결과

exam-13

-- 날짜 선택 --

번호	년도	월	교통사고 건수	사망자 수	부상자 수
1	2005년	1월	15,494건	504건	25,413명
2	2005년	2월	13,244건	431건	21,635명
3	2005년	3월	16,580건	477건	25,550명
4	2005년	4월	17,817건	507건	28,131명
5	2005년	5월	19,085건	571건	29,808명
6	2005년	6월	18,092건	476건	28,594명
7	2005년	7월	18,675건	528건	29,984명
8	2005년	8월	19,035건	562건	31,603명
9	2005년	9월	18,759건	577건	29,831명
10	2005년	10월	19,757건	639건	31,597명
11	2005년	11월	19,129건	574건	30,337명
12	2005년	12월	18,504건	530건	29,750명
13	2006년	1월	14,971건	420건	24,533명
14	2006년	2월	14,270건	373건	22,903명
15	2006년	3월	16,767건	465건	26,013명
16	2006년	4월	17,948건	469건	28,725명
17	2006년	5월	19,140건	531건	30,279명

exam-13

2012년도

번호	년도	월	교통사고 건수	사망자 수	부상자 수
85	2012년	1월	16,818건	418건	26,281명
86	2012년	2월	16,656건	393건	25,998명
87	2012년	3월	18,255건	403건	27,899명
88	2012년	4월	19,372건	483건	29,628명

89	2012년	5월	19,672건	444건	30,163명
90	2012년	6월	18,854건	476건	28,314명
91	2012년	7월	19,333건	416건	29,642명
92	2012년	8월	18,407건	409건	29,051명
93	2012년	9월	19,234건	486건	29,462명
94	2012년	10월	19,557건	533건	29,759명
95	2012년	11월	19,750건	508건	30,054명
96	2012년	12월	17,748건	423건	28,314명
합계			223,656건	5,392명	344,565명

AccidentSlice.js

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

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

    try {result = await axios.get("http://localhost:3004/traffic_acc", {
            params: {
              year: payload,
            }
          })} 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 AccidentSlice = createSlice({
  name: "accident",
  initialState: {
    data: null,
    loading: false,
    error: null,
  },
  // 내부 action 및 동기 action
```

```
  reducer: {},
  // 외부 action 및 비동기 (Ajax용)
  extraReducers: {
    [getAccidentList.pending]: (state, { payload }) => {
      return { ...state, loading: true };
    },
    [getAccidentList.fulfilled]: (state, { payload }) => {
      return {
        data: payload?.data,
        loading: false,
        error: null,
      };
    },
    [getAccidentList.rejected]: (state, { payload }) => {
      return {
        data: payload?.data,
        loading: false,
        error: {
          code: payload?.status ? payload.status : 500,
          message: payload?.statusText ? payload.statusText : "Server Error",
        },
      };
    },
  },
});
// 리듀서 객체 내보내기
export default AccidentSlice.reducer;
```

store.js

```
import { configureStore } from "@reduxjs/toolkit";
import AccidentSlice from "../slices/AccidentSlice";

const store = configureStore({
  reducer: {
    accident: AccidentSlice,
  },
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware({ serializableCheck: false }),
  devTools: true,
});

export default store;
```

index.js

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

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

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </Provider>
  </React.StrictMode>
);
```

App.js

```
import React from "react";
import Accident from "./pages/Accident";

function App() {
  return (
    <div>
      <h1>exam-13</h1>
      <hr />
      <Accident />
    </div>
  );
}

export default App;
```

accident.js

```
import React from "react";
/* 마운트 완료 후 사용될 것 */
import useMountedRef from "../hooks/useMounterRef";
import { useSelector, useDispatch } from "react-redux";
import { getAccidentList } from "../slices/AccidentSlice";
```

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

import styled from "styled-components";

/* 드롭다운을 배치하기 위한 박스 */
const SelectContainer = styled.div`
  position: sticky;
  top: 0;
  background-color: #fff;
  border-top: 1px solid #eee;
  border-bottom: 1px solid #eee;
  padding: 10px 0;
  margin: 0;

  select {
    margin-right: 15px;
    font-size: 16px;
    padding: 5px 10px;
  }
`;
const Tr = styled.tr`
td{
  background-color: #f0f6f9;
   color : #168;
  font-weight: bolder;
}
`
const Accident = () => {
  // hook을 통해 slice가 관리하는 상태값 가져오기
  const { data, loading, error } = useSelector((state) => state.accident);
  let acSum = 0;
  let deSum = 0;
  let inSum = 0;
  // dispatch 함수 생성
  const dispatch = useDispatch();
  /* 변수화 */
  const mountedRef = useMountedRef();
  // 이 컴포넌트가 화면에 마운트 되었는지를 확인하기 위한 hook

  const [year, setYear] = React.useState("");

  // 컴포넌트가 마운트되면 데이터 조회를 위한 액션함수를 디스패치 함
  React.useEffect(() => {
    if(mountedRef.current){
      dispatch(getAccidentList(year));
    }else {
      dispatch(getAccidentList());
    }
  }, [dispatch, year, mountedRef]);

  /* 드롭다운 선택 변경 시 호출되는 이벤트 */
```

```
  const onSelectChange = React.useCallback(
    (e) => {
      e.preventDefault();

      // 드롭다운의 입력값 취득
      const current = e.target;
      const value = current[current.selectedIndex].value;
      let newState = null;
      // 기존의 상태값을 그대로 복사한 상태에서
      //  드롭다운의 name속성과 일치하는 key에 대한 value를 수정

        newState = setYear(value);


      // 갱신된 상태값 확인
      console.log(newState);
    },
    []
  );

  return (
    <div>
      {/* 로딩바 */}
      <Spinner visible={loading} />

      {/* 검색 조건 드롭다운 박스 */}
      <SelectContainer>
        <select
          name="year"
          onChange={onSelectChange}
          style={{ border: "3px solid #168", borderRadius: "7%" }}
        >
          <option value="">-- 날짜 선택 --</option>
          {[...new Array(2018 - 2005 + 1)].map((v, i) => {
            return <option value={2005 + i}>{2005 + i}년도</option>;
          })}
        </select>
      </SelectContainer>
      {error ? (
        <Error error={error} />
      ) : (
        <Table>
          <thead>
            <tr>
              <th>번호</th>
              <th>년도</th>
              <th>월</th>
              <th>교통사고 건수</th>
              <th>사망자 수</th>
              <th>부상자 수</th>
            </tr>
          </thead>
          <tbody>
            {data &&
```

```
              data.map(({ id, year, month, accident, death, injury }, i) => {
                acSum += accident;
                deSum += death;
                inSum += injury;
                return (
                  <tr key={i}>
                    <td>{id}</td>
                    <td>{year}년</td>
                    <td>{month}월</td>
                    <td>{accident.toLocaleString()}건</td>
                    <td>{death.toLocaleString()}건</td>
                    <td>{injury.toLocaleString()}명</td>
                  </tr>
                );
              })}
            <Tr>
              <td colSpan={3}>합계</td>
              <td>{acSum.toLocaleString()}건</td>
              <td>{deSum.toLocaleString()}명</td>
              <td>{inSum.toLocaleString()}명</td>
            </Tr>
          </tbody>
        </Table>
      )}
    </div>
  );
};

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