소스 코드

초기 소스 코드입니다

src/App.js

```
/**
 * @filename: App.js
 * @description: index.js에 페이지 라우팅
 * @author: 최수진(sujin971008@gmail.com)
 */
import React from "react";

import Misea from "./pages/misea";

function App() {
  return (
    <div>
      <Misea />
    </div>
  );
}

export default App;
```

src/index.js

```
/**
 * @filename: index.js
 * @description: store로 부터 구독받아서 App.js를 통해서 전달
 * @author: 최수진(sujin971008@gmail.com)
 */

import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { BrowserRouter } from "react-router-dom";
import Meta from "./Meta";

/* 리덕스 구성을 위한 참조 */
import { Provider } from "react-redux";
import store from "./store/misaeStore";

const root = ReactDOM.createRoot(document.getElementById("root"));
```

```
root.render(
  <React.StrictMode>
    <Meta />
    <Provider store={store}>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </Provider>
  </React.StrictMode>
);
```

src/Meta.js

```
import React from "react";
import { Helmet, HelmetProvider } from "react-helmet-async";

const Meta = (props) => {
  return (
    <HelmetProvider>
      <Helmet>
        <meta charset="utf-8" />
        <title>{props.title}</title>
        {/* SEO 태그 */}
        <meta name="description" content={props.description} />
        <meta name="keywords" content={props.keywords} />
        <meta name="author" content={props.author} />
        <meta property="og:type" content="website" />
        <meta property="og:title" content={props.title} />
        <meta property="og:description" content={props.description} />
        <meta property="og:image" content={props.image} />
        <meta property="og:url" content={props.url} />
        <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

        {/* Helmet 안에서 CSS적용하기 */}
        <style type="text/css">{`
            *{
                list-style: none;
            }

            body {
                margin: 0;
                padding: 0;
            }

            `}</style>
        {/* 추가적으로 적용해야 할 외부 js나 css로 여기서 명시할 수 있다. */}
      </Helmet>
    </HelmetProvider>
  );
```

```
};
/* 이게 props */
Meta.defaultProps = {
  title: "useage of OpenAPI",
  description: "OpenAPI 활용 제출물입니다.",
  keywords: "OpenAPI",
  author: "최수진,박세영",
  url: window.location.href,
};

export default Meta;
```

src/slices/misaeSlice.js

```
/**
 * @filename: misaeSlice.js
 * @description: axios처리 및 상태값 관리하는 slice.js
 * @author: 최수진(sujin971008@gmail.com)
 */

import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";

/* KEY */
const KEY =

"H1klrl5LXfzjFdnp01Zfej7sKKi5WysCM9D5tQRb0tp3pOXckKcmZnf%2FG3Gm8ESgORTZfC6QCMleUYP
p%2F5MHaw%3D%3D";

/* 비동기 처리 함수 구현 */
//  payload는 이 함수를 호출할 때 전달되는 파라미터
export const getInfo = createAsyncThunk(
  "openAPI/getInfo",
  async (payload, { rejectWidthValue }) => {
    let result = null;

    try {
      result = await axios.get(

`http://apis.data.go.kr/B552584/ArpltnInforInqireSvc/getCtprvnRltmMesureDnsty?
sidoName=서울&pageNo=1&numOfRows=100&returnType=json&serviceKey=${KEY}&ver=1.0`
      );
    } catch (err) {
      // 에러 발생 시 'rejectWidthValue()'함수에 에러 데이터를 전달하면 extraReducer
의 rejected 함수가 호출된다.
      result = rejectWidthValue(err.response);
    }
    return result;
  }
```

```
);

/* Slice 정의 (Action함수 + Reducer의 개념) */
const misaeSlice = createSlice({
  name: "getInfo",
  initialState: {
    items: null,
    loading: false,
    error: null,
  },
  // 내부 action 및 동기 action
  reducers: {},
  // 외부 action 및 비동기 (Ajax용)
  extraReducers: {
    [getInfo.pending]: (state, { payload }) => {
      return { ...state, loading: true };
    },
    [getInfo.fulfilled]: (state, { payload }) => {
      return {
        /* 원하는 key가 깊이 있어서 items까지 접근 */
        items: payload?.data?.response?.body?.items,
        loading: false,
        error: null,
      };
    },
    [getInfo.rejected]: (state, { payload }) => {
      return {
        items: null,
        loading: false,
        error: {
          code: payload?.status ? payload.status : 500,
          message: payload?.statusText ? payload.statusText : "Server Error",
        },
      };
    },
  },
});
// 리듀서 객체 내보내기
export default misaeSlice.reducer;
```

src/store/misaeStore.js

```
/**
 * @filename: misaeStore.js
 * @description: slice 묶어서 index.js에 전달
 * @author: 최수진(sujin971008@gmail.com)
 */
import { configureStore } from "@reduxjs/toolkit";
import misaeSlice from "../slices/misaeSlice";
```

```
const store = configureStore({
  reducer: { getInfo: misaeSlice },
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware({ serializableCheck: false }),
  devTools: true,
});

export default store;
```

src/pages/misae.js

```
/**
 * @filename: misae.js
 * @description: 화면에 실질적으로 보여지는 컴포넌트 (액션함수를 dispatch한다)
 * @author: 최수진(sujin971008@gmail.com)
 */
import React from "react";
import styled from "styled-components";
import { useSelector, useDispatch } from "react-redux";

/* slice  */
import { getInfo } from "../slices/misaeSlice";

import Spinner from "../components/Spinner";
import Error from "../components/Error";

/* styledComponent */
const Div = styled.div``;
const SelectContainer = styled.div`
  position: sticky;
  top: 0;
  background-color: #fff;
  padding: 10px 0;
  margin: 0;

  select {
    margin-right: 15px;
    font-size: 16px;
    padding: 5px 10px;
  }
`;

const Table = styled.table``;

const Misae = () => {
  React.useEffect(() => console.clear(), []);
  // hook을 통해 slice가 관리하는 상태값 가져오기
  const { items, loading, error } = useSelector((state) => state.getInfo);
```

```
  /* Grade 함수1 */
  function Grade(x) {
    let state = null;
    if (x === "1") {
      state = "좋음";
    } else if (x === "2") {
      state = "보통";
    } else if (x === "3") {
      state = "나쁨";
    } else if (x === "4") {
      state = "매우나쁨";
    } else {
      state = "오류";
    }
    return state;
  }
  /* Grade값 함수2 */
  function Emoji(y) {
    let state = null;
    if (y === "1") {
      state = "😀";
    } else if (y === "2") {
      state = "😐";
    } else if (y === "3") {
      state = "🥸";
    } else if (y === "4") {
      state = "😂";
    } else {
      state = "오류";
    }
    return state;
  }

  // dispatch 함수 생성
  const dispatch = useDispatch();

  /* 상태값을 '중구'로 기본값을 잡아줌으로써, 페이지가 열리자 마자 정보를 보여주게 된다
(앱과 같은 느낌을 주고 싶었다)  */
  const [stationName, setStationName] = React.useState("중구");

  // 컴포넌트가 마운트되면 데이터 조회를 위한 액션함수를 디스패치 함
  React.useEffect(() => {
    dispatch(getInfo());
  }, [dispatch, stationName]);

  /* 자치구 선택 시 Event */
  const onSelectChange = React.useCallback((e) => {
    e.preventDefault();
    // 드롭다운의 입력값 취득
    const current = e.target;
    const value = current[current.selectedIndex].value;
    setStationName((stationName) => value);
  }, []);
```

```
  return (
    <>
      <Spinner visible={loading} />

      {/* 검색 조건 드롭다운 박스 */}
      <SelectContainer>
        <select
          name="location"
          onChange={onSelectChange}
          style={{ border: "3px solid #168", borderRadius: "7%" }}
        >
          {/* items의 지역명을 반복 돌려서 option 생성 */}
          <option value="">-- 자치구 선택 --</option>
          {items &&
            items.map((v, i) => {
              return (
                <option value={v.stationName} key={i}>
                  {v.stationName}
                </option>
              );
            })}
        </select>
      </SelectContainer>

      {error ? (
        <Error error={error} />
      ) : (
        <Div>
          <div>
            <h1>서울특별시</h1>
            <h3>{stationName}</h3>
          </div>
          {items &&
            items.map((v, i) => {
              return (
                <div>
                  {/* 통합지수 */}
                  <p>
                    {v.stationName === stationName ? Grade(v.khaiGrade) : ""}
                  </p>
                  {/* 통합지수 이모지 */}
                  <span style={{ fontSize: "120px" }}>
                    {v.stationName === stationName ? Emoji(v.khaiGrade) : ""}
                  </span>
                  {/* 미세먼지 */}
                  <div>
                    {v.stationName === stationName ? (
                      <ul>
                        <li>
                          미세먼지: {v.pm10Value}㎍/㎥
                          {Emoji(v.pm10Grade)}
                        </li>
                        <li>
```

```
                        초미세먼지: {v.pm25Value}㎍/㎥
                        {Emoji(v.pm25Grade)}
                      </li>
                    </ul>
                  ) : (
                    <ul></ul>
                  )}
                </div>
              </div>
            );
          })}
      </Div>
    )}
    {/* 기타 대기오염지수 테이블 */}
    {error ? (
      <Error error={error} />
    ) : (
      <Table>
        {items &&
          items.map((v, i) => {
            return v.stationName === stationName ? (
              <thead>
                <tr>
                  <th>아황산가스</th>
                  <td>
                    {Grade(v.so2Grade)}
                    {Emoji(v.so2Grade)}
                  </td>
                  <td>{v.so2Value}</td>
                </tr>
                <tr>
                  <th>일산화탄소</th>
                  <td>
                    {Grade(v.coGrade)}
                    {Emoji(v.coGrade)}
                  </td>
                  <td>{v.coValue}</td>
                </tr>
                <tr>
                  <th>오존</th>
                  <td>
                    {Grade(v.o3Grade)}
                    {Emoji(v.o3Grade)}
                  </td>
                  <td>{v.o3Value}</td>
                </tr>
                <tr>
                  <th>이산화질소</th>
                  <td>
                    {Grade(v.no2Grade)}
                    {Emoji(v.no2Grade)}
                  </td>
                  <td>{v.no2Value}</td>
                </tr>
```

```
                </thead>
              ) : (
                <thead></thead>
              );
            })}
        </Table>
      )}
      <div>
        <i class="fa fa-exclamation-triangle" aria-hidden="true"></i>
        <p>자료 출처: 공공데이터포털을 통한 환경부/한국환경공단 에어코리아 </p>
        <p>
          주의사항: 해당 기관이 제공하는 자료는 “인증을 받지 않은
          실시간자료”이므로 자료 오류 및 표출방식에 따라 값이 다를 수 있습니다.
        </p>
      </div>
    </>
  );
};

export default React.memo(Misae);
```

src/components/Error.js

```
import React from "react";

const Error = ({ error }) => {
  return (
    <div>
      <h1>Oops~!!! {error.code} Error.</h1>
      <p>{error.message}</p>
    </div>
  );
};

export default React.memo(Error);
```

src/components/Spinner.js

```
import React from "react";
import PropTypes from "prop-types";
import styled from "styled-components";

/* 로딩바 컴포넌트 */
// --> https://mhnpd.github.io/react-loader-spinner/
```

```
import { Bars } from "react-loader-spinner";

/* 로딩바 뒤에 표시될 반투명 막 */
const TransLayer = styled.div`
  position: fixed;
  left: 0;
  top: 0;
  z-index: 9999;
  background-color: #0003;
  width: 100%;
  height: 100%;
`;
// visible은 boolean값
const Spinner = ({ visible, color }) => {
  return (
    <div>
      {visible && (
        <TransLayer>
          <Bars
            color={color}
            wrapperStyle={{
              position: "absolute",
              zIndex: 10000,
              left: "50%",
              top: "50%",
              transform: "translate(-50%,-50%)",
            }}
          />
        </TransLayer>
      )}
    </div>
  );
};

/* 기본값 정의 */
Spinner.defaultProps = {
  visible: false,
  color: "#06f",
  width: 100,
  height: 100,
};

/* 데이터 타입 설정 */
Spinner.propTypes = {
  visible: PropTypes.bool.isRequired,
  color: PropTypes.string,
  width: PropTypes.number,
  height: PropTypes.number,
};
// React.memo()를 사용하여 함수형 컴포넌트의 리렌더링 성능을 최적화
export default React.memo(Spinner);
```