

## 22.11.16-Redux-CRUD(1) 연습문제 홍승택

### 파일 구조

```
src
├── components
│   ├── ErrorView.js
│   ├── MenuLink.js
│   ├── Spinner.js
│   ├── Table.js
│   └── TableEx.js
├── helper
│   └── ReduxHelper.js
├── pages
│   ├── StudentAdd.js
│   ├── StudentEdit.js
│   ├── StudentList.js
│   └── StudentView.js
├── slices
│   ├── DepartmentSlice.js
│   ├── ProfessorSlice.js
│   └── StudentSlice.js
├── App.js
├── index.js
└── store.js
```

/backend/data.json

.env.development

.env.production

### pages

#### StudentAdd

```
import React, { memo, useCallback, useEffect, useMemo } from 'react';
/** 리덕스 사용 */
import { useSelector, useDispatch } from 'react-redux';

/** 파라미터, 네비게이터 사용 */
import { useParams, useNavigate } from 'react-router-dom';

/** 슬라이스 */
import { getDepartmentList } from '../slices/DepartmentSlice';
import { getList } from '../slices/ProfessorSlice';
```

```

import { postItem } from '../slices/StudentSlice';

/** 컴포넌트 */
import Spinner from '../components/Spinner';
import TableEx from '../components/TableEx';
import ErrorView from '../components/ErrorView';

const StudentAdd = memo(() => {
  /** path 파라미터 받기 */
  const { id } = useParams();

  /** 리덕스 관련 초기화 */
  const dispatch = useDispatch();
  const { loading: loading1, error: error1 } = useSelector(state =>
state.StudentSlice);
  const { data: professor, loading: loading2, error: error2 } =
useSelector(state => state.ProfessorSlice);
  const { data: department, loading: loading3, error: error3 } =
useSelector(state => state.DepartmentSlice);

  /** 데이터 가져오기 */
  useEffect(() => {
    dispatch(getDepartmentList());
    dispatch(getList());
  }, []);

  /** 페이지 강제 이동을 위한 Navigate */
  const navigate = useNavigate();

  /** form의 submit이 눌렸을 때 호출될 이벤트 핸들러 */
  const onSubmitAdd = useCallback(e => {
    e.preventDefault();
    const current = e.currentTarget;

    dispatch(postItem({
      id: null,
      name: current.name.value,
      userid: current.userid.value,
      grade: current.grade.value,
      idnum: current.idnum.value.replace("-", ""),
      birthdate: current.birthdate.value,
      tel: current.tel.value,
      height: current.height.value,
      weight: current.weight.value,
      deptno: current.deptno.value,
      profno: current.profno.value
    })).then((result) => {
      navigate(`/student_view/${result.payload.id}`);
    });
  }, []);
  return (
    <>
    <Spinner loading={loading1 || loading2} />

```

```

{
  error1 || error2 || error3 ? (
    <ErrorView error={error1 || error2 || error3} />
  ) : (
    <form onSubmit={onSubmitAdd}>

      <TableEx>
        <colgroup>
          <col width='120' />
          <col />
        </colgroup>
        <tbody>
          <tr>
            <th>이름</th>
            <td className='inputWrapper'>
              <input type="text" className="field"
name='name' />
            </td>
          </tr>

          <tr>
            <th>접속아이디</th>
            <td className='inputWrapper'>
              <input type="text" className="field"
name='userid' />
            </td>
          </tr>

          <tr>
            <th>학년</th>
            <td className='inputWrapper'>
              <select className="field" name='grade' >
                <option value="">-- 학년을 선택해주세요 -
- </option>

                <option value="1">1학년</option>
                <option value="2">2학년</option>
                <option value="3">3학년</option>
                <option value="4">4학년</option>
              </select>
            </td>
          </tr>

          <tr>
            <th>주민번호</th>
            <td className='inputWrapper'>
              <input type="text" className="field"
name='idnum' />
            </td>
          </tr>

          <tr>
            <th>생년월일</th>
            <td className='inputWrapper'>
              <input type="date" className="field"
name='birthdate' />
            </td>
          </tr>
        </tbody>
      </TableEx>
    </form>
  )
}

```

```

        </td>
      </tr>
      <tr>
        <th>전화번호</th>
        <td className='inputWrapper'>
          <input type="text" className="field"
name='tel' />
        </td>
      </tr>

      <tr>
        <th>키</th>
        <td className='inputWrapper'>
          <input type="number" className="field"
name='height' />
        </td>
      </tr>

      <tr>
        <th>몸무게</th>
        <td className='inputWrapper'>
          <input type="number" className="field"
name='weight' />
        </td>
      </tr>

      <tr>
        <th>전공</th>
        <td className='inputWrapper'>
          <select name="deptno" className='field'>
            <option value="">--전공을 선택해주세요--
- </option>
            {
              department && department.map((v,
i) => {
                return (
                  <option key={v.id} value=
{v.id}>{v.dname}</option>
                );
              })
            }
          </select>
        </td>
      </tr>
      <tr>
        <th>담당교수</th>
        <td className='inputWrapper'>
          <select name="profno" className='field'>
            <option value="">--담당교수를 선택해주
세요--</option>
            {
              professor && professor.map((v, i)

```

```

=> {
    return (
      <option key={v.id} value=
{v.id}>{v.name}</option>
    );
  })
}
</select>
</td>
</tr>
</tbody>
</TableEx>

<div style={{ textAlign: 'center', marginTop: '20px' }}>
  <button type="submit">저장하기</button>
</div>
</form>
  )}
</>
);
});

export default StudentAdd;

```

## StudentEdit

```

import React, { memo, useCallback, useEffect, useMemo } from 'react';
/** 리덕스 사용 */
import { useSelector, useDispatch } from 'react-redux';

/** 파라미터, 네비게이터 사용 */
import { useParams, useNavigate } from 'react-router-dom';

/** 고용일자를 위한 dayjs */
import dayjs from 'dayjs';

/** 동기로 받아오는 형식을 위함 */
import { getCurrentData } from '../slices/StudentSlice';

/** 슬라이스 */
import { getDepartmentList } from '../slices/DepartmentSlice';
import { getList } from '../slices/ProfessorSlice';
import { getItem, putItem } from '../slices/StudentSlice';
/** 컴포넌트 */
import Spinner from '../components/Spinner';
import TableEx from '../components/TableEx';
import ErrorView from '../components/ErrorView';

const StudentEdit = memo(() => {
  /** path 파라미터 받기 */

```

```

    const { id } = useParams();

    /** 리덕스 관련 초기화 */
    const dispatch = useDispatch();
    const { data: student, loading: loading1, error: error1 } = useSelector(state
=> state.StudentSlice);
    const { data: professor, loading: loading2, error: error2 } =
useSelector(state => state.ProfessorSlice);
    const { data: department, loading: loading3, error: error3 } =
useSelector(state => state.DepartmentSlice);

    /** 데이터 가져오기 */
    useEffect(() => {
        dispatch(getCurrentData());
        dispatch(getDepartmentList());
        dispatch(getList());
    }, []);

    /** student 데이터 변경에 따른 사이드 이펙트 처리 */
    const item = useMemo(() => {
        if (student) {
            return student.find((v, i) => v.id == id);
        } else {
            dispatch(getItem({ id: id }));
        }
    }, [student]);
    /** 페이지 강제 이동을 위한 Navigate */
    const navigate = useNavigate();

    /** form의 submit이 눌렸을 때 호출될 이벤트 핸들러 */
    const onStudentSubmit = useCallback(e => {
        e.preventDefault();
        const current = e.currentTarget;

        dispatch(putItem({
            id: current.id.value,
            name: current.name.value,
            userid: current.userid.value,
            grade: current.grade.value,
            idnum: current.idnum.value.replace("-", ""),
            birthdate: current.birthdate.value,
            tel: current.tel.value,
            height: current.height.value.replace("cm", ""),
            weight: current.weight.value.replace("kg", ""),
            deptno: current.deptno.value,
            profno: current.profno.value
        })).then((result) => {
            navigate(`/student_view/${result.payload.id}`);
        });
    }, []);
    return (
        <>
        <Spinner loading={loading1 || loading2} />
        {

```

```

error1 || error2 || error3 ? (
  <ErrorView error={error1 || error2 || error3} />
) : (
  <form onSubmit={onStudentSubmit}>
    <input type="hidden" name="id" defaultValue={item?.id} />
    <TableEx>
      <colgroup>
        <col width='120' />
        <col />
      </colgroup>
      <tbody>
        <tr>
          <th>이름</th>
          <td className='inputWrapper'>
            <input type="text" className="field"
name='name' defaultValue={item?.name} />
          </td>
        </tr>

        <tr>
          <th>접속아이디</th>
          <td className='inputWrapper'>
            <input type="text" className="field"
name='userid' defaultValue={item?.userid} />
          </td>
        </tr>

        <tr>
          <th>학년</th>
          <td className='inputWrapper'>
            <select className="field" name='grade'
defaultValue={item?.grade} >
              <option value="1">1학년</option>
              <option value="2">2학년</option>
              <option value="3">3학년</option>
              <option value="4">4학년</option>
            </select>
          </td>
        </tr>

        <tr>
          <th>주민번호</th>
          <td className='inputWrapper'>
            <input type="text" className="field"
name='idnum' defaultValue={item.idnum.substring(0,6)+"-"+item.idnum.substring(6)}
/>
          </td>
        </tr>

        <tr>
          <th>생년월일</th>
          <td className='inputWrapper'>
            <input type="date" className="field"

```

```

name='birthdate' defaultValue={dayjs(item?.birthdate).format("YYYY-MM-DD")} />
      </td>
    </tr>

    <tr>
      <th>전화번호</th>
      <td className='inputWrapper'>
        <input type="text" className="field"
name='tel' defaultValue={item?.tel} />
      </td>
    </tr>

    <tr>
      <th>키</th>
      <td className='inputWrapper'>
        <input type="text" className="field"
name='height' defaultValue={item?.height+"cm"} />
      </td>
    </tr>

    <tr>
      <th>몸무게</th>
      <td className='inputWrapper'>
        <input type="text" className="field"
name='weight' defaultValue={item?.weight+"kg"} />
      </td>
    </tr>

    <tr>
      <th>전공</th>
      <td className='inputWrapper'>
        <select name="deptno" className='field'
defaultValue={item?.deptno} >
          {
            department && department.map((v,
i) => {
              return (
                <option key={v.id} value=
{v.id}>{v.dname}</option>
              );
            })
          }
        </select>
      </td>
    </tr>
    <tr>
      <th>담당교수</th>
      <td className='inputWrapper'>
        <select name="profno" className='field'
defaultValue={item?.profno} >
          {

```



```

        professor && professor.map((v, i)
=> {
            return (
                <option key={v.id} value=
{v.id}>{v.name}</option>
            );
        })
    }
    </select>
  </td>
</tr>
</tbody>
</TableEx>

    <div style={{ textAlign: 'center', marginTop: '20px' }}>
      <button type="submit">수정하기</button>
    </div>
  </form>
)}
</>
);
});

export default StudentEdit;

```

## StudentList

```

import React, { memo, useEffect, useCallback } from "react";
import { NavLink, useNavigate } from "react-router-dom";

import Spinner from "../components/Spinner";
import ErrorView from "../components/ErrorView";
import Table from "../components/Table";
import dayjs from "dayjs";
import { useDispatch, useSelector } from "react-redux";
import { deleteItem, getList } from "../slices/StudentSlice";

import styled from "styled-components";

import { useQueryString } from "../hooks/useQueryString";

const ControlContainer = styled.form`
  position: sticky;
  top: 0;
  background-color: #fff;
  border-top: 1px solid #eee;
  border-bottom: 1px solid #eee;
  padding: 10px 0;

  .control {
    margin-right: 5px;

```

```

    display: inline-block;
    font-size: 16px;
    padding: 7px 10px 5px 10px;
    border: 1px solid #ccc;
  }

  .clickable {
    background-color: #fff;
    color: #000;
    text-decoration: none;
    cursor: pointer;

    &:hover {
      background-color: #06f2;
    }

    &:active {
      transform: scale(0.9, 0.9);
    }
  }
};

const StudentList = memo(() => {
  // queryString 추출
  const { keyword } = useQueryString();
  /** 리덕스 초기화 */
  const dispatch = useDispatch();
  const { data, loading, error } = useSelector((state) => state.StudentSlice);

  /** 최초 마운트 시 리덕스를 통한 목록 조회 */
  React.useEffect(() => {
    dispatch(getList({
      keyword: keyword
    }));
  }, [keyword]);

  /** 페이지 강제 이동 */
  const navigate = useNavigate();
  /** 검색 이벤트 */
  const onSubmitSearch = useCallback(e => {
    e.preventDefault();
    const current = e.currentTarget;
    const keyword = current.keyword;

    let redirectUrl = keyword.value ? `/student/?keyword=${keyword.value}` :
    '/student'
    navigate(redirectUrl);
  },[navigate])

  /** 삭제 버튼에 대한 이벤트 리스너 */
  const onStudentItemDelete = useCallback(e => {
    e.preventDefault();
    const current = e.currentTarget;
    const {id, name} = current.dataset;

```

```

    if(window.confirm(`정말 ${name}을 삭제하시겠습니까?`)){
      dispatch(deleteItem({
        id:id
      })).then(({meta, payload}) => {
        navigate('/student');
      });
    }
  },[]);
  /** 수정 버튼에 대한 이벤트 리스너 */
  const onStudentEditClick = useCallback(e => {
    e.preventDefault();

    const current = e.currentTarget;
    const { id } = current.dataset;

    navigate(`/student_edit/${id}`);
  });
  return (
    <div>
      {/* 로딩바 */}
      <Spinner loading={loading} />

      {/* 검색폼 */}
      <ControlContainer onSubmit={onSubmitSearch}>
        <input type="text" name='keyword' className="control" />
        <button type='submit' className="control clickable">Search</button>
        <NavLink to = '/student_add' className="control clickable">학생정보 추가하
기</NavLink>
      </ControlContainer>

      {/* 조회결과 */}
      {error ? (
        <ErrorView error={error} />
      ) : (
        //Ajax 처리 결과 존재
        data && (
          <Table>
            <thead>
              <tr>
                <th>학번</th>
                <th>이름</th>
                <th>아이디</th>
                <th>학년</th>
                <th>주민번호</th>
                <th>생년월일</th>
                <th>전화번호</th>
                <th>키</th>
                <th>몸무게</th>
                <th>전공번호</th>
                <th>교수번호</th>
                <th>수정하기</th>
                <th>삭제하기</th>
              </tr>

```

```

    </thead>
    <tbody>
      {
        data.length > 0 ? (
          data.map(item => {
            return (
              <tr key={item.id}>
                <td>{item.id}</td>
                <td>
                  <NavLink to={` /student_view/${item.id}`}>{item.name}
</NavLink>
                </td>
                <td>{item.userid}</td>
                <td>{item.grade+"학년"}</td>
                <td>{item.idnum.substring(0,6) + "-*****"}</td>
                <td>{dayjs(item.birthdate).format("YYYY-MM-DD")}</td>
                <td>{item.tel}</td>
                <td>{item.height+"cm"}</td>
                <td>{item.weight+"kg"}</td>
                <td>{item.deptno}</td>
                <td>{item.profno}</td>
                <td>
                  <button type="button" data-id = {item.id} onClick=
{onStudentEditClick}>
                    수정하기
                  </button>
                </td>
                <td>
                  <button type="button" data-id={item.id} data-name=
{item.name} onClick={onStudentItemDelete}>
                    삭제하기
                  </button>
                </td>
              </tr>
            )
          })
        ) : (
          <tr>
            <td colspan='12' align='center'>
              검색결과가 없습니다.
            </td>
          </tr>
        )
      }
    </tbody>
  </Table>
)
})
</div>
);
});

export default StudentList;

```

## StudentView

```

import React, { memo, useCallback, useMemo, useEffect } from 'react';
import { NavLink, useParams, useNavigate } from "react-router-dom";
import { useSelector, useDispatch } from "react-redux";
import { getCurrentData, deleteItem, getItem } from "../slices/StudentSlice";
import dayjs from "dayjs";

import Spinner from "../components/Spinner";
import ErrorView from "../components/ErrorView";
import Table from "../components/TableEx";
const StudentView = memo(() => {
  /** path 파라미터 */
  const { id } = useParams();
  /** 리덕스 초기화 */
  const dispatch = useDispatch();
  const { data, loading, error } = useSelector((state) => state.StudentSlice);

  /** 렌더링 시 데이터 가져오기 */
  useEffect(() => {
    dispatch(getItem({ id: id }));
  }, []);

  /** data 값의 변경에 따른 사이드 이펙트 처리 */
  const item = useMemo(() => {
    if (data) {
      return data.find((v, i) => v.id == id);
    } else {
      dispatch(getItem({ id: id }));
    }
  }, [data]);

  /** 페이지 강제 이동을 위한 navigate */
  const navigate = useNavigate();

  /** 삭제 버튼에 대한 이벤트 리스너 */
  const onStudentItemDelete = useCallback(e => {
    e.preventDefault();

    const current = e.currentTarget;
    const { id, name } = current.dataset;

    if (window.confirm(`정말 ${name}을 삭제하시겠습니까?`)) {
      dispatch(deleteItem({
        id: id
      })).then(({ meta, payload }) => {
        navigate('/student');
      });
    }
  }, []);
  /** 수정 버튼 리스너 */
  const onStudentEdit = useCallback(e => {

```

```

    e.preventDefault();

    const current = e.currentTarget;
    const { id } = current.dataset;

    navigate(`/student_edit/${id}`);
  }, []);
  return (
    <div>
      <Spinner loading={loading} />

      {error ? (
        <ErrorView error={error} />
      ) : (
        item && (
          <div>
            <Table>
              <colgroup>
                <col width='120' />
                <col />
              </colgroup>

              <tbody>
                <tr>
                  <th>학번</th>
                  <td>{item.id}</td>
                </tr>

                <tr>
                  <th>이름</th>
                  <td>{item.name}</td>
                </tr>

                <tr>
                  <th>아이디</th>
                  <td>{item.userid}</td>
                </tr>

                <tr>
                  <th>학년</th>
                  <td>{item.grade+"학년"}</td>
                </tr>

                <tr>
                  <th>주민번호</th>
                  <td>{item.idnum.substring(0,6)+"-*****"}
            </td>
          </tr>

          <tr>
            <th>생년월일</th>
            <td>{dayjs(item.birthdate).format("YYYY-MM-DD")}</td>

```

```

        </tr>

        <tr>
            <th>전화번호</th>
            <td>{item.tel}</td>
        </tr>

        <tr>
            <th>키</th>
            <td>{item.height+"cm"}</td>
        </tr>

        <tr>
            <th>몸무게</th>
            <td>{item.weight+"kg"}</td>
        </tr>

        <tr>
            <th>전공번호</th>
            <td>{item.deptno}</td>
        </tr>

        <tr>
            <th>교수번호</th>
            <td>{item.profno}</td>
        </tr>
    </tbody>
</Table>
<div style={{ textAlign: 'center' }}>
    <NavLink to="/student">목록</NavLink>
    &nbsp;|&nbsp;
    <NavLink to="/student_add">등록</NavLink>
    &nbsp;|&nbsp;
    <NavLink to="/student_edit" data-id={item.id} onClick=
{onStudentEdit}>수정</NavLink>
    &nbsp;|&nbsp;
    <NavLink to="/student/#!" data-id={item.id} data-name=
{item.name} onClick={onStudentItemDelete}>삭제</NavLink>
</div>
</div>
    )
  })
</div>
);
});

export default StudentView;

```

slices

## DepartmentSlice.js

```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";
import { pending, fulfilled, rejected } from '../helper/ReduxHelper';

export const getDepartmentList =
createAsyncThunk("DepartmentSlice/getDepartmentList", async(payload,
{rejectWithValue}) => {
  let result = null;
  const URL = process.env.REACT_APP_API_DEPARTMENT_LIST;
  try{
    const response = await axios.get(URL);
    result= response.data;
  } catch(e){
    result = rejectWithValue(e.response);
  }

  return result;
});

const DepartmentSlice = createSlice({
  name: 'DepartmentSlice',
  initialState: {
    data: null,
    loading: false,
    error: null
  },
  extraReducers: {
    [getDepartmentList.pending] : pending,
    [getDepartmentList.fulfilled] : fulfilled,
    [getDepartmentList.rejected] : rejected
  }
});

export default DepartmentSlice.reducer;
```

## ProfessorSlice.js

```
/**
 * 이 단계를 수행하기 위해서는 ../helper/ReduxHelper.js의 작업이 선행되어야 한다.
 */
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";
import { pending, fulfilled, rejected } from "../helper/ReduxHelper";
import { cloneDeep } from "lodash";

/** 다중행 데이터 조회를 위한 비동기 함수 */
export const getList = createAsyncThunk("ProfessorSlice/getList", async (payload,
{ rejectWithValue }) => {
```



```
let result = null;
const URL = process.env.REACT_APP_API_PROFESSOR_LIST;

let params = null;

if (payload?.keyword) {
  params = {
    name: payload.keyword,
  };
}

try {
  const response = await axios.get(URL, { params: params });
  result = response.data;
} catch (err) {
  result = rejectWithValue(err.response);
}

return result;
});

/** 단일행 데이터 조회를 위한 비동기 함수 */
export const getItem = createAsyncThunk("ProfessorSlice/getItem", async (payload, { rejectWithValue }) => {
  let result = null;

  // 환경설정 파일에 정의된 URL에서 ':id' 부분을 찾아 payload를 통해 전달된 일련번호로 치환
  const URL = process.env.REACT_APP_API_PROFESSOR_ITEM.replace(":id", payload.id);

  try {
    const response = await axios.get(URL);
    result = response.data;
  } catch (err) {
    result = rejectWithValue(err.response);
  }

  return result;
});

/** 데이터 저장을 위한 비동기 함수 */
export const postItem = createAsyncThunk("ProfessorSlice/postItem", async (payload, { rejectWithValue }) => {
  let result = null;
  const URL = process.env.REACT_APP_API_PROFESSOR_LIST;

  try {
    const response = await axios.post(URL, {
      name: payload.name,
      userid: payload.userid,
      position: payload.position,
      sal: payload.sal,
    });
  } catch (err) {
    result = rejectWithValue(err.response);
  }

  return result;
});
```

```
        hiredate: payload.hiredate,
        comm: payload.comm,
        deptno: payload.deptno
    });
    result = response.data;
} catch (err) {
    result = rejectWithValue(err.response);
}

return result;
});

/** 데이터 수정을 위한 비동기 함수 */
export const putItem = createAsyncThunk("ProfessorSlice/putItem", async (payload,
{ rejectWithValue }) => {
    let result = null;

    // 환경설정 파일에 정의된 URL에서 ':id' 부분을 찾아 payload를 통해 전달된 일련번호로
    치환
    const URL = process.env.REACT_APP_API_PROFESSOR_ITEM.replace(":id", payload.id);

    try {
        const response = await axios.put(URL, {
            name: payload.name,
            userid: payload.userid,
            position: payload.position,
            sal: payload.sal,
            hiredate: payload.hiredate,
            comm: payload.comm,
            deptno: payload.deptno
        });
        result = response.data;
    } catch (err) {
        result = rejectWithValue(err.response);
    }

    return result;
});

/** 데이터 삭제를 위한 비동기 함수 */
export const deleteItem = createAsyncThunk("ProfessorSlice/deleteItem", async
(payload, { rejectWithValue }) => {
    let result = null;

    // 환경설정 파일에 정의된 URL에서 ':id' 부분을 찾아 payload를 통해 전달된 일련번호로
    치환
    const URL = process.env.REACT_APP_API_PROFESSOR_ITEM.replace(":id", payload.id);

    try {
        const response = await axios.delete(URL);
        result = response.data;
    } catch (err) {
        result = rejectWithValue(err.response);
    }
}
```

```

    return result;
  });

const ProfessorSlice = createSlice({
  name: "ProfessorSlice",
  initialState: {
    data: null,
    loading: false,
    error: null,
  },

  reducers: {
    getCurrentData: (state, action) => {
      return state;
    },
  },

  extraReducers: {
    /** 다중행 데이터 조회를 위한 액션 함수 */
    [getList.pending]: pending,
    [getList.fulfilled]: fulfilled,
    [getList.rejected]: rejected,
    /** 단일행 데이터 조회를 위한 액션 함수 */
    [getItem.pending]: pending,
    [getItem.fulfilled]: (state, { meta, payload }) => {
      return {
        // 전체적으로 데이터가 배열이지만, 단일행 조회의 경우 단건의 데이터만 응답결과로
        // 수신되므로,
        // 배열로 묶어서 처리한다.
        data: [payload],
        loading: false,
        error: null,
      };
    },
    [getItem.rejected]: rejected,
    /** 데이터 저장을 위한 액션 함수 */
    [postItem.pending]: pending,
    [postItem.fulfilled]: (state, { meta, payload }) => {
      // 기존의 상태값을 복사한다. (원본이 JSON이므로 깊은 복사를 수행해야 한다.)
      const data = cloneDeep(state.data);
      console.log(data);

      // 새로 저장된 결과를 기존 상태값 배열의 맨 뒤에 추가한다.
      data.push(payload);

      return {
        data: data,
        loading: false,
        error: null,
      };
    },
    [postItem.rejected]: rejected,
  },
});

```

```

/** 데이터 수정을 위한 액션 함수 */
[putItem.pending]: pending,
[putItem.fulfilled]: (state, { meta, payload }) => {
  const data = cloneDeep(state.data);

  const targetId = data.findIndex((v, i) => v.id == meta.arg.id);
  console.log(targetId);

  data.splice(targetId, 1, payload);

  return {
    data: data,
    loading: false,
    error: null,
  };
},
[putItem.rejected]: rejected,
/** 데이터 삭제를 위한 액션 함수 */
[deleteItem.pending]: pending,
[deleteItem.fulfilled]: (state, { meta, payload }) => {
  // 기존의 상태값을 복사한다. (원본이 JSON 이므로 깊은 복사를 수행해야 한다.)
  const data = cloneDeep(state.data);

  // id값이 일치하는 항목의 배열 인덱스를 찾는다.
  const targetId = data.findIndex((v, i) => v.id == meta.arg.id);
  console.log(targetId);

  // 해당 인덱스의 원소를 삭제한다.
  data.splice(targetId, 1);

  return {
    data: data,
    loading: false,
    error: null,
  };
},
[deleteItem.rejected]: rejected,
});

export const { getCurrentData } = ProfessorSlice.actions;
export default ProfessorSlice.reducer;

```

## StudentSlice.js

```

/**
 * 이 단계를 수행하기 위해서는 ../helper/ReduxHelper.js의 작업이 선행되어야 한다.
 */
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";

```

```
import { pending, fulfilled, rejected } from "../helper/ReduxHelper";
import { cloneDeep } from "lodash";

/** 다중행 데이터 조회를 위한 비동기 함수 */
export const getList = createAsyncThunk("ProfessorSlice/getList", async (payload,
{ rejectWithValue }) => {
  let result = null;
  const URL = process.env.REACT_APP_API_PROFESSOR_LIST;

  let params = null;

  if (payload?.keyword) {
    params = {
      name: payload.keyword,
    };
  }

  try {
    const response = await axios.get(URL, { params: params });
    result = response.data;
  } catch (err) {
    result = rejectWithValue(err.response);
  }

  return result;
});

/** 단일행 데이터 조회를 위한 비동기 함수 */
export const getItem = createAsyncThunk("ProfessorSlice/getItem", async (payload,
{ rejectWithValue }) => {
  let result = null;

  // 환경설정 파일에 정의된 URL에서 ':id' 부분을 찾아 payload를 통해 전달된 일련번호로
  치환
  const URL = process.env.REACT_APP_API_PROFESSOR_ITEM.replace(":id", payload.id);

  try {
    const response = await axios.get(URL);
    result = response.data;
  } catch (err) {
    result = rejectWithValue(err.response);
  }

  return result;
});

/** 데이터 저장을 위한 비동기 함수 */
export const postItem = createAsyncThunk("ProfessorSlice/postItem", async
(payload, { rejectWithValue }) => {
  let result = null;
  const URL = process.env.REACT_APP_API_PROFESSOR_LIST;
```

```
try {
  const response = await axios.post(URL, {
    name: payload.name,
    userid: payload.userid,
    position: payload.position,
    sal: payload.sal,
    hiredate: payload.hiredate,
    comm: payload.comm,
    deptno: payload.deptno
  });
  result = response.data;
} catch (err) {
  result = rejectWithValue(err.response);
}

return result;
});

/** 데이터 수정을 위한 비동기 함수 */
export const putItem = createAsyncThunk("ProfessorSlice/putItem", async (payload,
{ rejectWithValue }) => {
  let result = null;

  // 환경설정 파일에 정의된 URL에서 ':id' 부분을 찾아 payload를 통해 전달된 일련번호로
  치환
  const URL = process.env.REACT_APP_API_PROFESSOR_ITEM.replace(":id", payload.id);

  try {
    const response = await axios.put(URL, {
      name: payload.name,
      userid: payload.userid,
      position: payload.position,
      sal: payload.sal,
      hiredate: payload.hiredate,
      comm: payload.comm,
      deptno: payload.deptno
    });
    result = response.data;
  } catch (err) {
    result = rejectWithValue(err.response);
  }

  return result;
});

/** 데이터 삭제를 위한 비동기 함수 */
export const deleteItem = createAsyncThunk("ProfessorSlice/deleteItem", async
(payload, { rejectWithValue }) => {
  let result = null;

  // 환경설정 파일에 정의된 URL에서 ':id' 부분을 찾아 payload를 통해 전달된 일련번호로
  치환
  const URL = process.env.REACT_APP_API_PROFESSOR_ITEM.replace(":id", payload.id);
```

```

    try {
      const response = await axios.delete(URL);
      result = response.data;
    } catch (err) {
      result = rejectWithValue(err.response);
    }

    return result;
  });

const ProfessorSlice = createSlice({
  name: "ProfessorSlice",
  initialState: {
    data: null,
    loading: false,
    error: null,
  },

  reducers: {
    getCurrentData: (state, action) => {
      return state;
    },
  },

  extraReducers: {
    /** 다중행 데이터 조회를 위한 액션 함수 */
    [getList.pending]: pending,
    [getList.fulfilled]: fulfilled,
    [getList.rejected]: rejected,
    /** 단일행 데이터 조회를 위한 액션 함수 */
    [getItem.pending]: pending,
    [getItem.fulfilled]: (state, { meta, payload }) => {
      return {
        // 전체적으로 데이터가 배열이지만, 단일행 조회의 경우 단건의 데이터만 응답결과로
        // 수신되므로,
        // 배열로 묶어서 처리한다.
        data: [payload],
        loading: false,
        error: null,
      };
    },
    [getItem.rejected]: rejected,
    /** 데이터 저장을 위한 액션 함수 */
    [postItem.pending]: pending,
    [postItem.fulfilled]: (state, { meta, payload }) => {
      // 기존의 상태값을 복사한다. (원본이 JSON이므로 깊은 복사를 수행해야 한다.)
      const data = cloneDeep(state.data);
      console.log(data);

      // 새로 저장된 결과를 기존 상태값 배열의 맨 뒤에 추가한다.
      data.push(payload);

      return {

```

```

        data: data,
        loading: false,
        error: null,
    };
},
[postItem.rejected]: rejected,
/** 데이터 수정을 위한 액션 함수 */
[putItem.pending]: pending,
[putItem.fulfilled]: (state, { meta, payload }) => {
    const data = cloneDeep(state.data);

    const targetId = data.findIndex((v, i) => v.id == meta.arg.id);
    console.log(targetId);

    data.splice(targetId, 1, payload);

    return {
        data: data,
        loading: false,
        error: null,
    };
},
[putItem.rejected]: rejected,
/** 데이터 삭제를 위한 액션 함수 */
[deleteItem.pending]: pending,
[deleteItem.fulfilled]: (state, { meta, payload }) => {
    // 기존의 상태값을 복사한다. (원본이 JSON 이므로 깊은 복사를 수행해야 한다.)
    const data = cloneDeep(state.data);

    // id값이 일치하는 항목의 배열 인덱스를 찾는다.
    const targetId = data.findIndex((v, i) => v.id == meta.arg.id);
    console.log(targetId);

    // 해당 인덱스의 원소를 삭제한다.
    data.splice(targetId, 1);

    return {
        data: data,
        loading: false,
        error: null,
    };
},
[deleteItem.rejected]: rejected,
},
});

export const { getCurrentData } = ProfessorSlice.actions;
export default ProfessorSlice.reducer;

```

## 기본 구성요소



## App.js

```

import React, { memo } from "react";
import { NavLink, Routes, Route } from "react-router-dom";
import MenuLink from "../components/MenuLink";
import ProfessorList from "../pages/ProfessorList";
import ProfessorAdd from "../pages/ProfessorAdd";
import ProfessorView from "../pages/ProfessorView";
import ProfessorEdit from "../pages/ProfessorEdit";
import StudentAdd from "../pages/StudentAdd";
import StudentEdit from "../pages/StudentEdit";
import StudentList from "../pages/StudentList";
import StudentView from "../pages/StudentView";

const App = memo(() => {
  return (
    <div>
      <nav>
        <NavLink to="/" exact="true" style={{ textDecoration: 'none', color:
'black' }}><h1>Redux-CRUD 연습문제</h1></NavLink>
        <hr />
        <MenuLink to="/professor">Professor</MenuLink>
        <MenuLink to="/student">Student</MenuLink>

        {/* <MenuLink to = '/student'>Student</MenuLink>&nbsp;|
        <MenuLink to = '/traffic_acc'>Traffic</MenuLink>&nbsp; */}
      </nav>
      <hr />

      <Routes>
        {/* professor */}
        <Route path="/professor" element={<ProfessorList />} />
        <Route path="/professor_add" element={<ProfessorAdd />} />
        <Route path="/professor_view/:id" element={<ProfessorView />} />
        <Route path="/professor_edit/:id" element={<ProfessorEdit />} />

        {/* student */}
        <Route path="/student" element={<StudentList />} />
        <Route path="/student_add" element={<StudentAdd />} />
        <Route path="/student_view/:id" element={<StudentView />} />
        <Route path="/student_edit/:id" element={<StudentEdit />} />
      </Routes>
    </div>
  );
});

export default App;

```

## index.js

```
import React from "react";
import ReactDOM from "react-dom/client";

/**/
import App from "./App";
/**/
import App from "./Test";
/**/

import { BrowserRouter } from "react-router-dom";
import { Provider } from "react-redux";
import store from "./store";

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

store.js

```
import { configureStore } from "@reduxjs/toolkit";
import ProfessorSlice from "./slices/ProfessorSlice";
import DepartmentSlice from "./slices/DepartmentSlice";
import StudentSlice from "./slices/StudentSlice";

const store = configureStore({
  reducer: {
    ProfessorSlice: ProfessorSlice,
    DepartmentSlice: DepartmentSlice,
    StudentSlice: StudentSlice
  },
});
export default store;
```

## 실행화면

---

List

ViewEditEdit 후 View

localhost:3000/student\_view/10101

Redux-CRUD 연습문제

Professor | Student

학번	10101
이름	전민하
아이디	jun123
학년	4학년
주민번호	790702-*****
생년월일	1979-07-02
전화번호	051)781-2158
키	11cm
몸무게	66kg
전공번호	101
교수번호	9910

목록 | 등록 | 수정 | 삭제

Edit 후 List

localhost:3000/student

Redux-CRUD 연습문제

Professor | Student

Search

학생정보 추가하기

학번	이름	아이디	학년	주민번호	생년월일	전화번호	키	몸무게	전공번호	교수번호	수정하기	삭제하기
10101	전민하	jun123	4학년	790702-*****	1979-07-02	051)781-2158	11cm	66kg	101	9910	수정하기	삭제하기
10102	백민걸	ansei414	1학년	840516-*****	1984-05-16	055)261-8947	168cm	52kg	101		수정하기	삭제하기
10103	김영균	mandu	3학년	810321-*****	1981-03-21	051)824-9637	170cm	88kg	101	9906	수정하기	삭제하기
10104	최소걸	gomo00	2학년	800412-*****	1980-04-12	055)418-9627	161cm	42kg	101	9907	수정하기	삭제하기
10105	윌슨친	youjin12	2학년	830121-*****	1983-01-21	02)312-9838	171cm	54kg	101	9907	수정하기	삭제하기
10106	서정진	seoly	1학년	851129-*****	1985-11-29	051)239-4861	186cm	72kg	101		수정하기	삭제하기
10107	이광훈	huriky	4학년	810913-*****	1981-09-13	055)736-4981	175cm	92kg	101	9903	수정하기	삭제하기
10108	클린스키	cleansky	2학년	810819-*****	1981-08-19	055)248-3679	162cm	72kg	101	9907	수정하기	삭제하기
10201	김진영	simply	2학년	820606-*****	1982-06-06	055)419-6328	164cm	48kg	102	9905	수정하기	삭제하기
10202	윤우성	yousuk	4학년	770912-*****	1977-09-12	051)724-9618	177cm	92kg	102	9905	수정하기	삭제하기
10203	한나리	hanal	1학년	850109-*****	1985-01-09	055)296-3784	160cm	68kg	102		수정하기	삭제하기
10204	삼보름	samba7	3학년	790402-*****	1979-04-02	053)487-2698	171cm	70kg	102	9905	수정하기	삭제하기
20101	이종훈	dais	1학년	831210-*****	1983-12-10	055)426-1752	172cm	64kg	201		수정하기	삭제하기
20102	반동진	ping2	1학년	851124-*****	1985-11-24	051)742-6384	182cm	70kg	201		수정하기	삭제하기
20103	김진걸	lovely	2학년	830228-*****	1983-02-28	052)175-3941	166cm	51kg	201	9902	수정하기	삭제하기
20104	프랭클	rader214	1학년	841214-*****	1984-12-14	02)785-6984	184cm	62kg	201		수정하기	삭제하기
20105			학년	*****	Invalid Date		cm	kg			수정하기	삭제하기

Delete

localhost:3000/student

Redux-CRUD 연습문제

Professor | Student

Search

학생정보 추가하기

localhost:3000 내용:  
정말 조영훈을 삭제하시겠습니까?

확인 취소

학번	이름	아이디	학년	주민번호	생년월일	전화번호	키	몸무게	전공번호	교수번호	수정하기	삭제하기
10101	전민하	jun123	4학년	790702-*****	1979-07-02	051)781-2158	11cm	66kg	101	9910	수정하기	삭제하기
10102	백민걸	ansei414	1학년	840516-*****	1984-05-16	055)261-8947	168cm	52kg	101		수정하기	삭제하기
10103	김영균	mandu	3학년	810321-*****	1981-03-21	051)824-9637	170cm	88kg	101	9906	수정하기	삭제하기
10104	최소걸	gomo00	2학년	800412-*****	1980-04-12	055)418-9627	161cm	42kg	101	9907	수정하기	삭제하기
10105	윌슨친	youjin12	2학년	830121-*****	1983-01-21	02)312-9838	171cm	54kg	101	9907	수정하기	삭제하기
10106	서정진	seoly	1학년	851129-*****	1985-11-29	051)239-4861	186cm	72kg	101		수정하기	삭제하기
10107	이광훈	huriky	4학년	810913-*****	1981-09-13	055)736-4981	175cm	92kg	101	9903	수정하기	삭제하기
10108	클린스키	cleansky	2학년	810819-*****	1981-08-19	055)248-3679	162cm	72kg	101	9907	수정하기	삭제하기
10201	김진영	simply	2학년	820606-*****	1982-06-06	055)419-6328	164cm	48kg	102	9905	수정하기	삭제하기
10202	윤우성	yousuk	4학년	770912-*****	1977-09-12	051)724-9618	177cm	92kg	102	9905	수정하기	삭제하기
10203	한나리	hanal	1학년	850109-*****	1985-01-09	055)296-3784	160cm	68kg	102		수정하기	삭제하기
10204	삼보름	samba7	3학년	790402-*****	1979-04-02	053)487-2698	171cm	70kg	102	9905	수정하기	삭제하기
20101	이종훈	dais	1학년	831210-*****	1983-12-10	055)426-1752	172cm	64kg	201		수정하기	삭제하기
20102	반동진	ping2	1학년	851124-*****	1985-11-24	051)742-6384	182cm	70kg	201		수정하기	삭제하기
20103	김진걸	lovely	2학년	830228-*****	1983-02-28	052)175-3941	166cm	51kg	201	9902	수정하기	삭제하기
20104	프랭클	rader214	1학년	841214-*****	1984-12-14	02)785-6984	184cm	62kg	201		수정하기	삭제하기

Professor | [Student](#)Professor | [Student](#)

Search

확장정보 추가하기

학번	이름	아이디	학년	주민번호	생년월일	전화번호	키	몸무게	잔고번호	교수번호	수강하기	삭제하기
10101	최은진	jun123	4학년	790702-*****	1979-07-02	051)781-2158	11cm	66kg	101	9910	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10102	박지현	anse1414	1학년	840516-*****	1984-05-16	053)681-8947	168cm	53kg	101		<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10103	김영준	mandu	3학년	810321-*****	1981-03-21	051)824-9637	170cm	88kg	101	9906	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10104	최소정	gomo00	2학년	800412-*****	1980-04-12	055)418-9627	161cm	42kg	101	9907	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10105	최은진	yujin12	2학년	830121-*****	1983-01-21	02)312-8638	171cm	54kg	101	9907	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10106	서은진	seoly	1학년	851129-*****	1985-11-29	051)239-4861	186cm	72kg	101		<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10107	이영준	hulky	4학년	810913-*****	1981-09-13	053)736-4981	175cm	93kg	101	9903	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10108	홍지현	cleansky	2학년	810819-*****	1981-08-19	055)448-3679	162cm	72kg	101	9907	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10201	최은진	simply	2학년	820606-*****	1982-06-06	055)419-6328	164cm	48kg	102	9905	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10202	홍은진	youtuk	4학년	770912-*****	1977-09-12	051)724-9618	177cm	92kg	102	9905	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10203	최나희	hanel	1학년	850109-*****	1985-01-09	055)286-3794	160cm	68kg	102		<a href="#">수강하기</a>	<a href="#">삭제하기</a>
10204	윤은준	samba7	3학년	790402-*****	1979-04-02	053)487-2680	171cm	70kg	102	9905	<a href="#">수강하기</a>	<a href="#">삭제하기</a>
20101	이영준	dais	1학년	831210-*****	1983-12-10	055)426-1752	172cm	64kg	201		<a href="#">수강하기</a>	<a href="#">삭제하기</a>
20102	박은진	ping2	1학년	851124-*****	1985-11-24	051)742-6384	182cm	79kg	201		<a href="#">수강하기</a>	<a href="#">삭제하기</a>
20103	최은진	lovely	2학년	830228-*****	1983-02-28	062)175-5941	166cm	51kg	201	9902	<a href="#">수강하기</a>	<a href="#">삭제하기</a>