MERN STACK

비트캠프 KDT 5기
심민혜

[목차]

---

1. 개요

웹과 앱의 디렉토리 구조를 결정할 때는 설계 패턴을 고려해야한다. 이를 위해 전략패턴을 사용한다.
전략 패턴이란 필요에 따라 각 구성마다 다른 전략을 이용한 패턴이다.

2. 아키텍처

먼저 웹 개발을 위한 아키텍처로 Stack 기술을 사용한다.
Stack이란 한 쪽 끝에서만 자료를 넣거나 뺄 수 있는 선형구조 (LIFO : Last In First Out)이다. Front-end(Client), Back-end(Server)와 DataBase(API)를 통합적으로 관리해 웹 사이트와 애플리케이션을 구축하는데 사용된다.
Stack에는 다양한 종류가 있으나 JavaScript를 기반으로 한 Back-End 개발 프로그램인 MongoDB, Node.js가 출시되면서 개발자의 접근성과 편리성을 위해 JavaScript 한가지 언어로 Front-end와 Back-end 모두 작성 가능한 Full stack 웹 개발 기술인 MERN STACK를 사용한다.

MERN STACK 구성 요소들은 자유 오픈 소스 JavaScript 소프트웨어로 모두 무료로 사용 가능하다. 이를 기반으로 MERN STACK은 MongoDB, Express.js, React.js, Node.js로 구성되어 있다. 특히, 단일 페이지 애플리케이션을 구축하는데 사용되기 때문에 동적 웹 사이트와 애플리케이션을 개발하기 유리하다.

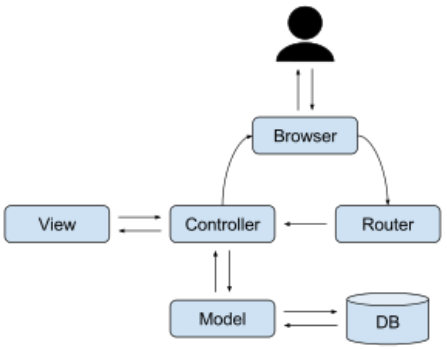
3. 구성

3-1. MERN STACK 구성

- MongoDB : 문서 DataBase
- Express.js : Node.js 웹 프레임워크 (Back-End)
- React.js : 사용자 인터페이스 제작을 위한 JavaScript 라이브러리 (Front-End)
- Node.js : JavaScript 웹 서버 (프로그램 실행을 위해 제공하는 환경)

3-2. MVC 패턴 구성

웹 애플리케이션 개발 프로세스를 위한 아키텍처로 MVC(Model- View- Controller) 패턴과 Flux 패턴을 기반으로 한다.
먼저, MVC 패턴은 Model-View- Controller로 수어 되어있는 소프트웨어 디자인 패턴으로, 사용자 인터페이스로부터 비즈니스 로직을 분리하여 애플리케이션의 시각적 요소나 그 이면에서 실행되는 비즈니스 로직을 서로 영향 없이 쉽게 고칠 수 있는 웹과 애플리케이션을 만들 수 있다. *(출처 : MVC- wikipedia)*



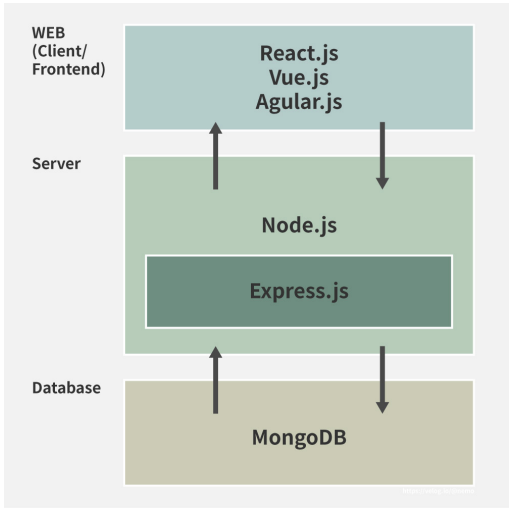
(Image 2 : MVC 구성요소 다이어그램)

- Model : Controller가 호출을 하면 DB와 연동하여 사용자의 입출력 데이터를 다루는 일과 해당 데이터와 연관된 비즈니스 로직을 처리하는 역할을

한다.
데이터 CRUD(Create, Read, Update, Delete) 역할을 수행한다.

- View : 사용자에게 보여주는 화면(UI).
별도의 데이터를 저장하지 않으며, Controller로 부터 받은 model의 결과값을 사용자에게 화면으로 출력한다.
- Controller : Action이 발생하면, Controller는 Model의 데이터를 가공하여 View로 전달하고, View는 사용자에게 Action의 결과를 보여준다.

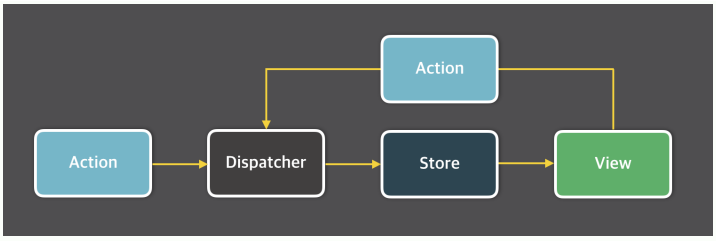
MERN STACK 구성 요소들은 MVC 패턴에 따라 Client, Server, DataBase로 구분된다



(Image 2 : MERN STACK 구성)

3-3. Flux 패턴 구성

Flux는 단방향 데이터 흐름을 활용해 MVC패턴의 복잡성을 해소하기 위해 사용 된다. Flux 패턴은 Dispatcher, Stores, View로 기본 구성 되어 있다. Action이 발생하면 Dispatcher에서 이를 받아와 해석한 후 Store에서 저장된 정보에 변경을 가하고 그 결과가 View로 다시 전달 되도록 한다.



(Image 3 : Flux 패턴 흐름)

- Dispatcher : 데이터 흐름을 관리하는 허브 역할. Action이 발생하면 Dispatcher로 Action 객체 전달되고 Dispatcher에서 등록된 콜백함수를 통해 Store에 전달. ⇒ Action을 Store에 전달. 동기적으로 실행된다.
- Stores : 상태 저장소. 무조건 Dispatcher를 통해 Action을 보내야만 데이터 변경이 가능하다.
- View : Action 발생(Action 생성자를 통해 Action을 준비). Store에 데이터가 변경되면 Store에서 데이터가 변경된 것을 View에 알려주고 다시 렌더링한다.
- Action 생성자 : 타입(type) 과 페이로드(payload) 를 포함한 Action을 생성
- Action : Action 생성자를 통하여 만들어 진다. Store에 변경할 데이터를 가지고 있다.

흐름은 Dispatcher → Stores → View 순서로 이뤄진다. View에서 데이터 입력이 일어나면 Action을 전달하여 Dispatcher를 통해 변경이 이뤄진다. 이때 View에서 데이터를 직접 변경할 수 없기 때문에 데이터 변경의 복잡도가 줄어든다.

3-4. 설치, 실행 및 데이터 반환 순서

Stack 구조에 따라 순서는 아래와 같다.
- 설치 : Node.js → React.js → Express.js → MongoDB
- 실행 : MongoDB → Express.js → React.js → Node.js
- 데이터 : React.js → Express.js → MongoDB → Express.js → React.js

4. 프로그램 환경 설정 및 설치

4-1. Node.js 개발 환경
URL : https://nodejs.org/ko/
Node 홈페이지에서 프로그램을 다운 받고 설치한다.
설치 확인을 위해서 명령창을 열고 “node-v”명령으로 버전을 확인한다.