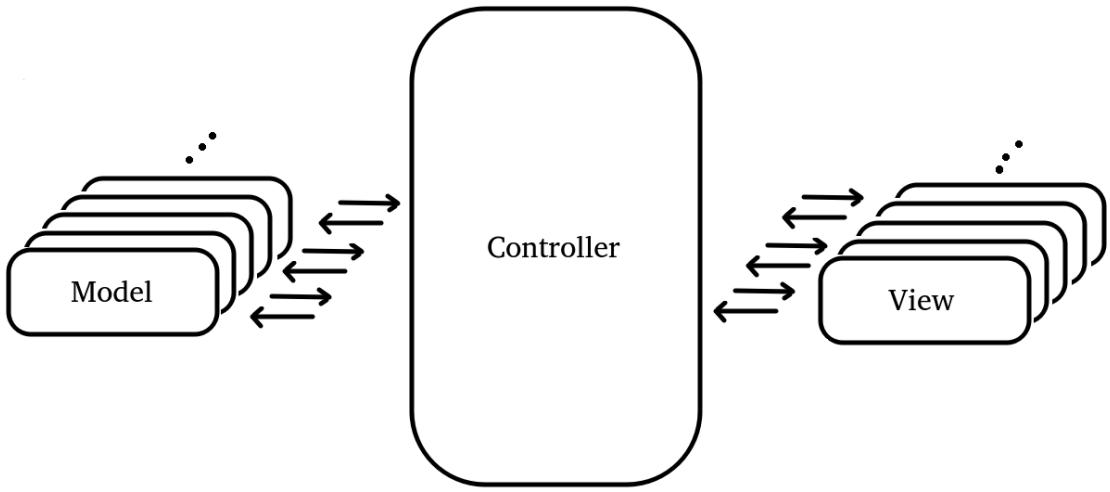
1. 사용자 입력을 Controller가 감지, validate
2. Controller가 action을 확인하고 business logic을 수행(Model을 조작).
3. Controller에서 해당 action 후에 보여줄 View를 선택
4. View가 Model을 이용하여 화면 표시

Controller의 역할은 보여줄 View를 선택하는것 까지이기 때문에, View는 Model의 메소드를 이용해 데이터를 가져오게 된다.

😅 특징

Controller와 View의 관계가 **1:1**이 아닌 1:N이다.

따라서 애플리케이션의 사이즈가 커질수록 Controller의 부담은 늘어만 가고, 이게 병목이 될 수 있다.



Massive ViewController

이런 상황을 풍자해 **Massive ViewController**라고 한다(이니셜은 같네).

그리고 View가 업데이트를 해야할때 Model을 이용하는 구조이기 때문에, Model과 View의 의존성이 높은 패턴이라고 할 수 있다.

🥰 그 대안으로 나온 패턴들

- MVP Pattern
- MVVM Pattern
- Flux Pattern