



YouTube

NAVER 카페

구멍가게 코딩단

코드로 배우는 리액트

4. 리액트와 API 서버 통신

4장. 리액트와 API 서버 통신

- 대부분의 프론트엔드 기술은 비동기 통신을 필요로 함.
- 리액트도 서버 데이터를 Ajax를 활용 하여 기능 구현.
- 이번 장에서는 이전 장의 스프링 부트 서버로 Todo 기능 완성 예정.

개발목표

1. Axios 라이브러리를 이용한 서버와의 통신
2. useEffect()를 활용한 비동기 처리와 상태 변경
3. 커스텀 훅을 이용한 공통 코드 재사용하기
4. 컴포넌트에서 모달창을 이용해서 결과 보여주기

Index

- 4.1 개발 목표의 이해
- 4.2 Ajax 통신 처리
- 4.3 useEffect()
- 4.4 조회 화면의 버튼 처리
- 4.5 등록 컴포넌트와 모달 처리
- 4.6 조회 화면과 조회 컴포넌트
- 4.7 수정/삭제 처리

개발 목표의 이해

→ 완료된 기능

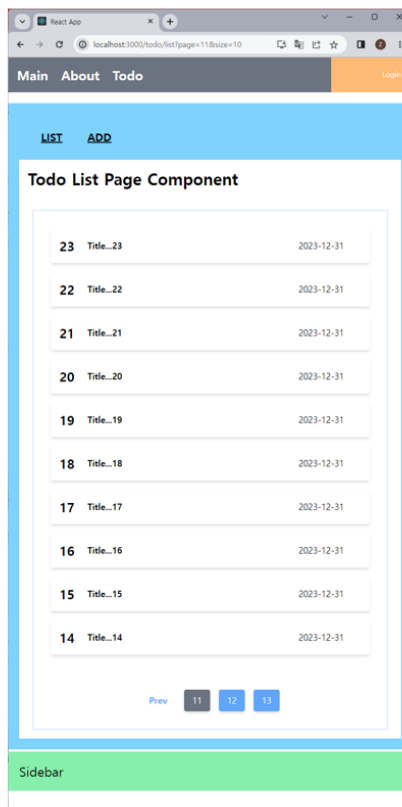
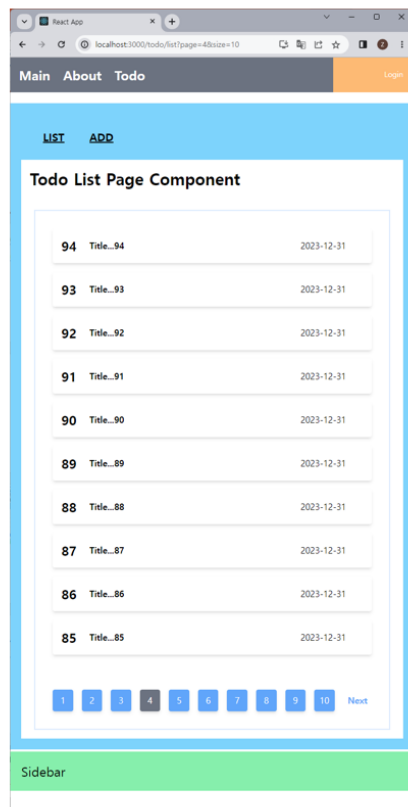
- ✓ React-Router를 이용하여 브라우저 주소 처리 완료.
- ✓ 스프링 부트를 활용하여 서버사이드 데이터 처리 가능.

→ 개발 목표

- ✓ 각 화면에 필요한 컴포넌트 추가하여 실제 내용 구현 진행.

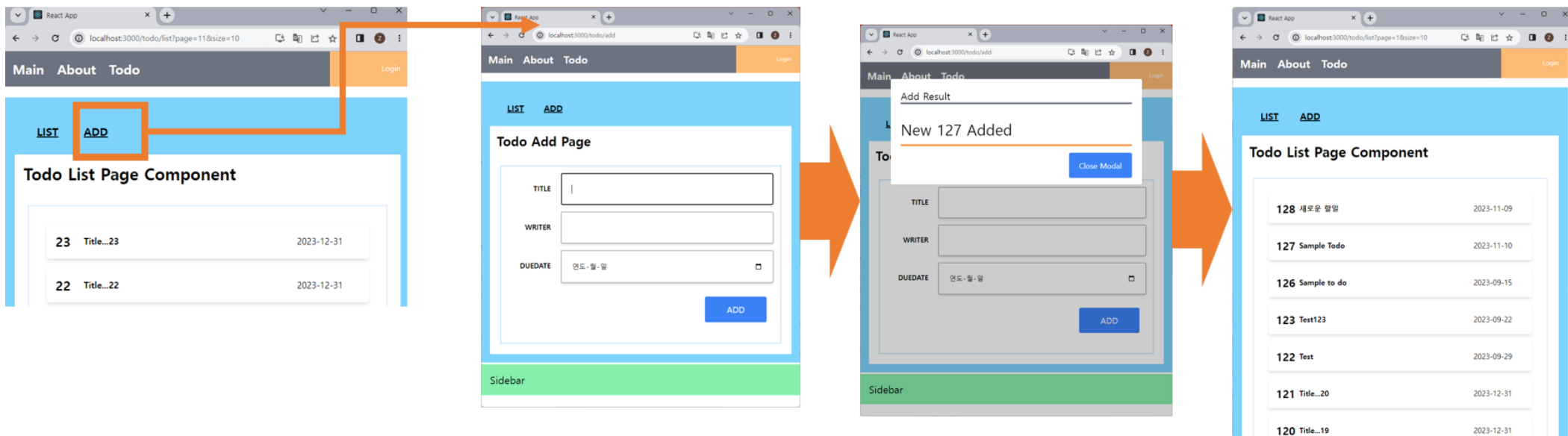
개발 목표의 이해

→ 목록 화면



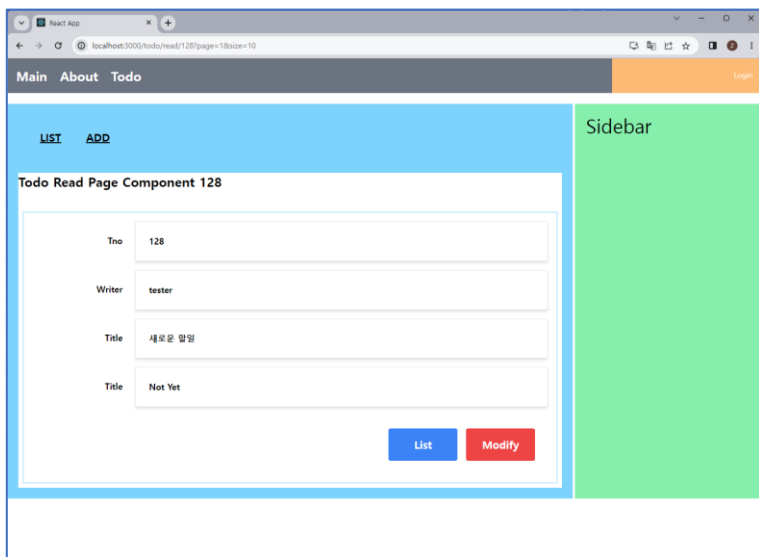
개발 목표의 이해

→ 입력 화면



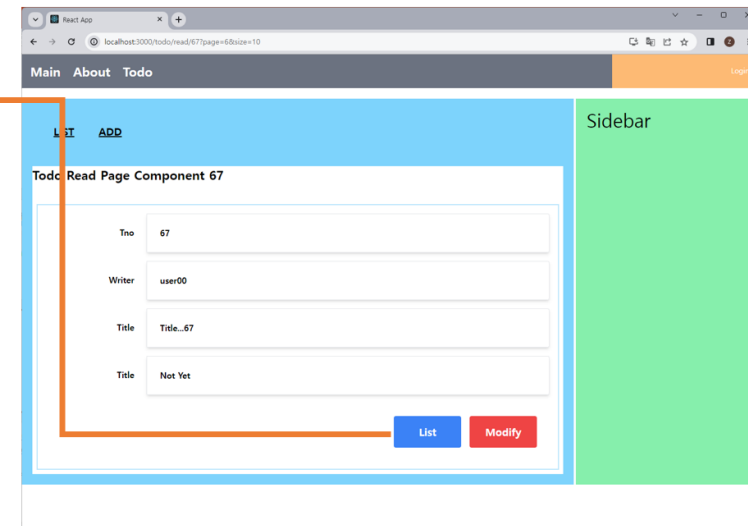
개발 목표의 이해

→ 조회 화면



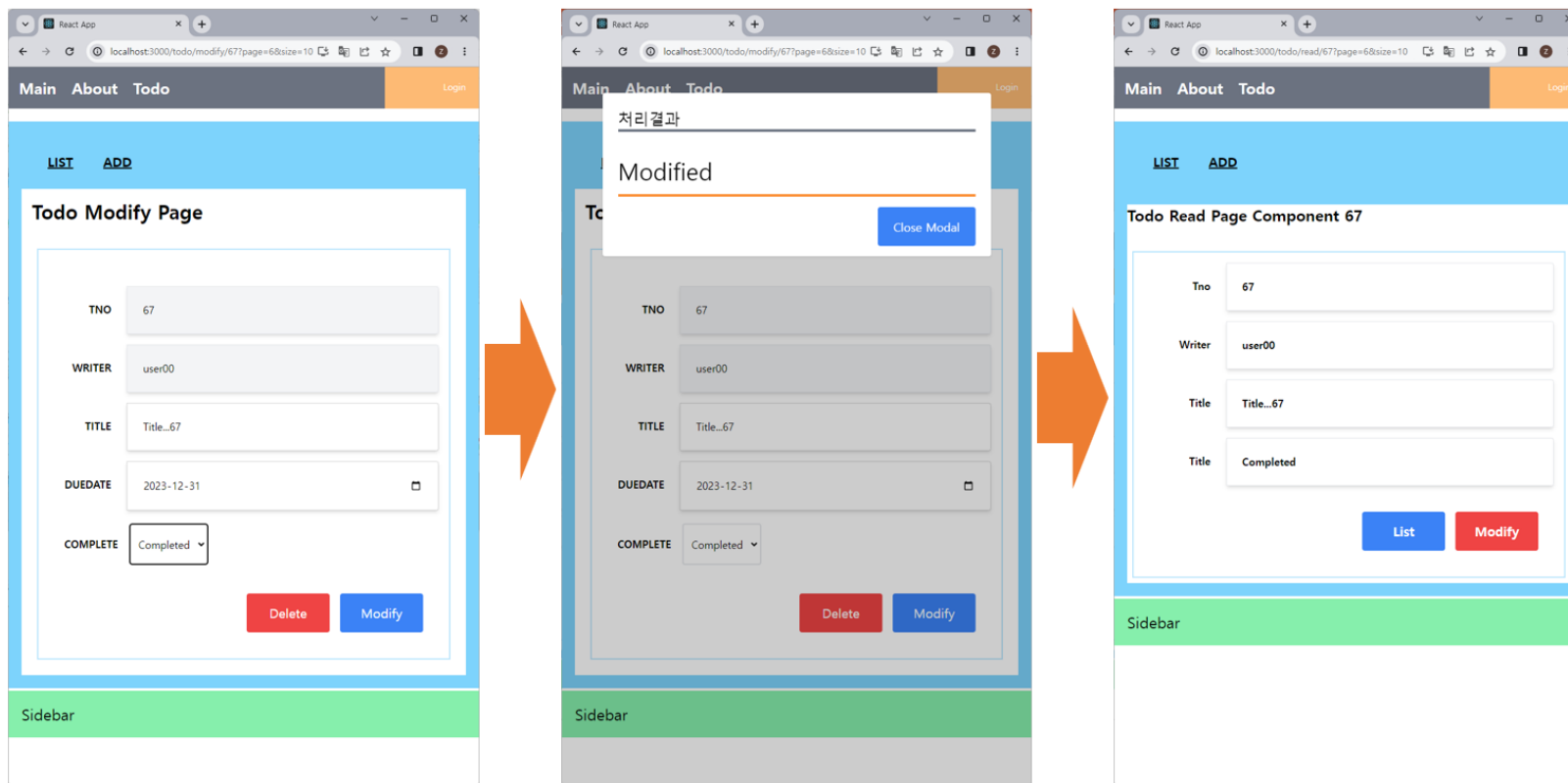
69	Title...69	2023-12-31
68	Title...68	2023-12-31
67	Title...67	2023-12-31

1 2 3 4 5 6 7 8 9 10 Next

</todo/read/67?page=6&size=10>

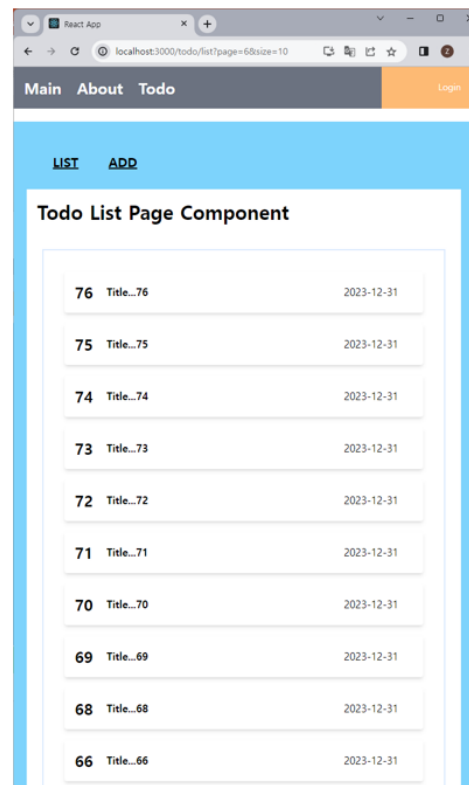
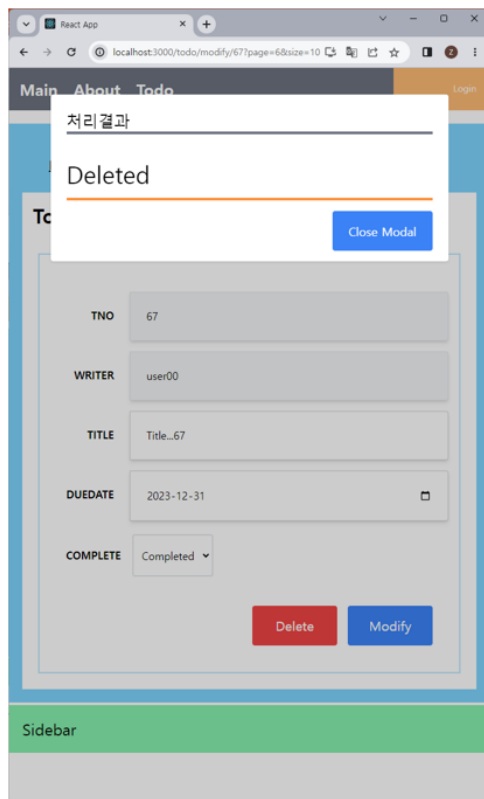
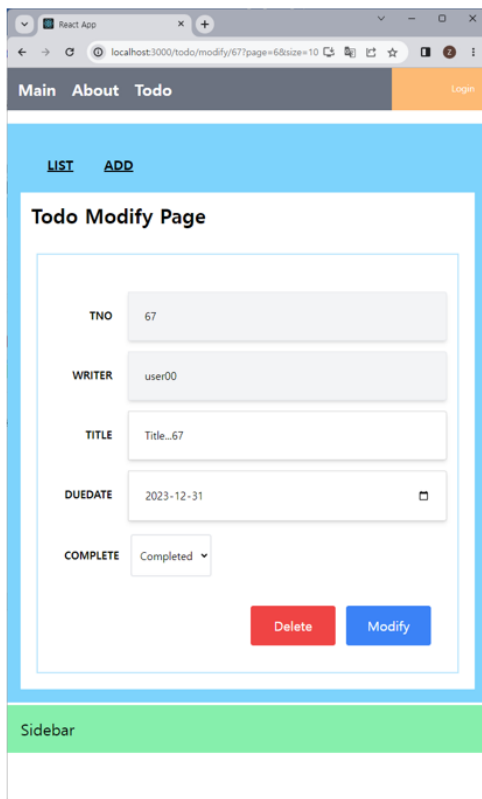
개발 목표의 이해

→ 수정/삭제



개발 목표의 이해

→ 수정/삭제



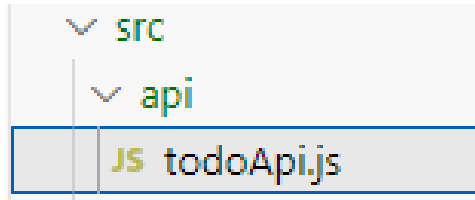
Ajax 통신 처리

→ Axios 라이브러리를 추가

```
npm install axios
```

Ajax 통신 처리

→ 리액트 프로젝트 내에 api 폴더를 추가하고 todoApi.js 파일을 추가



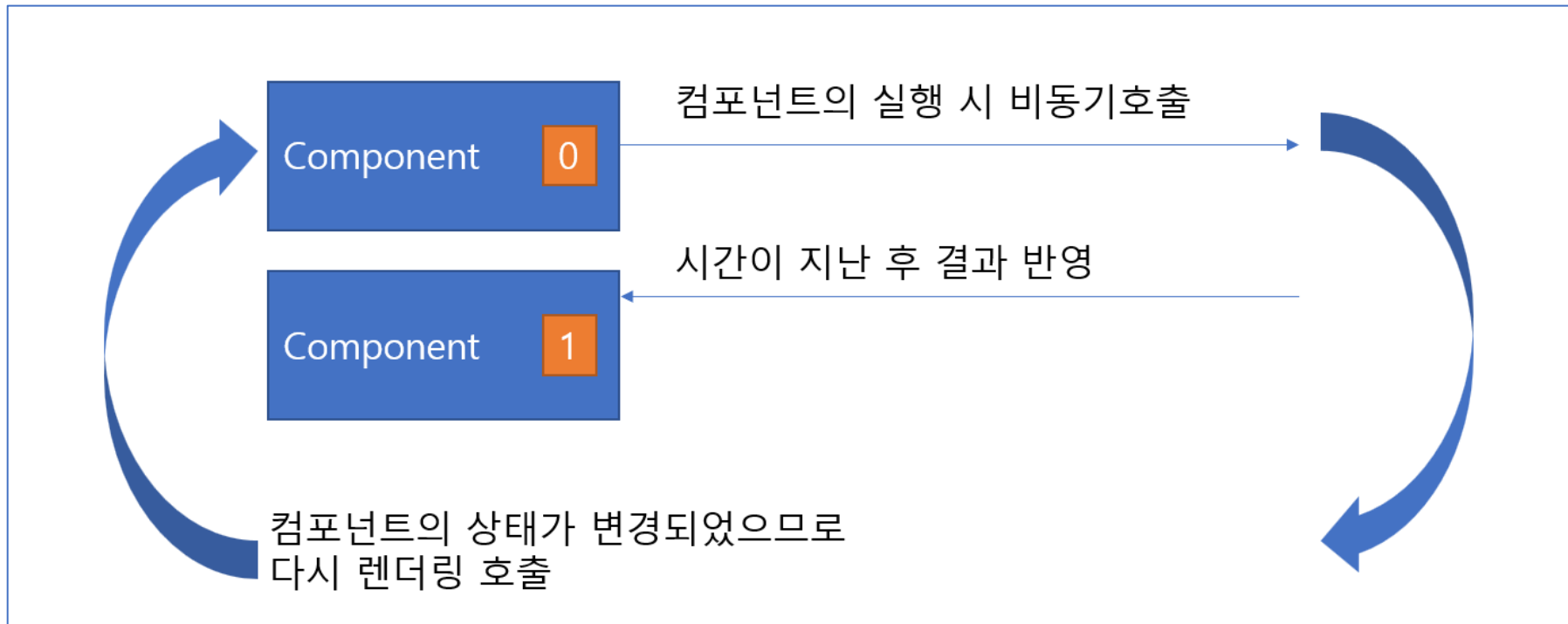
```
import axios from "axios"

export const API_SERVER_HOST = 'http://localhost:8080'
const prefix = `${API_SERVER_HOST}/api/todo`
export const getOne = async (tno) => {
  const res = await axios.get(`${prefix}/${tno}` )
  return res.data
}

export const getList = async ( pageParam ) => {
  const {page,size} = pageParam
  const res = await axios.get(`${prefix}/list`, {params: {page:page,size:size }})
  return res.data
}
```

useEffect()

→ 컴포넌트 상태 변경 시 렌더링과 함께 Axios 함수를 재호출하여 무한한 반복 구조 발생!!

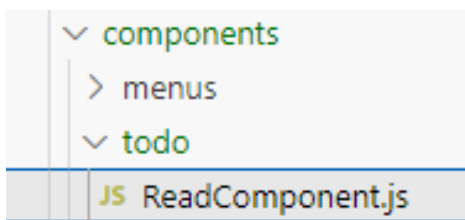


useEffect()

→ useEffect()를 활용하여 컴포넌트 내에서 특정 조건을 충족할 때 동작하는 방법 제공

- ✓ 컴포넌트 실행 시 단 한 번만 실행되는 비동기 처리
- ✓ 컴포넌트의 여러 상태 중 특정 상태 변경 시 비동기 처리

조회를 위한 컴포넌트



```
import { useEffect, useState } from "react"
import { getOne } from "../../api/todoApi"

const initState = {tno:0, title:'', writer:'', dueDate: null, complete: false }

const ReadComponent = ({tno}) => {

  const [todo, setTodo] = useState(initState) //아직 todo는 사용하지 않음

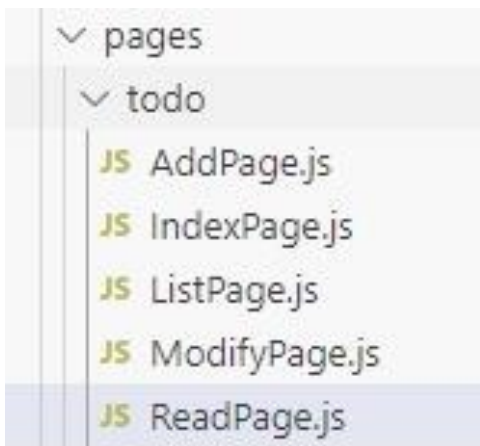
  useEffect(() => {
    getOne(tno).then(data => {
      console.log(data)
      setTodo(data)
    })
  }, [tno])

  return (
    <div></div>
  )
}

export default ReadComponent
```

조회를 위한 컴포넌트

→ ReadPage 컴포넌트에 ReadComponent를 import



```
import { useCallback } from "react";
import { createSearchParams, useNavigate, useParams, useSearchParams }
  from "react-router-dom";
import ReadComponent from "../../components/todo/ReadComponent";

const ReadPage = () => {
  ...생략
  const moveToList = useCallback(() => {
    navigate({pathname: `/todo/list`, search: queryStr})
  }, [page, size])

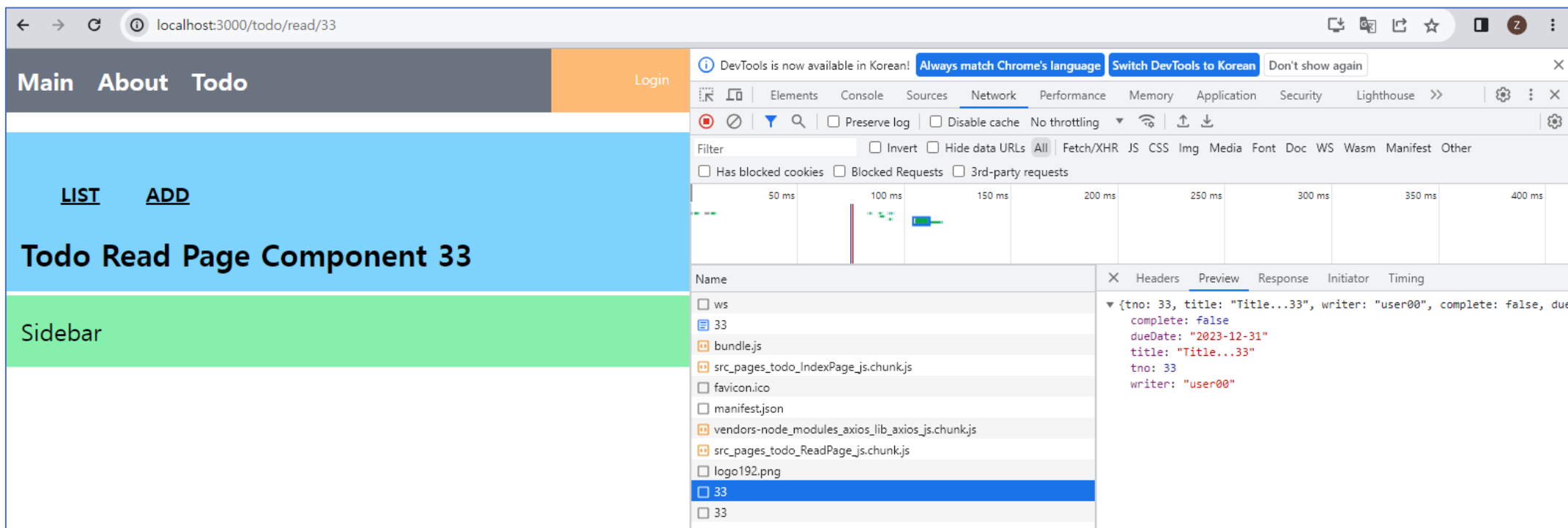
  return (
    <div className="font-extrabold w-full bg-white mt-6">
      <div className="text-2xl "> Todo Read Page Component {tno} </div>
      <ReadComponent tno={tno}></ReadComponent>
    </div>
  );
}

export default ReadPage;
```

조회를 위한 컴포넌트

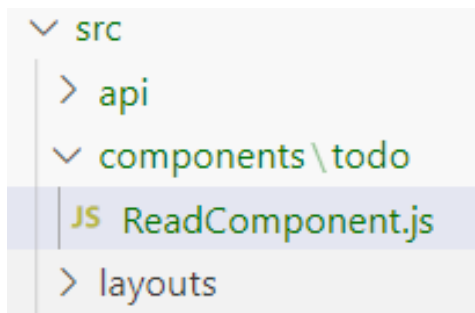
비동기 통신을 이용하는 경우 <StrictMode> 설정으로 인해 비동기 호출은 두 번 호출되는 경우가 발생
→ src/index.js 파일을 strict 모드를 사용하지 않도록 수정

→ ReadPage 컴포넌트에 ReadComponent를 import



조회를 위한 컴포넌트

→ ReadComponent 내부에 공통 구성을 가진 JSX를 반환하는 함수로 출력 구현



```
import { useEffect, useState } from "react"
import { getOne } from "../../api/todoApi"

const initState = { tno:0, title:'', writer: '', dueDate: null, complete: false }

const ReadComponent = ({tno}) => {

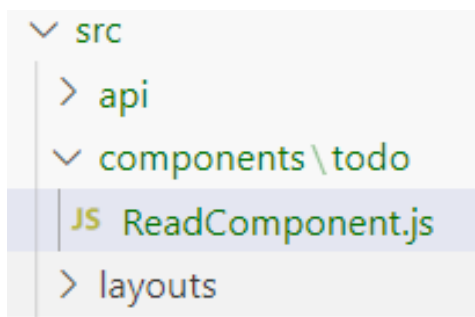
  const [todo, setTodo] = useState(initState) //아직 todo는 사용하지 않음

  useEffect(() => {
    getOne(tno).then(data => {
      console.log(data)
      setTodo(data)
    })
  }, [tno])

  // 다음페이지 계속
```

조회를 위한 컴포넌트

→ ReadComponent 내부에 공통 구성을 가진 JSX를 반환하는 함수로 출력 구현



```
return (
  <div className = "border-2 border-sky-200 mt-10 m-2 p-4">
    {makeDiv('Tno', todo.tno)}
    {makeDiv('Writer', todo.writer)}
    {makeDiv('Title', todo.title)}
    {makeDiv('Title', todo.complete ? 'Completed' : 'Not Yet')}
  </div>
)
}
const makeDiv = (title,value) =>
<div className="flex justify-center">
  <div className="relative mb-4 flex w-full flex-wrap items-stretch">
    <div className="w-1/5 p-6 text-right font-bold">{title}</div>
    <div className="w-4/5 p-6 rounded-r border border-solid shadow-md">
      {value}
    </div>
  </div>
</div>
export default ReadComponent
```

조회 화면의 버튼 처리

→ 페이지 컴포넌트

React-Router를 이용해 내용 컴포넌트에 필요한 기능과 속성 설정.

ReadPage에 라우팅 함수 정의 및 ReadComponent에 함수 속성 전달.

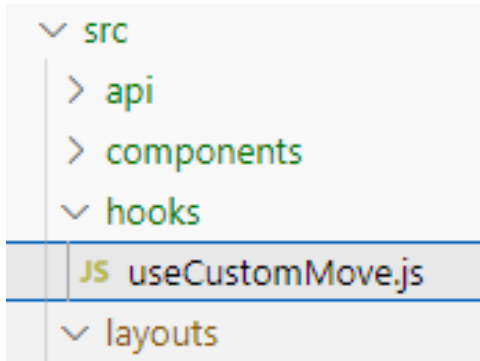
→ 내용 컴포넌트

실제 화면 구성, 버튼 등 내용 처리.

ReadComponent에 전체 구성 요소 처리.

목록 페이지로 이동

→ src 폴더에 hooks 폴더를 추가하고 useCustomMove.js 파일을 추가



```
import { createSearchParams, useNavigate, useSearchParams } from "react-router-dom"
const getNum = (param, defaultValue) => {
  if(!param){
    return defaultValue
  }
  return parseInt(param)
}

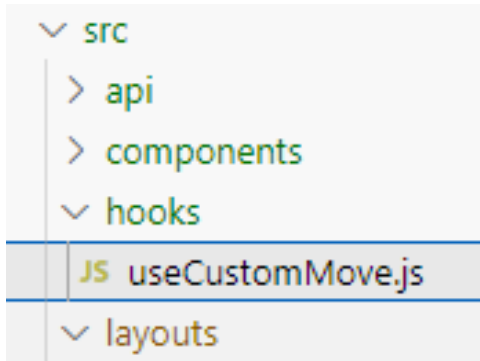
const useCustomMove = () => {
  const navigate = useNavigate()
  const [queryParams] = useSearchParams()

  const page = getNum(queryParams.get('page'), 1)
  const size = getNum(queryParams.get('size'), 10)

  const queryDefault = createSearchParams({page, size}).toString() //새로 추가
```

목록 페이지로 이동

→ src 폴더에 hooks 폴더를 추가하고 useCustomMove.js 파일을 추가



```
const moveToList = (pageParam) => {
  let queryStr = ""
  if(pageParam){
    const pageNum = getNum(pageParam.page, 1)
    const sizeNum = getNum(pageParam.size, 10)

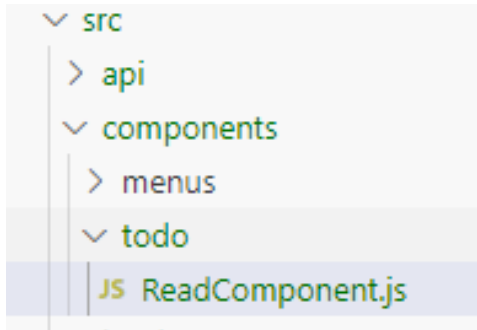
    queryStr = createSearchParams({page:pageNum, size: sizeNum}).toString()
  }else {
    queryStr = queryDefault
  }
}

return {moveToList, moveToModify, page, size}
}

export default useCustomMove
```

목록 페이지로 이동

→ ReadComponent에 useCustomMove()에 moveToList()를 이용하고 버튼을 추가



```
import { useEffect, useState } from "react"
import { getOne } from "../../api/todoApi"
import useCustomMove from "../../hooks/useCustomMove" //추가
...생략

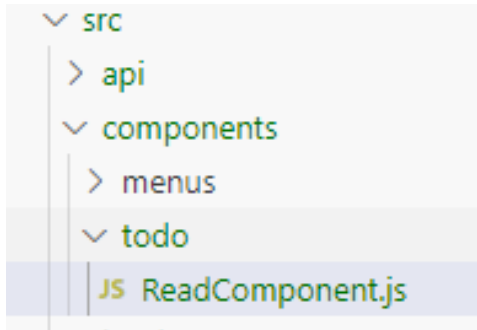
const ReadComponent = ({ tno }) => {
  const [todo, setTodo] = useState(initState) //아직 todo는 사용하지 않음
  const { moveToList } = useCustomMove()

  useEffect(() => { ...생략 }, [tno])

  return (
    <div className = "border-2 border-sky-200 mt-10 m-2 p-4 ">
      {makeDiv('Tno', todo.tno)}
      {makeDiv('Writer', todo.writer)}
      {makeDiv('Title', todo.title)}
      {makeDiv('Title', todo.complete ? 'Completed' : 'Not Yet ')}
    </div>
  )
}
```

목록 페이지로 이동

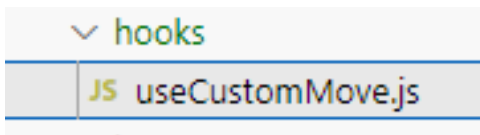
→ ReadComponent에 useCustomMove()에 moveToList()를 이용하고 버튼을 추가



```
    { /* buttons.....start */  
    <div className="flex justify-end p-4">  
  
      <button type="button"  
        className="rounded p-4 m-2 text-xl w-32 text-white bg-blue-500"  
        onClick={() => moveToList()}  
      >  
        List  
      </button>  
    </div>  
  
  </div>  
)  
}  
...생략  
  
export default ReadComponent
```

수정/삭제 페이지로 이동

→ useCustomMove에 수정/삭제로 이동할 수 있는 기능을 추가



```
import { useCallback } from "react"
import { createSearchParams, useNavigate, useSearchParams } from "react-router-dom"

const useCustomMove = () => {

  const navigate = useNavigate()
  const [queryParams] = useSearchParams()
  const page = queryParams.get("page") ? parseInt(queryParams.get("page")) : 1
  const size = queryParams.get("size") ? parseInt(queryParams.get("size")) : 10
  const queryDefault = createSearchParams({page, size}).toString()

  const moveToList = useCallback((pageParam) => { ...생략 }, [page, size])

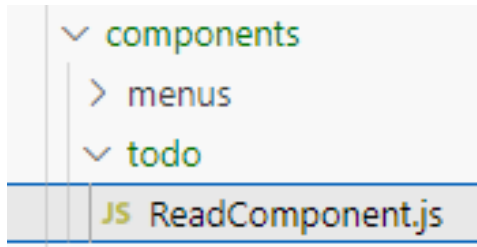
  const moveToModify = useCallback((num) => {
    console.log(queryDefault)
    navigate({ pathname: `../modify/${num}`, search: queryDefault })
  }, [page, size])

  return {moveToList, moveToModify, page, size}
}

export default useCustomMove
```


수정/삭제 페이지로 이동

→ useCustomMove와 ReadComponent를 수정해서 수정/삭제 버튼을 추가



```
import { useEffect, useState } from "react"
import { getOne } from "../../api/todoApi"
import useCustomMove from "../../hooks/useCustomMove"

const initState = { ...생략 }

const ReadComponent = ({ tno }) => {

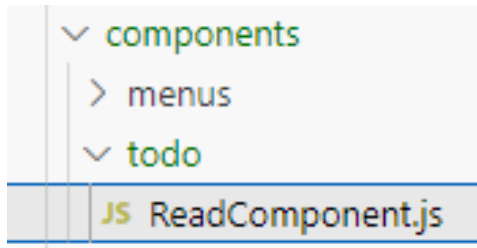
  const [todo, setTodo] = useState(initState) //아직 todo는 사용하지 않음

  //이동과 관련 기능은 모두 useCustomMove()로
  const { moveToList, moveToModify } = useCustomMove()

  useEffect(() => { ...생략 }, [tno])
```

수정/삭제 페이지로 이동

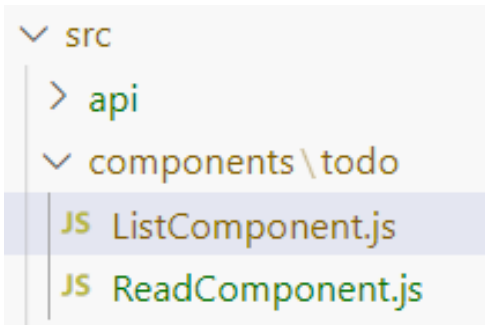
→ useCustomMove와 ReadComponent를 수정해서 수정/삭제 버튼을 추가



```
return (  
  <div className = "border-2 border-sky-200 mt-10 m-2 p-4 ">  
    ...생략  
    { /* buttons.....start */ }  
    <div className="flex justify-end p-4">  
      <button type="button"  
        className="rounded p-4 m-2 text-xl w-32 text-white bg-blue-500"  
        onClick={() => moveToList()} > List </button>  
      <button type="button"  
        className="rounded p-4 m-2 text-xl w-32 text-white bg-red-500"  
        onClick={() => moveToModify(tno)} > Modify </button>  
    </div>  
  </div>  
)  
}  
  
const makeDiv = ..생략  
  
export default ReadComponent
```

목록 데이터 처리

→ components/todo 폴더에
ListComponent 생성



```
import { useEffect, useState } from "react";
import { getList } from "../../api/todoApi";
import useCustomMove from "../../hooks/useCustomMove";

const initState = {
  dtoList: [], pageNumList: [], pageRequestDT0: null, prev: false, next: false,
  totoalCount: 0, prevPage: 0, nextPage: 0, totalPage: 0, current: 0 }

const ListComponent = () => {
  const {page, size} = useCustomMove()
  //serverData는 나중에 사용
  const [serverData, setServerData] = useState(initState)

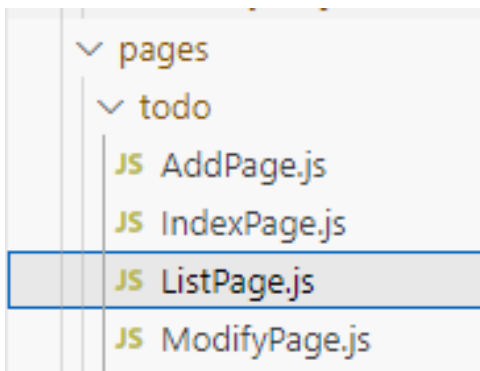
  useEffect(() => {
    getList({page, size}).then(data => {
      console.log(data)
      setServerData(data)
    })
  }, [page, size])

  return (
    <div> Todo List Component </div>
  );
}

export default ListComponent;
```

목록 데이터 처리

→ ListPage에서 ListComponent를 import 하고 page, size 속성을 전달



```
import ListComponent from "../../components/todo/ListComponent";

const ListPage = () => {

  return (
    <div className="p-4 w-full bg-white">
      <div className="text-3xl font-extrabold">
        Todo List Page Component
      </div>

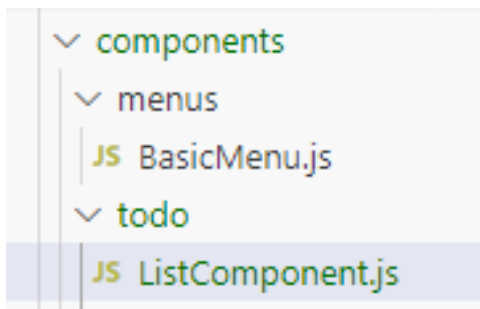
      <ListComponent/>

    </div>
  );
}

export default ListPage;
```

목록 데이터 처리

→ 서버에서 가져온 데이터들을 ListComponent에서 출력



```
return (  
  <div className="border-2 border-blue-100 mt-10 mr-2 ml-2">  
    <div className="flex flex-wrap mx-auto justify-center p-6">  
  
      {serverData.dtoList.map(todo =>  
        <div key= {todo.tno} className="w-full min-w-[400px] p-2 m-2 rounded shadow-md">  
          <div className="flex ">  
            <div className="font-extrabold text-2xl p-2 w-1/12"> {todo.tno} </div>  
            <div className="text-1xl m-1 p-2 w-8/12 font-extrabold">{todo.title}</div>  
            <div className="text-1xl m-1 p-2 w-2/10 font-medium"> {todo.dueDate} </div>  
          </div>  
        </div>  
      )}  
  
    </div>  
  </div>  
);
```

페이징 처리

→ 서버에서 전달 받은 데이터

```
▼ Object ⓘ  
  current: 1  
  ▶ dtoList: (10) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}]  
  next: false  
  nextPage: 0  
  ▶ pageNumList: (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
  ▶ pageRequestDTO: {page: 1, size: 10}  
  prev: false  
  prevPage: 0  
  totalCount: 100  
  totalPages: 10  
  ▶ [[Prototype]]: Object
```

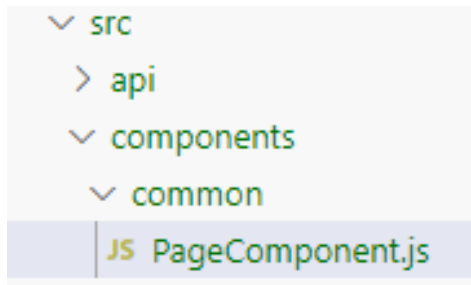
ListComponent.js:30

페이징 처리

→ components >

common >

PageComponent.js



```
const PageComponent = ({serverData, movePage}) => {

  return (
    <div className="m-6 flex justify-center">
      {serverData.prev ?
        <div className="m-2 p-2 w-16 text-center font-bold text-blue-400 "
          onClick={() => movePage({page:serverData.prevPage} )}>
            Prev </div> : <></>

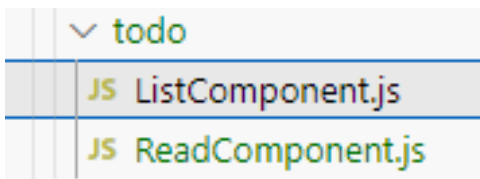
        {serverData.pageNumList.map(pageNum =>
          <div key={pageNum}
            className={`m-2 p-2 w-12 text-center rounded shadow-md text-white
              ${serverData.current === pageNum? 'bg-gray-500':'bg-blue-400'}}`
            onClick={() => movePage( {page:pageNum})}>
              {pageNum}
            </div>
          )}

        {serverData.next ?
          <div className="m-2 p-2 w-16 text-center font-bold text-blue-400"
            onClick={() => movePage( {page:serverData.nextPage})}>
            Next
          </div> : <></>
        </div>
      </div>
    );
  }

  export default PageComponent;
```

페이징 처리

→ PageComponent import



```
import { useEffect, useState } from "react";
import { getList } from "../api/todoApi";
import useCustomMove from "../hooks/useCustomMove";
import PageComponent from "../common/PageComponent";

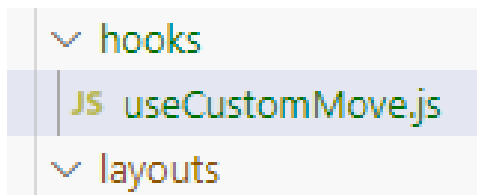
const initState = { ... }

const ListComponent = () => {
  ...
  return (
    <div className="border-2 border-blue-100 mt-10 mr-2 ml-2">
      <div className="flex flex-wrap mx-auto justify-center p-6">
        {serverData.dtoList.map(todo => ... )}
      </div>
      <PageComponent serverData={serverData} movePage={moveToList}></PageComponent>
    </div>
  );
}

export default ListComponent;
```


동일 페이지 클릭 시 문제

→ 컴포넌트 내부에 매번 변하는 상태(state)값을 이용



```
import { useState } from "react"
import { createSearchParams, useNavigate, useSearchParams } from "react-router-dom"

const getNum = (param, defaultValue) => { ... 생략 ... }

const useCustomMove = () => {

  const navigate = useNavigate()

  const [refresh, setRefresh] = useState(false) //추가

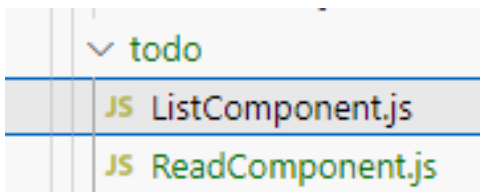
  const [queryParams] = useSearchParams()

  const page = getNum(queryParams.get('page'), 1)
  const size = getNum(queryParams.get('size'), 10)

  const queryDefault = createSearchParams({page, size}).toString()
```

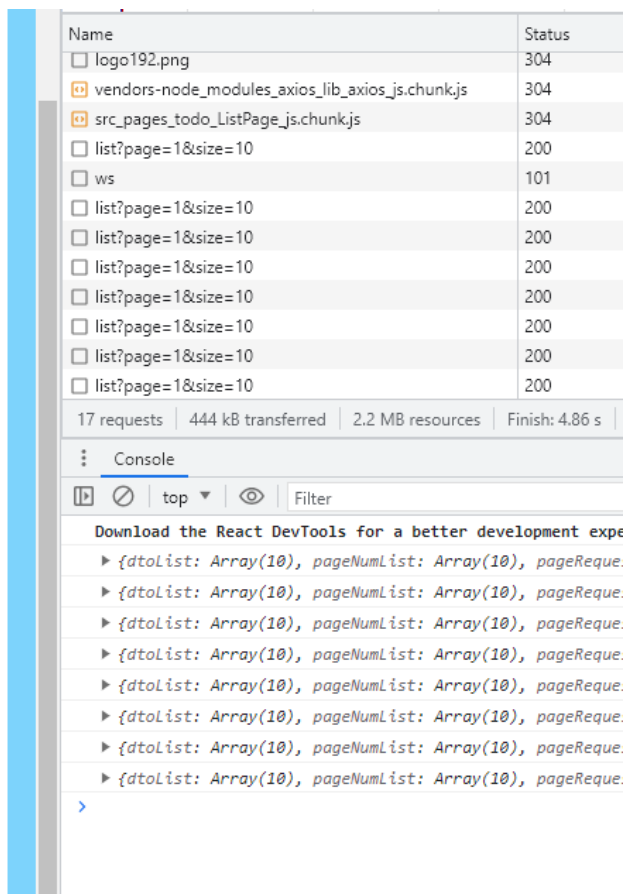
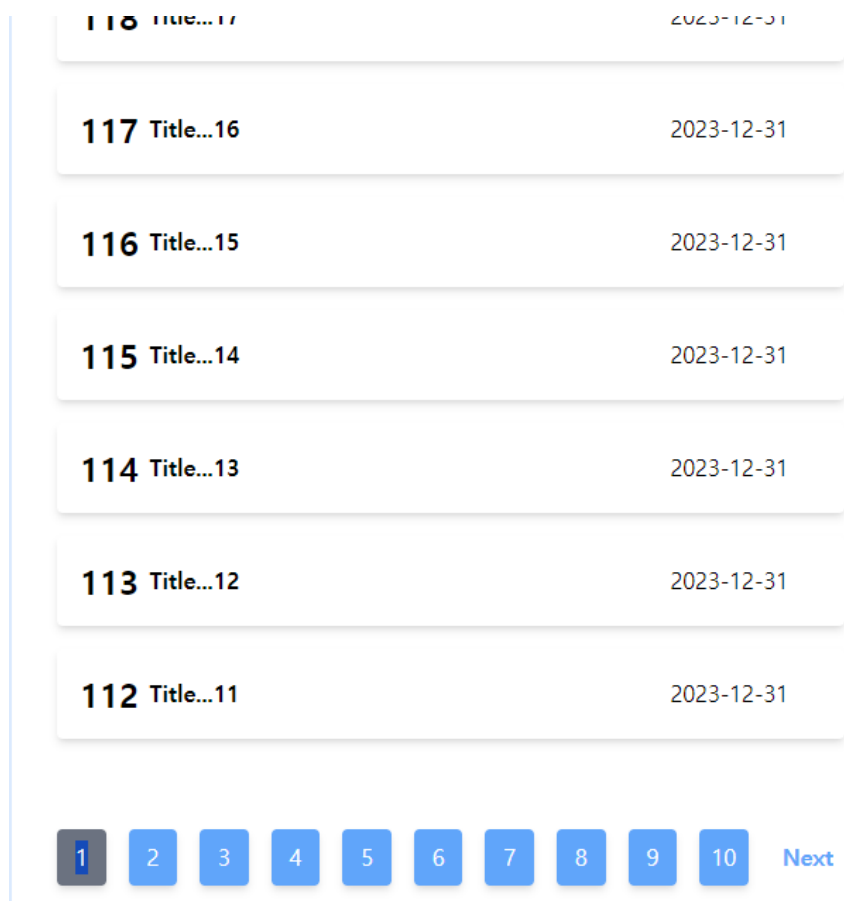
동일 페이지 클릭 시 문제

→ refresh값은 ListComponent의 useEffect()에서 사용하도록 설정



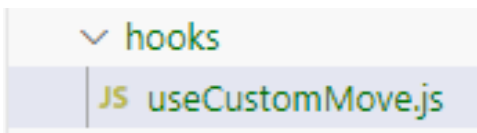
```
const ListComponent = () => {  
  const {page, size, refresh, moveToList} = useCustomMove();//refresh 추가  
  
  //serverData는 나중에 사용  
  const [serverData, setServerData] = useState(initState)  
  
  useEffect(() => {  
    getList({page, size}).then(data => {  
      console.log(data)  
      setServerData(data)  
    })  
  }, [page, size, refresh])  
  
  return (  
    ...
```

동일 페이지 클릭 시 문제



조회 페이지 이동

→ useCustomMove()를 이용해서 moveToRead() 함수 추가



```
import { useState } from "react"
import { createSearchParams, useNavigate, useSearchParams } from "react-router-dom"

const useCustomMove = () => {

  ...생략

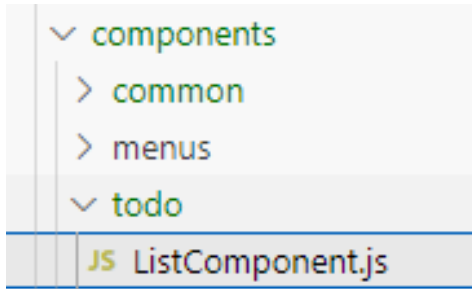
  const moveToRead = (num) => {
    console.log(queryDefault)
    navigate({
      pathname: `../read/${num}`,
      search: queryDefault
    })
  }

  return {moveToList, moveToModify, moveToRead, page, size, refresh} //moveToRead 추가
}

export default useCustomMove
```

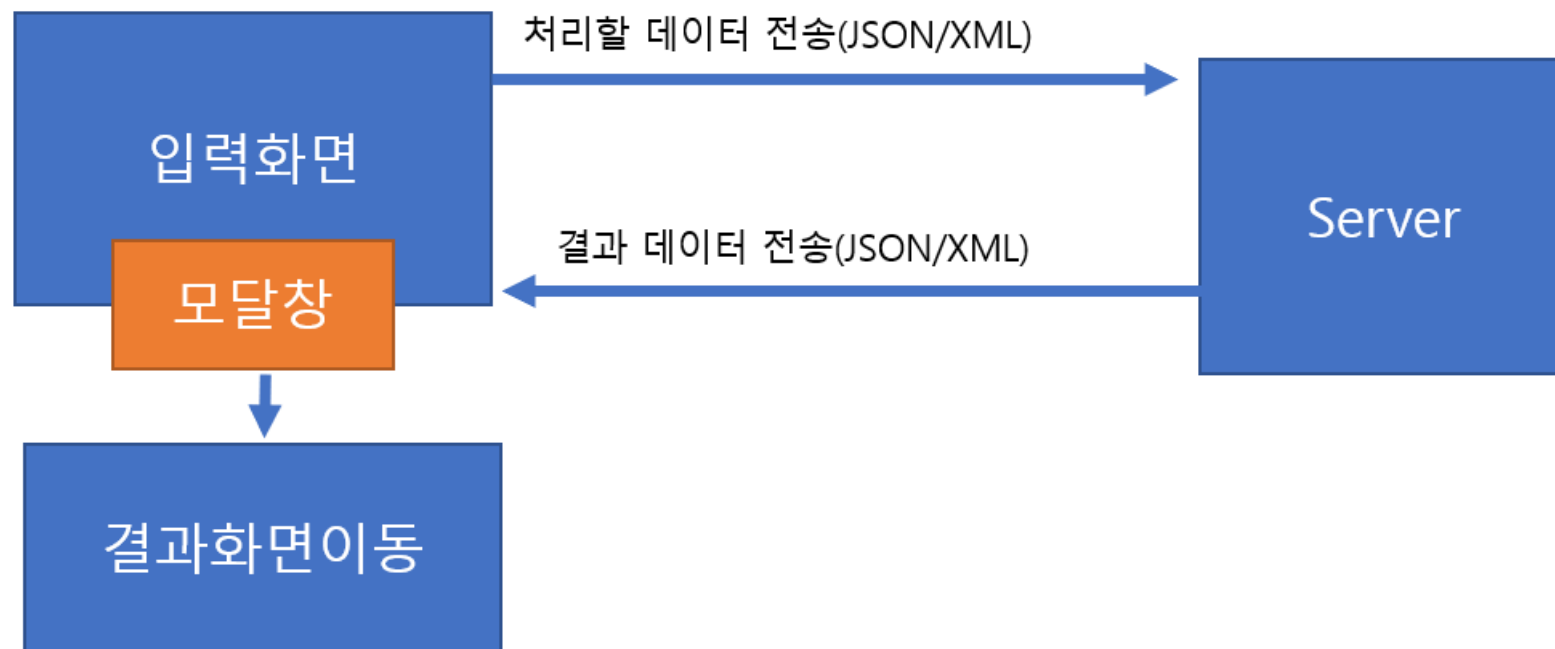
조회 페이지 이동

→ useCustomMove()를 이용해서 moveToRead() 함수 추가



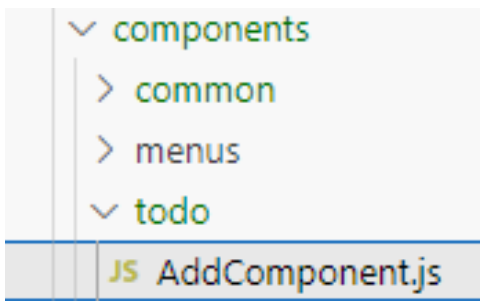
```
const ListComponent = () => {  
  //moveToRead  
  const {page, size, refresh, moveToList, moveToRead} = useCustomMove()  
  
  ...생략  
  
  return (  
    <div className="border-2 border-blue-100 mt-10 mr-2 ml-2">  
  
      <div className="flex flex-wrap mx-auto justify-center p-6">  
  
        {serverData.dtoList.map(todo =>  
  
          <div key= {todo.tno}  
            className="w-full min-w-[400px] p-2 m-2 rounded shadow-md"  
            onClick={() => moveToRead(todo.tno)} //이벤트 처리 추가  
          >  
  
            ...생략
```

등록 컴포넌트와 모달 처리



등록 컴포넌트와 모달 처리

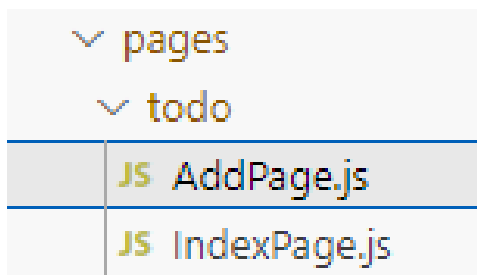
→ components>todo>AddComponent.js



```
const AddComponent = () => {  
  return (  
    <>Add Component</>  
  );  
}  
  
export default AddComponent;
```

등록 컴포넌트와 모달 처리

→ pages > todo > AddPageAddPage.js에 AddComponent import



```
import AddComponent from "../../components/todo/AddComponent";

const AddPage = () => {

  return (
    <div className="p-4 w-full bg-white">
      <div className="text-3xl font-extrabold">
        Todo Add Page
      </div>

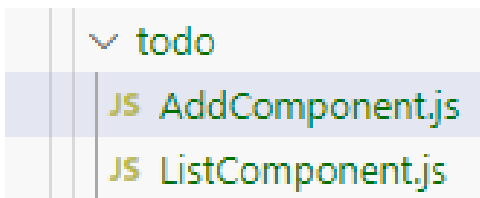
      <AddComponent/>

    </div>
  );
}

export default AddPage;
```


등록 컴포넌트와 모달 처리

→ AddComponent 기능 구현



```
import { useState } from "react";

const initState = {
  title: ' ', writer: ' ', dueDate: ' ' }

const AddComponent = () => {

  const [todo, setTodo] = useState({...initState})
  const handleChangeTodo = (e) => {
    todo[e.target.name] = e.target.value
    setTodo({...todo})
  }

  const handleClickAdd = () => { console.log(todo) }

  return ( // 버튼 추가 );
}

export default AddComponent;
```

등록 컴포넌트와 모달 처리

→ AddComponent.js

```
<div className = "border-2 border-sky-200 mt-10 m-2 p-4">
  <div className="flex justify-center">
    <div className="relative mb-4 flex w-full flex-wrap items-stretch">
      <div className="w-1/5 p-6 text-right font-bold">TITLE</div>
      <input className="w-4/5 p-6 rounded-r border border-solid border-neutral-500 shadow-md"
        name="title" type={'text'} value={todo.title} onChange={handleChangeTodo}></input>
    </div>
  </div>
  <div className="flex justify-center">
    <div className="relative mb-4 flex w-full flex-wrap items-stretch">
      <div className="w-1/5 p-6 text-right font-bold">WRITER</div>
      <input className="w-4/5 p-6 rounded-r border border-solid border-neutral-500 shadow-md"
        name="writer" type={'text'} value={todo.writer} onChange={handleChangeTodo}></input>
    </div>
  </div>
</div>
```

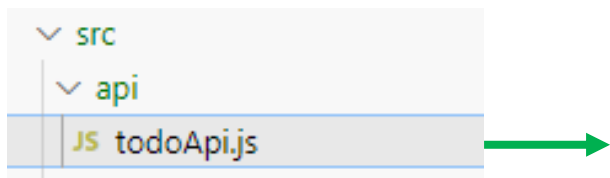
등록 컴포넌트와 모달 처리

→ AddComponent.js

```
<div className="flex justify-center">
  <div className="relative mb-4 flex w-full flex-wrap items-stretch">
    <div className="w-1/5 p-6 text-right font-bold">DUEDATE</div>
    <input className="w-4/5 p-6 rounded-r border border-solid border-neutral-500 shadow-md"
      name="dueDate" type={'date'} value={todo.dueDate} onChange={handleChangeTodo}></input>
  </div>
</div>
<div className="flex justify-end">
  <div className="relative mb-4 flex p-4 flex-wrap items-stretch">
    <button type="button"
      onClick={handleClickAdd}
      className="rounded p-4 w-36 bg-blue-500 text-xl text-white" > ADD </button>
  </div>
</div>
</div>
```

서버 호출 결과 확인

→ todoApi.js : POST 방식으로 동작하는 함수를 추가



```
import axios from "axios"

export const API_SERVER_HOST = 'http://localhost:8080'

const prefix = `${API_SERVER_HOST}/api/todo`

...생략

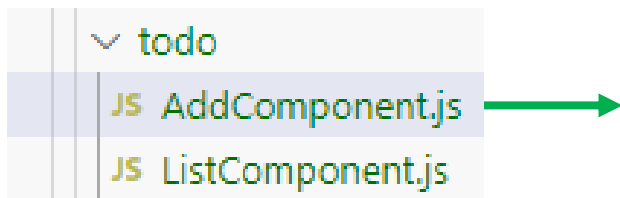
export const postAdd = async (todoObj) => {

  const res = await axios.post(`${prefix}/`, todoObj)

  return res.data
}
```

서버 호출 결과 확인

→ AddComponent



```
import { useState } from "react";
import { postAdd } from "../../api/todoApi";

const initState = { title:'', writer: '', dueDate: '' }

const AddComponent = () => {
  const [todo, setTodo] = useState({...initState})

  const handleChangeTodo = (e) => {
    todo[e.target.name] = e.target.value
    setTodo({...todo})
  }

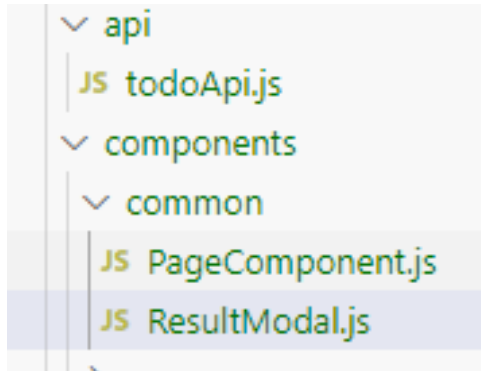
  const handleClickAdd = () => {
    postAdd(todo)
    .then(result => {
      console.log(result)
      setTodo({...initState}) //초기화
    }).catch(e => { console.error(e) })
  }

  return (
    ...생략
  );
}

export default AddComponent;
```

모달 컴포넌트의 제작

→ common > ResultModal.js 추가

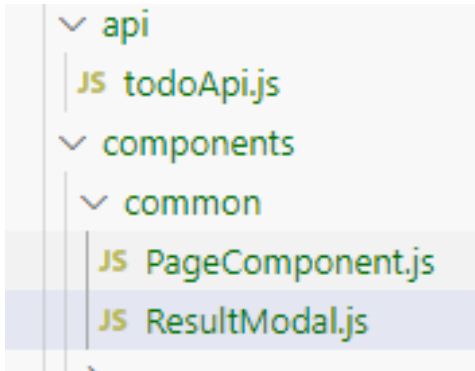


```
const ResultModal = ( {title,content, callbackFn} ) => {
  return (
    <div className={`fixed top-0 left-0 z-[1055] flex h-full w-full justify-center bg-black
bg-opacity-20`} onClick={() => { if(callbackFn) { callbackFn() } }}>

      <div className="absolute bg-white shadow dark:bg-gray-700 opacity-100 w-1/4 rounded mt-
10 mb-10 px-6 min-w-[600px]">
        <div className="justify-center bg-warning-400 mt-6 mb-6 text-2xl border-b-4 border-
gray-500"> {title} </div>
        <div className="text-4xl border-orange-400 border-b-4 pt-4 pb-4"> {content} </div>
        <div className="justify-end flex ">
          <button className="rounded bg-blue-500 mt-4 mb-4 px-6 pt-4 pb-4 text-lg text-white"
onClick={() => { if(callbackFn) { callbackFn() } }}>Close Modal</button>
        </div>
      </div>
    </div>
  );
}
export default ResultModal;
```

모달 컴포넌트의 제작

→ AddComponent에 ResultModal을 import 하고 ResultModal을 사용하게 되는 상태 처리 추가



```
import { useState } from "react";
import { postAdd } from "../../api/todoApi";
import ResultModal from "../../common/ResultModal";

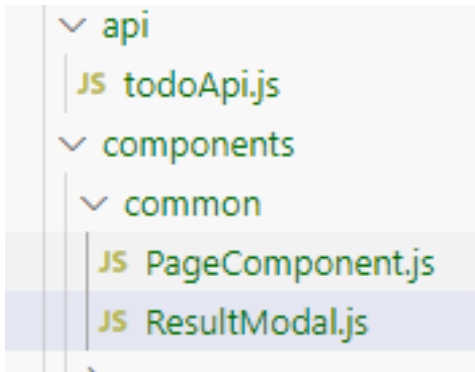
const initState = { title: '', writer: '', dueDate: '' }

const AddComponent = () => {
  const [todo, setTodo] = useState({...initState})

  //결과데이터가 있는 경우에는 ResultModal을 보여준다.
  const [result, setResult] = useState(null) //결과 상태
  ...생략
  const handleClickAdd = () => {
    postAdd(todo)
    .then(result => {
      setResult(result.TNO) //결과 데이터 변경
      setTodo({...initState})
    }).catch(e => { console.error(e) })
  }
}
```

모달 컴포넌트의 제작

→ AddComponent에 ResultModal을 import 하고 ResultModal을 사용하게 되는 상태 처리 추가



```
const closeModal = () => {
  setResult(null)
}

return (
  <div className = "border-2 border-sky-200 mt-10 m-2 p-4">

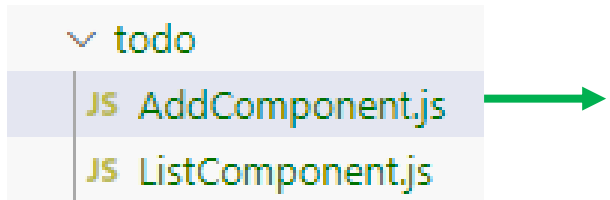
    {/* 모달 처리 */}
    {result ? <ResultModal title={'Add Result'} content={`New ${result} Added`}
    callbackFn={closeModal}/>: <></>}

    <div className="flex justify-center">
      ...생략
    </div>
  </div>
);
}

export default AddComponent;
```


페이지 이동

→useCustomMove() 수정



```
import { useState } from "react";
import { postAdd } from "../../api/todoApi";
import ResultModal from "../../common/ResultModal";
import useCustomMove from "../../hooks/useCustomMove";

...생략
const AddComponent = () => {
  const [todo, setTodo] = useState({...initState})
  const [result, setResult] = useState(null)
  const {moveToList} = useCustomMove() //useCustomMove 활용
  ...생략
  const closeModal = () => {
    setResult(null)
    moveToList() //moveToList( )호출
  }

  return (
    <div className = "border-2 border-sky-200 mt-10 m-2 p-4">
      {result ? <ResultModal title={'Add Result'} content={`New ${result} Added`}
        callbackFn={closeModal}/>: <>/>}
      ...생략
    </div>
  );
}
export default AddComponent;
```

수정/삭제 처리

→ 삭제>Delete 버튼)

: 삭제 결과를 모달창으로 보여주고 '/todo/list'로 이동

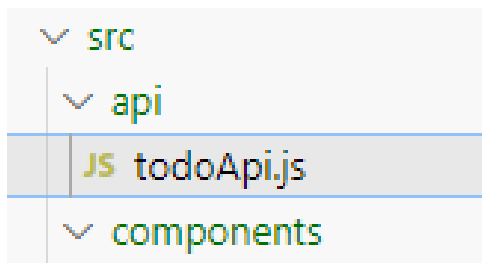
→ 수정>Modify 버튼)

: 수정 결과를 모달창으로 보여주고 '/todo/read/번호'로 이동

→ 결과의 출력은 공통적으로 모달 창을 이용

수정/삭제 호출 기능 작성

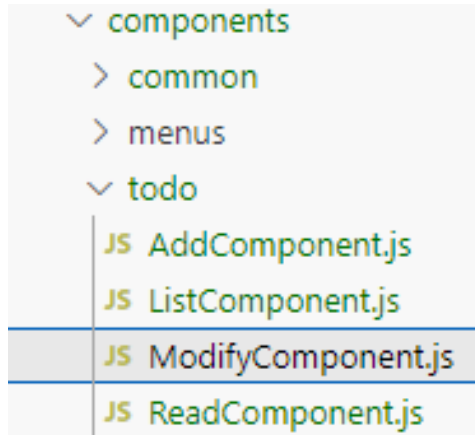
→ todoApi.js에 수정/삭제 함수를 추가



```
export const deleteOne = async (tno) => {  
  const res = await axios.delete(`${prefix}/${tno}`)  
  return res.data  
}  
  
export const putOne = async (todo) => {  
  const res = await axios.put(`${prefix}/${todo.tno}`, todo)  
  return res.data  
}
```

수정/삭제를 위한 컴포넌트

→ ModifyComponent를 추가



```
import { useEffect, useState } from "react";

const initState = { tno:0, title:'', writer: '', dueDate: null, complete: false }

const ModifyComponent = ({tno}) => {
  const [todo, setTodo] = useState({...initState})

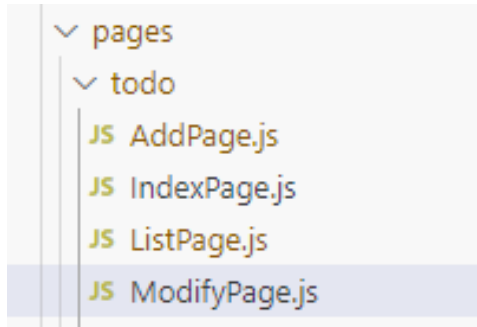
  useEffect(() => { },[tno])

  return (
    <div className = "border-2 border-sky-200 mt-10 m-2 p-4">
      <div className="flex justify-end p-4">
        <button type="button"
          className="inline-block rounded p-4 m-2 text-xl w-32 text-white bg-red-500" >
          Delete
        </button>
        <button type="button"
          className="rounded p-4 m-2 text-xl w-32 text-white bg-blue-500" >
          Modify
        </button>
      </div>
    </div>
  );
}

export default ModifyComponent;
```

수정/삭제를 위한 컴포넌트

→ ModifyPage에 ModifyComponent를 추가



```
import { useParams } from "react-router-dom";
import ModifyComponent from "../../components/todo/ModifyComponent";

const ModifyPage = () => {

  const {tno} = useParams()

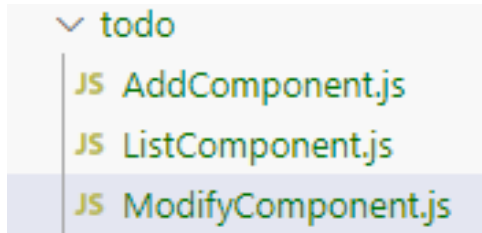
  return (
    <div className="p-4 w-full bg-white">
      <div className="text-3xl font-extrabold">
        Todo Modify Page
      </div>

      <ModifyComponent tno={tno}/>
    </div>
  );
}

export default ModifyPage;
```

서버 데이터 출력

→ ModifyComponent에 기능 구성 필요 : 데이터 출력/변경



```
import { useEffect, useState } from "react";
import { getOne } from "../../api/todoApi";

const initState = { tno:0, title:'', writer: '', dueDate: '', complete: false }

const ModifyComponent = ({tno, moveList, moveRead}) => {
  const [todo, setTodo] = useState({...initState})

  useEffect(() => { getOne(tno).then(data => setTodo(data)) },[tno])

  const handleChangeTodo = (e) => {
    todo[e.target.name] = e.target.value
    setTodo({...todo})
  }

  const handleChangeTodoComplete = (e) => {
    const value = e.target.value
    todo.complete = (value === 'Y')
    setTodo({...todo})
  }
}
```

서버 데이터 출력

→ ModifyComponent에 기능 구성 필요 : 데이터 출력/변경

```
return (  
  <div className = "border-2 border-sky-200 mt-10 m-2 p-4">  
  
    <div className="flex justify-center mt-10">  
      <div className="relative mb-4 flex w-full flex-wrap items-stretch">  
        <div className="w-1/5 p-6 text-right font-bold">TNO</div>  
        <div className="w-4/5 p-6 rounded-r border border-solid shadow-md bg-gray-100"> {todo.tno} </div>  
      </div>  
    </div>  
  
    <div className="flex justify-center">  
      <div className="relative mb-4 flex w-full flex-wrap items-stretch">  
        <div className="w-1/5 p-6 text-right font-bold">WRITER</div>  
        <div className="w-4/5 p-6 rounded-r border border-solid shadow-md bg-gray-100"> {todo.writer} </div>  
      </div>  
    </div>  
  </div>  
)
```

서버 데이터 출력

→ ModifyComponent에 기능 구성 필요 : 데이터 출력/변경

```
<div className="flex justify-center">
  <div className="relative mb-4 flex w-full flex-wrap items-stretch">
    <div className="w-1/5 p-6 text-right font-bold">TITLE</div>
    <input className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md"
      name="title" type={'text'} value={todo.title} onChange={handleChangeTodo} ></input>
  </div>
</div>

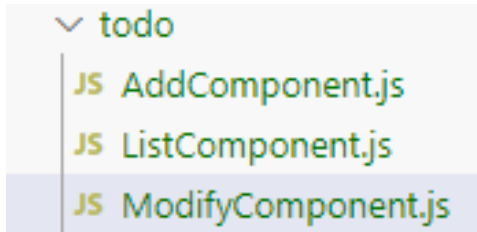
<div className="flex justify-center">
  <div className="relative mb-4 flex w-full flex-wrap items-stretch">
    <div className="w-1/5 p-6 text-right font-bold">DUEDATE</div>
    <input className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md"
      name="dueDate" type={'date'} value={todo.dueDate} onChange={handleChangeTodo} ></input>
  </div>
</div>
```


ModifyComponent의 출력

```
<div className="flex justify-center">
  <div className="relative mb-4 flex w-full flex-wrap items-stretch">
    <div className="w-1/5 p-6 text-right font-bold">COMPLETE</div>
    <select name="status" className="border-solid border-2 rounded m-1 p-2"
      onChange={handleChangeTodoComplete} value = {todo.complete? 'Y':'N'} >
      <option value='Y'>Completed</option>
      <option value='N'>Not Yet</option>
    </select>
  </div>
</div>
<div className="flex justify-end p-4">
  <button type="button" className="inline-block rounded p-4 m-2 text-xl w-32 text-white bg-red-500">
    Delete </button>
  <button type="button" className="rounded p-4 m-2 text-xl w-32 text-white bg-blue-500"> Modify </button>
</div>
</div>
);
}
export default ModifyComponent;
```

수정/삭제와 모달창

→ ModifyComponent에 handleClickModify(), handleClickDelete() 함수 정의, 버튼에 이벤트 처리



```
import { useEffect, useState } from "react";
import { deleteOne, getOne, putOne } from "../../api/todoApi";

const initState = { ...생략 }

const ModifyComponent = ({tno}) => {
  const [todo, setTodo] = useState({...initState})

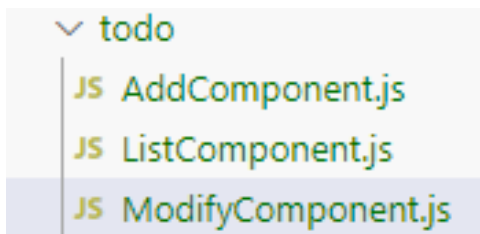
  //모달 창을 위한 상태
  const [result, setResult] = useState(null)

  useEffect(() => { getOne(tno).then( data => { setTodo(data) }) },[tno])

  const handleClickModify = () => { //버튼 클릭시
    putOne(todo).then(data => { console.log("modify result: " + data) })
  }
  const handleClickDelete = () => { //버튼 클릭시
    deleteOne(tno).then( data => { console.log("delete result: " + data) })
  }
}
```

수정/삭제와 모달창

→ ModifyComponent에 handleClickModify(), handleClickDelete() 함수 정의, 버튼에 이벤트 처리



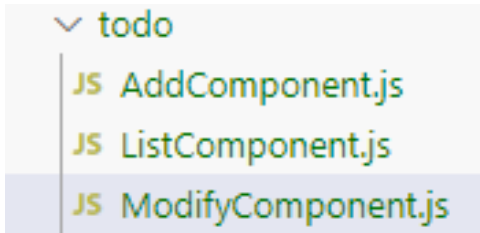
```
const handleChangeTodo = (e) => {...}
const handleChangeTodoComplete = (e) => {...}

return (
  <div className = "border-2 border-sky-200 mt-10 m-2 p-4">
    <div className="flex justify-center mt-10">
      ...생략
    <div className="flex justify-end p-4">
      <button type="button"
        className="inline-block rounded p-4 m-2 text-xl w-32 text-white bg-red-500"
        onClick={handleClickDelete} > Delete </button>
      <button type="button" onClick={handleClickModify}
        className="rounded p-4 m-2 text-xl w-32 text-white bg-blue-500"> Modify
      </button>
    </div>
  </div>
);
}

export default ModifyComponent;
```

수정/삭제와 모달창

→ ModifyComponent에 화면 이동에 필요한 기능을 가져오고 모달창이 close될 때 호출하도록 변경



```
import { useEffect, useState } from "react";
import { deleteOne, getOne, putOne } from "../../api/todoApi";
import useCustomMove from "../../hooks/useCustomMove";

import ResultModal from "../../common/ResultModal";

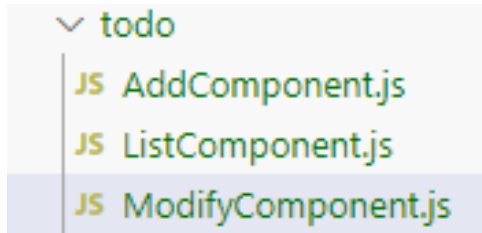
const initState = {  ...생략  }

const ModifyComponent = ({tno}) => {
  const [todo, setTodo] = useState({...initState})
  //모달 창을 위한 상태
  const [result, setResult] = useState(null)
  //이동을 위한 기능들
  const {moveToList, moveToRead} = useCustomMove()

  useEffect(() => { ... },[tno])
```

수정/삭제와 모달창

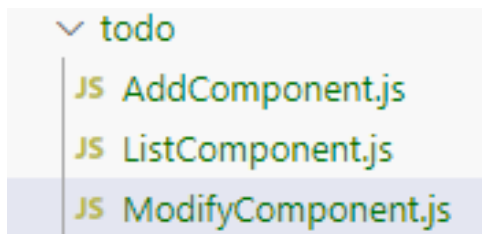
→ ModifyComponent에 화면 이동에 필요한 기능을 가져오고 모달창이 close될 때 호출하도록 변경



```
const handleClickModify = ( ) => {  
  putOne(todo).then(data => { setResult('Modified' ) })  
}  
  
const handleClickDelete = ( ) => {  
  deleteOne(tno).then( data => { setResult('Deleted' ) })  
}  
  
//모달 창이 close될때  
const closeModal = () => {  
  if(result === 'Deleted') {  
    moveToList()  
  }else {  
    moveToRead(tno)  
  }  
}  
  
...생략
```

수정/삭제와 모달창

→ ModifyComponent에 화면 이동에 필요한 기능을 가져오고 모달창이 close될 때 호출하도록 변경



```
return (  
  <div className = "border-2 border-sky-200 mt-10 m-2 p-4">  
    {result ? <ResultModal title={'처리결과'} content={result}  
      callbackFn={closeModal}></ResultModal> : <></>}  
    ...생략  
  );  
}  
  
export default ModifyComponent;
```

감사합니다.