

Individuals Project

비트캠프 KDT 5기 김승현

목차

- Visual Studio CODE
- MERN Stack
- Docker 설치 및 DB연결
- 설계
- 실행

개발환경설정

VSCODE

VSCODE는 마이크로소프트에서 오픈소스로 개발하고 있는 소스코드 에디터이다. 웹 기반의 기술들로 데스크톱 애플리케이션을 만들 수 있기에 맥, 리눅스, 윈도우등 메이저 운영체제를 모두 지원 하고 있다. 마이크로 소프트의 비슷한이름 Visual Studio와 이름이 비슷하지만 별개로 코드에디터에 가깝다.

VSCODE의 주요한 특징으로는 디버깅, 리팩토링, 빌트인 깃등을 기본 포함하고 있으며 웹 기반 기술로 이루어져 있어 타입스크립트로 확장 개발이 가능하다는 점이다.

VSCODE 설치방법

1. <https://code.visualstudio.com/> 에 접속하여 자신이 사용하는 운영체제에 맞춰 다운로드 한다.

2.

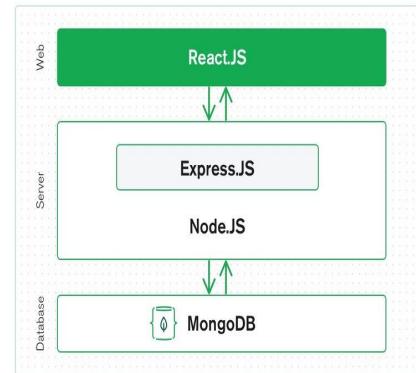
Stack

Stack이란? 한 쪽 끝에서만 자료를 넣고 뺄 수 있는 LIFO형식의 자료 구조이다. LIFO는 Last In First Out의 약자로 가장 최근에 들어온 자료가 가장 먼저나가게 되는 형태를 말한다. Stack의 입출력은 맨 위에서만 일어나기 때문에 스택의 중간에서는 데이터를 삭제하는 것이 불가능하다.

MERN Stack

MERN은 MongoDB + Express + React + Node.js의 줄임말이다. MERN stack은 위의 네가지 요소를 사용하여 웹사이트를 개발하는 것을 말한다.

왜 MERN Stack으로 개발해야 하는가?



MERN stack의 가장 아래 계층인 MongoDB부터 보자.

MongoDB는 JSON을 저장하기 위해 고안되어진 것이다.

(JSON의 binary 버전인 BSON) MongoDB는 명령어와 쿼리 언어 모두가 JSON과 JavaScript로 만들어져 있다. 때문에 MongoDB와 Node.js는 호환이 좋으며 MongoDB의 모든 다른 계층에서 JSON데이터를 저장, 조작, 표현이 가능하다.

MongoDB - Express.js - React.js의 결합은 JSON 데이터가 front에서 back으로 자연스럽게 흐르게 만든다. 따라서 디버깅이 쉬워지고 만들기 쉬워진다. 또한 가장 큰 장점은 단 하나의 프로그래밍 언어인 JavaScript와 JSON 구조만 알면 전체 시스템 이해가 가능하다는 것이다.

Node.js란?

Node.js 공식 사이트에서는 다음과 같이 설명하고 있다.

'Node.js는 Chrome V8 Javascript 엔진으로 빌드된 Javascript 런타임' 즉, Node.js를 통해서 다양한 자바스크립트 애플리케이션을 실행 할 수 있으며, 서버를 실행하는데 제일 많이 사용 된다.

Node.js 설치 방법

1. <https://nodejs.org/en/> 접속하여 윈도우에 맞는 버전을 다운로드 한다.(최신버전 보다는 안정성이 검증된 LTS를 추천한다.)
2. 설치 완료 후 명령 프롬프트(cmd)에 명령어 node -v를 입력하여 버전이 뜨는지 확인

Express란?

express 공식사이트에서는 express란 Node.js를 위한 빠르고 개방적인 간결한 웹 프레임워크' 라고 한다. 짧게 말하면, express란 Node.js를 사용하여 쉽게 서버를 구성할 수 있게 만든 함수형 프로그래밍이다.

Express 설치 방법

1. vscode 접속 후 새로운 terminal을 열어 npm 또는 yarn을 이용하여 express 를 설치한다. 명령어는 yarn add express 혹은 npm install –save express이다.
2. 설치 완료후 테스트를 위하여 임시로 server.js 파일을 생성해준다.

Next.js란?

Next.js는 React 라이브러리의 프레임워크이다. React를 쓰면서 왜 Next.js가 필요한지 의문을 가질 수 있다. 그 대답은 Next.js의 장점을 보면 알 수 있다.

Next.js의 가장 큰 장점으로는 SSR(Server-side Rendering)을 가능하게 한다는 것이다. 기본적으로 React는 CSR(Client Side Rendering)을 한다. CSR은 웹사이트를 요청 했을 때 빈 html을 가져와 script를 로딩하기 때문에 첫 로딩시간이 오래 걸린다. 반면 Next.js는 pre-loading을 통해 미리 데이터가 렌더링 된 페이지를 가져올 수 있게 해주므로 사용자는 좀 더 시간을 단축 할 수 있을 뿐만 아니라 검색엔진에서도 잘 노출 될 수 있도록 해주는 SEO에서도 장점을 얻을 수 있다.

Next.js 설치 방법

1. 설치 이전에 VScode 및 Node.js가 설치되어 있는지 확인
 2. 콘솔창에 npx create-next-app [프로젝트명]을 입력하여 프로젝트를 생성.
 3. 설치가 완료 되면 프로젝트명과 동일한 디렉토리가 생성 됨
 4. 프로젝트 디렉토리로 이동 후 npm run dev 명령어를 입력하여 로컬 3000번포트가 제대로 작동하는지 확인
- 여담으로 React와 달리 next에서는 CSR과 SSR 둘 다 만들 수 있다. 본래 React에서는 위와 같은 기능이 어려웠지만 18버전이 간단되면서 가능해졌다. 하지만 현재 React 18버전은 상용화 조금 어려운점이 있어 next를 사용하고자 한다.

MongoDB란?

MongoDB는 NoSQL 데이터베이스로, JSON 형태의 데이터를 저장하는 도큐먼트 지향 데이터베이스이다. 웹애플리케이션과 인터넷 기반을 위해 설계된 데이터베이스이다. 데이터 모델과 지속성 전략은 높은 읽기,쓰기 효율과 Failover를 통한 확장의 용이성을 염두해 두고 만들어졌다. 많은 개발자들이

MongoDB를 사용하는 이유로는 확장성 보다는 직관적인 데이터모델 때문이다. 데이터 값은 Row값이 아닌 document에 저장한다. MongoDB는 BSON(Binary JSON)이라고 하는 Binary 인코딩 형식으로 JSON문서를 저장한다. BSON은

JSON 모델을 확장하여 추가 데이터 유형, 순서 필드를 제공하고 서로 다른 언어 내에서 인코딩 및 디코딩에 효율적이다.

MongoDB 설치 방법

1. www.mongodb.com/try/download/enterprise에 접속하여 on-promist 박스 클릭 후 버전과 윈도우를 확인후 다운로드 한다.
2. 설치 완료후 시스템 환경변수에 접속, 시스템변수의 “Path” 선택 후 편집을 클릭한다. 새로만들기 클릭 후 몽고DB의 bin경로를 지정한다.
3. 윈도우 명령프롬프트(커맨드창)를 실행 후 명령어 “mongo”를 입력하면 기본포트 27017로 연결되어 있음을 확인 할 수 있다.

Docker란?

도커는 리눅스의 응용 프로그램들을 프로세스 격리 기술들을 사용해 컨테이너로 실행하고 관리하는 오픈 소스 프로젝트이다.

Docker 설치 방법

1. Docker Desktop for Mac and Windows | Docker에 접속하여 Download버튼을 클릭한다.
2. 설치가 완료되고 Docker가 실행되면 Tutorial01 나타난다.

Docker를 이용하여 몽고DB컨테이너 만들기

MongoDB를 설치하기 위해서는 도커허브에 MongoDB 이미지를 다운받아야 하기에 명령프롬트를 열어 docker pull mongo 를 입력해준다. 명령어가 실행 되면 Docker hub에 있는 MongoDB 이미지를 사용자 컴퓨터에 다운 받아 사용 가능하다. 이후에 아래 명령어를 차례대로 입력해주면 Docker에 mongodb_docker라는 container가 추가 돼 있는 것을 확인할 수 있다.

```
run --name mongodb_docker -e
MONGO_INITDB_ROOT_USERNAME=root -e
MONGO_INITDB_ROOT_PASSWORD=root -d -p 27017:27017
mongo
```

아래 명령 프롬프트 화면을 참고하여 DB컨테이너를 생성하여 임시로 내용을 넣었다 뺏는 작업이 가능함을 알 수 있다.

```

PS C:\Users\bitcamp\soccer\soccer-express> mongosh "mongodb://localhost:27017" --username "root" --password "root"
Current Mongosh Log ID: 624d2a62cde72cb3ba4bb3f
Connecting to: mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Using MongoDB: 5.0.6
Using Mongosh: 1.1.7

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting:
2022-04-06T05:51:25.756+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-04-06T05:51:26.359+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
-----

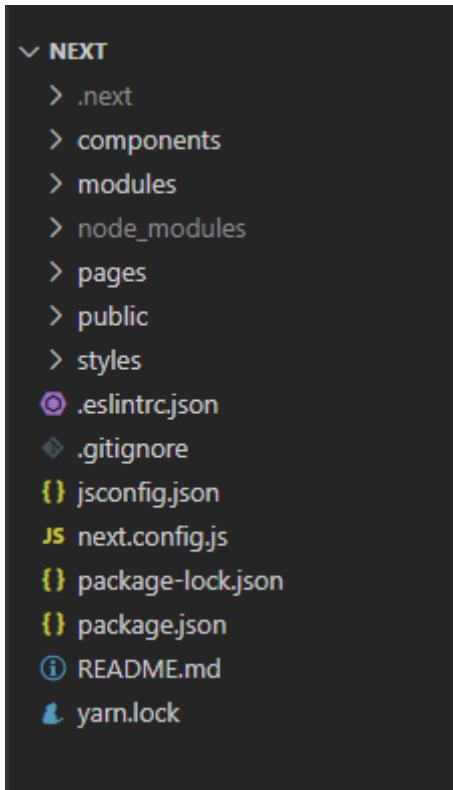
test> show dbs
admin 213 kB
config 36.9 kB
local 73.7 kB
test> use soccer_db
switched to db soccer_db
soccer_db> db
Browserslist: caniuse-lite is outdated. Please run:
npx browserslist@latest --update-db

Why you should do it regularly:
https://github.com/browserslist/browserslist#browsers-data-updating
soccer_db>
soccer_db> show dbs
admin 213 kB
config 36.9 kB
local 73.7 kB
soccer_db> db.user.insert({username:'test','password':'11','name':'테스트','telephone':'010-1234'})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '_id': ObjectId("624d2ba4ef7fa158d10a2d1f") }
}
soccer_db> _show dbs
admin 213 kB
config 73.7 kB
local 73.7 kB
soccer_db 8.19 kB
soccer_db> show collections
user
soccer_db> |

```

설계

컴포넌트란 리액트로 만들어진 앱을 이루는 최소한의 단위이다. 컴포넌트는 MVC의 뷰를 독립적으로 구성하여 재사용을 할 수 있고 이를 통해 새로운 컴포넌트를 쉽게 만들 수 있다.



우선 첫번째로, React부분인 components를 만들고 Redux와 Saga 인 modules, Next부분의 page폴더를 만든다. 각 폴더에 회원가입의 register와 로그인 login을 생성한다. 템플릿은 mui를 사용한다.

components - common - index.js

```

components > JS index.js
components > JS index.js
1  export * from './common/Footer'
2  export * from './common/Header'
3  export * from './common/Layout'
4  export * from './common/Modal'
5  export * from './common/Nav'
6  export * from './common/Home'
7  export * from './common/Pagination'
8  export * from './common/Table'
9  export * from './basic/Counter'
10 export * from './auth/Login'
11 export * from './auth/Register'

```

위와 같이 코딩을 해준다. components에는 화면을 구성하는 코딩이 작성되고 modules에는 각 components의 데이터 전달 상태를 관리하기 위한 상태관리 도구인 리덕스를 코딩해준다. 모든 코딩은 바닐라 스크립트로 이루어진다.

```

export function Login({onChange, onSubmit}) {
  return (
    <ThemeProvider theme={theme}>
      <Container component="main" maxWidth="xs">
        <CssBaseline />
        <Box sx={{ marginTop: 2, display: 'flex', flexDirection: 'column', alignItems: 'center' }}>
          <Avatar sx={{ m: 1, bgcolor: 'secondary.main' }}>
            <LockOutlinedIcon />
          </Avatar>
          <Typography component="h1" variant="h5">
            로그인
          </Typography>
          <Box component="form" noValidate sx={{ mt: 1 }} onSubmit={onSubmit}>
            <TextField
              margin="normal"
              required
              fullWidth
              id="userid"
              label="사용자ID"
              name="userid"
              autoComplete="email"
              autoFocus
              onChange={onChange}
            />
            <TextField
              margin="normal"
              required
              fullWidth
              name="password"
              label="비밀번호"
              type="password"
              id="password"
              autoComplete="current-password"
              onChange={onChange}
            />
          </Box>
        </Box>
      </Container>
    </ThemeProvider>
  );
}

```

components의 Login.js 부분이다. 상태를 가지지 않는 상태로 코딩 하며 로그인 템플릿을 적용한다. Onchange는 id password 와 같은 이벤트가 발생(props) 했을 때는 이벤트 값을 받으며 OnSubmit은 이벤트 데이터를 전송하는데 사용된다. CallBack 함수를 사용한다.

```

modules > auth > JS login.js ...
1 import {createAction, handleActions} from 'redux-actions';
2 import {
3   call,
4   delay,
5   put,
6   takeLatest,
7   select,
8   throttle
9 } from 'redux-saga/effects';
10 import {HYDRATE} from "next-redux-wrapper"
11 import axios from 'axios'
12
13
14 const SERVER = 'http://127.0.0.1:5000'
15 const headers = {
16   "Content-Type": "application/json",
17   Authorization: "JWT fefefe...",
18   withCredentials: true
19 }
20 export const initialState = {
21   loginUser: null,
22   isLoggedIn: false,
23   token: '',
24   loginError: null
25 }
26
27 const LOGIN_REQUEST = 'auth/LOGIN_REQUEST';
28 const LOGIN_SUCCESS = 'auth/LOGIN_SUCCESS';
29 const LOGIN_FAILURE = 'auth/LOGIN_FAILURE';
30 const LOGIN_CANCELLED = 'auth/LOGIN_CANCELLED';
31 const LOGOUT_REQUEST = 'auth/LOGOUT_REQUEST';
32 const LOGOUT_SUCCESS = 'auth/LOGOUT_SUCCESS';
33 const LOGOUT_FAILURE = 'auth/LOGOUT_FAILURE';
34 const SAVE_TOKEN = 'auth/SAVE_TOKEN';
35 const DELETE_TOKEN = 'auth/DELETE_TOKEN';
36
37 export const loginRequest = createAction(LOGIN_REQUEST, data => data)
38 export const loginCancelled = createAction(LOGIN_CANCELLED, data => data)
39 export const logoutRequest = createAction(LOGOUT_REQUEST)
40
41 export function* loginSaga() {
42   yield takeLatest(LOGIN_REQUEST, signin);
43   yield takeLatest(LOGIN_CANCELLED, loginCancel);
44   yield takeLatest(LOGOUT_REQUEST, logout);
45 }

```

modules의 login.js 파일로 server와 연결을 담당하며 Redux와 Saga가 존재한다. express 서버로 넘기는 역할을 하며 axios를 post를 사용하여 값을 이동 시킨다.

components에서 이벤트로 발생한 action을 이곳에서 받아 Reducer Saga를 거쳐 api로 전송시킨다.

```

const LoginPage = ({}) => {
  const [user, setUser] = useState({userid: '', password: ''})
  const dispatch = useDispatch()
  const router = useRouter()
  const onChange = e => {
    e.preventDefault()
    const {name, value} = e.target;
    setUser({
      ...user,
      [name]: value
    })
  }
  const {isLoggedIn, loginUser} = useSelector(state => state.login)
  const onSubmit = e => {
    e.preventDefault()
    console.log(`로그인 정보 ${JSON.stringify(user)}`)
    console.log(history)
    dispatch(loginRequest(user))
    console.log('모듈에 저장된 로그인값: '+JSON.stringify(loginUser))
    //router.push('/user/profile') 이동시 데이터소실
  }
  return (isLoggedIn ?
    <Profile loginUser={loginUser}/>
    :<Login onChange={onChange} onSubmit={onSubmit}/>)
};

const mapStateToProps = state => ({loginUser: state.login.loginUser})
const loginActions = {loginRequest}
export default connect(mapStateToProps, loginActions)(LoginPage);

```

pages의 login.js 파일이다. dispatch를 이용하여 React에서 Redux로 데이터를 전달하게 된다.

```

.env
PORT=5000
MONGO_URI=mongodb://root:root@localhost:27017/soccerdb?authSource=admin&authMechanism=SCRAM-SHA-1
JWT_SECRET=jwtSecret
ORIGIN=http://localhost:3000

```

다음은 express파일로 넘어와서 처음으로 .env파일을 생성 후에 위와 같이 입력해준다. 위의 코드는 MongoDB와 React를 연결하기 위한 코드이다. 여기에서는 Token은 PostMan을 사용한다.

```

server.js ...
1 import dotenv from 'dotenv'
2 import express from 'express'
3 import passport from 'passport'
4 import morgan from 'morgan'
5 import db from './app/models/index.js'
6 import api from "./app/routes/api.js"
7 import basic from "./app/routes/basic.js"
8 import user from "./app/routes/user.js"
9 import index from "./app/routes/index.js"
10 import getResponse from "./app/lambdas/getResponse.js"
11 import applyPassport from './app/lambdas/applyPassport.js'
12 import applyDotenv from './app/lambdas/applyDotenv.js'
13
14 async function startServer() {
15   const app = express();
16   const {mongoUri, port, jwtSecret } = applyDotenv(dotenv)
17   app.use(express.static('public'));
18   app.use(express.urlencoded({extended: true}));
19   app.use(express.json());
20   const _passport = applyPassport(passport, jwtSecret);
21   app.use(_passport.initialize());
22   app.use("/", index);
23   app.use("/api", api);
24   app.use("/basic", basic);
25   app.use("/user", user);
26   app.use(morgan('dev'))
27   db
28     .mongoose
29     .connect(mongoUri, {
30       useNewUrlParser: true,
31       useUnifiedTopology: true
32     })
33     .then(() => {
34       console.log(`### 몽고DB 연결 성공 ###`)
35     })
36     .catch(err => {
37       console.log(`몽고DB와 연결 실패`, err)
38     });
39 }

```

다음은 엔트리 포인트인 server.js 파일을 만들고 위와 같이 코딩을 해준다. 엔트리 포인트를 잡는 use를 사용하고 mongoose사용을 위하여 위와 같이 코딩해준다. 라이브러리 설치는 npm i mongoose를 입력하여 진행해준다.

```

const app = express()
app.use(cors());
app.use(function (_req, res, next) {
  res.header(
    "Access-Control-Allow-Headers",
    "x-access-token, Origin, Content-Type, Accept"
  );
  next();
});
app.post('/join', cors(corsOptions), (req, res) => {
  UserService().join(req, res)
})
app.post('/login', cors(corsOptions), (req, res) => {
  UserService().login(req, res)
})
app.get(
  '/logout',
  passport.authenticate('jwt', {session: false}),
  function (req, res) {
    UserService().logout(req, res)
    req.logout();
    res.json({msg: 'LOGOUT'});
  }
);
export default app

```

다음으로 Routes파일은 req, res, next 를 받는 middleware 함수를 사용한다.

```

import db from '../models/index.js'
import getDatabase from '../lambdas/getDatabase.js'

export default function UserService() {

  const User = db.User
  const dbo = getDatabase()
  const dbConnect = dbo.getDb();

  return {
    join(req, res) {
      new User(req.body).save(function (err) {
        if (err) {
          res
            .status(500)
            .send({message: err});
          console.log('회원가입 실패')
          return;
        } else {
          res
            .status(200)
            .json({ok: 'ok'})
        }
      })
    }
  }
}

```

service 파일에는 확인시켜주는 코드를 console.log로 확인하고 DB와 데이터가 잘 연결 돼 있는지 확인해준다.

실행

MongoDB 실행

Docker를 실행후 mongodb의 실행버튼을 눌러 활성화시켜준다.



Express 실행

```

PS C:\Users\swhan\bitcamp\TeamProject\express> node server
***** 서버가 정상적으로 실행되고 있습니다 *****
### 봄고DB 연결 성공 ###

```

Next 실행

```

PS C:\Users\swhan\bitcamp\TeamProject\next> yarn dev
yarn run v1.22.17
$ next dev
  ready - started server on 0.0.0.0:3000, url: http://localhost:3000
  event - compiled client and server successfully in 2.4s (840 modules)

```

위 단계가 모두 끝났다면 **Port 3000**에 접속하여 정상적으로 기능이 작동하는지 확인해준다.

The top screenshot shows a 'Sign in' page with fields for '사용자 아이디*' and '비밀번호*', a 'Remember me' checkbox, and a 'SIGN IN' button. Below it, a 'Forgot password?' link and a 'Don't have an account? Sign Up' link are visible. The bottom screenshot shows a '회원가입' (Registration) page with fields for '사용자 ID*', '이름*', '비밀번호*', '이메일*', '폰 번호*', '생년월일*', and '주소*'. A checkbox for receiving marketing emails is present, along with a '전송' (Send) button and a '로그인 화면으로 전환' (Switch to login screen) link.