YouTube
NAVER 카페
구멍가게 코딩단

코드로 배우는 리액트

6. 리액트와 상품 API 서버 연동

6장. 리액트와 상품 API 서버 연동

- 상품 API는 JSON 데이터 처리와 유사하나, 파일 데이터 추가로 처리 시간이 늘어남에 따라 모달 창 등의 부가 기능이 필요
- 컴포넌트 재사용을 통해 처리

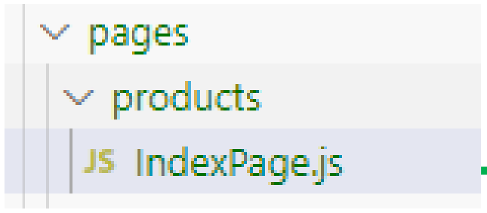
개발목표

1. 파일이 추가되는 데이터의 처리
2. 기존 공통 컴포넌트들의 재사용

Index

상품 관련 React-Router 설정

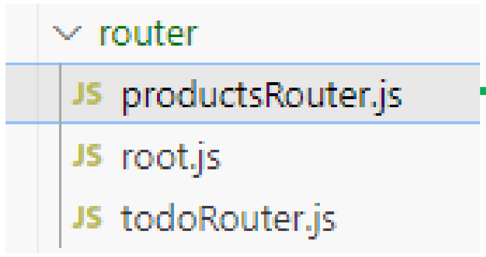
→ 상품 관련된 기능은 pages 폴더 내에 products 폴더를 생성



```
const IndexPage = () => {
  return (
    <></>
   );
}

export default IndexPage;
```

상품 관련 React-Router 설정

→ React-Router의 설정을 위해서 routes 폴더 내에 productsRouter.js 추가

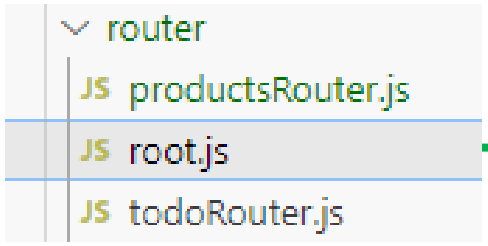


```
const productsRouter = () => {

  return [
  ]
}

export default productsRouter;
```

YouTube NAVER 카페

상품 관련 React-Router 설정

→ root.js에 productsRouter.js와 products폴더의 IndexPage컴포넌트를 추가

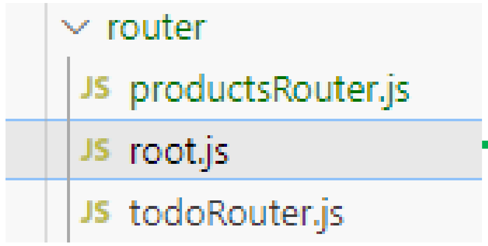

router
productsRouter.js
root.js
todoRouter.js

```
import { Suspense, lazy } from "react";
import todoRouter from ".todoRouter";
import productsRouter from "./productsRouter";

const { createBrowserRouter } = require("react-router-dom");

const Loading = <div>Loading....</div>
const Main = lazy(() => import("../pages/MainPage"))

const About = lazy(() => import("../pages/AboutPage"))

const TodoIndex = lazy(() => import("../pages/todo/IndexPage"))

const ProductsIndex = lazy(() => import("../pages/products/IndexPage"))

// 다음페이지에 이어집니다.
```

구멍가게 코딩단

상품 관련 React-Router 설정

→ root.js에 productsRouter.js와 products폴더의 IndexPage컴포넌트를 추가

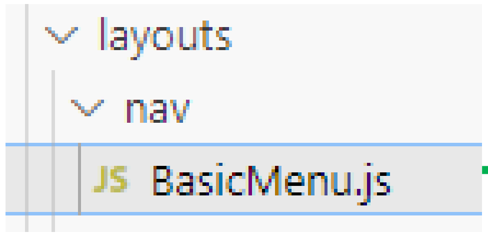


```
const root = createBrowserRouter([
 {
   path: "",
   element: <Suspense fallback={Loading}><Main/></Suspense>
 },{
   path: "about",
   element: <Suspense fallback={Loading}><About/></Suspense>
 }, {
   path: "todo",
   element: <Suspense fallback={Loading}><TodoIndex/></Suspense>,
   children: todoRouter()
 }, {
   path: "products",
   element: <Suspense fallback={Loading}><ProductsIndex/></Suspense>,
   children: productsRouter()
 }
])
export default root;
```

상품 관련 React-Router 설정

상품 메뉴의 추가

→ layouts/nav 폴더 내 BasicMenu.js 수정

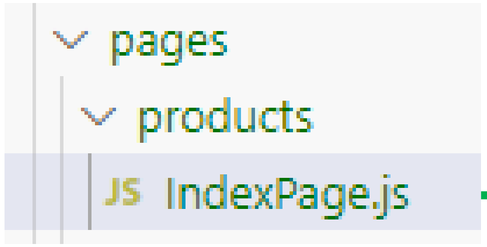


```
import { Link } from "react-router-dom";

const BasicMenu = () => {
  return (
  <nav id='navbar' className=" flex  bg-blue-300">
    <div className="w-4/5 bg-gray-500" >
      <ul className="flex p-4 text-white font-bold">
        <li className="pr-6 text-2xl"><Link to={'/'}>Main</Link></li>
        <li className="pr-6 text-2xl"><Link to={'/about'}>About</Link></li>
        <li className="pr-6 text-2xl"><Link to={'/todo/'}>Todo</Link></li>
        <li className="pr-6 text-2xl"><Link to={'/products/'}>Products</Link></li>
      </ul>
    </div>
    <div className="w-1/5 flex justify-end bg-orange-300 p-4 font-medium">
        <div className="text-white text-sm m-1 rounded" >Login</div>
    </div>
  </nav>
  );
}

export default BasicMenu;
```

상품 관련 React-Router 설정

상품 IndexPage

→ <Outlet>을 이용해서 조금 더 세밀한 레이아웃을 지정

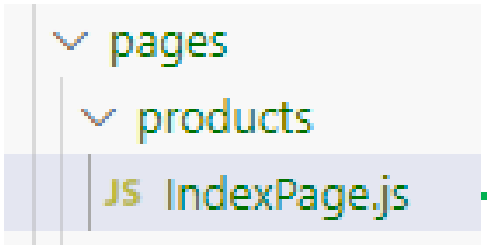


```
import { Outlet, useNavigate } from "react-router-dom";
import BasicLayout from "../../layouts/BasicLayout";
import { useCallback } from "react";

const IndexPage = () => {

  const navigate = useNavigate()

  const handleClickList = useCallback(() => {  navigate({ pathname:'list' })
  })

  const handleClickAdd = useCallback(() => {
    navigate({ pathname:'add' })
  })
```

상품 관련 React-Router 설정

상품 IndexPage

→ <Outlet>을 이용해서 조금 더 세밀한 레이아웃을 지정

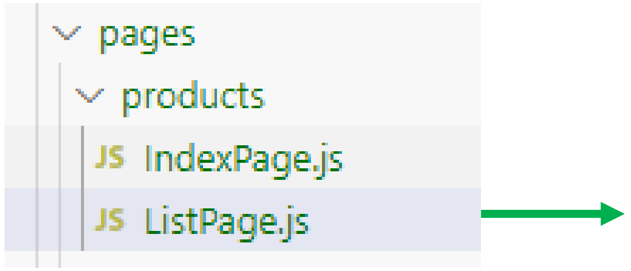


```
  return (
    <BasicLayout>
      <div className="text-black font-extrabold -mt-10">  Products Menus  </div>

      <div className="w-full flex m-2 p-2 ">
        <div className="text-xl m-1 p-2  w-20 font-extrabold text-center underline"
            onClick={handleClickList}>
          LIST</div>
        <div className="text-xl m-1 p-2 w-20 font-extrabold  text-center underline"
            onClick={handleClickAdd}>
          ADD
        </div>

      </div>
      <div className="flex flex-wrap w-full ">
        <Outlet/>
      </div>
    </BasicLayout>
   );
}

export default IndexPage;
```

상품 관련 React-Router 설정

ListPage

→ pages/products 폴더 내에 ListPage를 추가

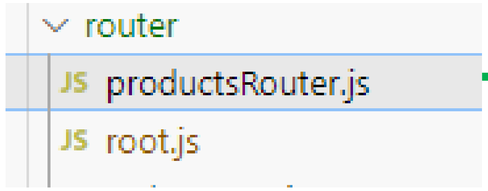


```
const ListPage = () => {
  return (
    <div className="w-full mt-4 border border-solid border-neutral-300
shadow-md">
      <div className="text-2xl m-4 font-extrabold">
        Products List Page
      </div>

    </div>
  );
}

export default ListPage;
```

상품 관련 React-Router 설정

ListPage

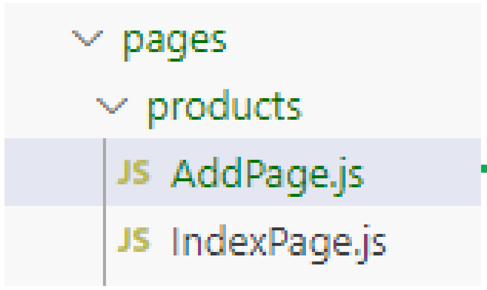
→ productsRouter.js 에서 라우팅 설정



```
import { Suspense, lazy } from "react";
import { Navigate } from "react-router-dom";

const productsRouter = () => {

  const Loading = <div>Loading....</div>
  const ProductsList =  lazy(() => import("../pages/products/ListPage"))

  return [{
      path: "list",
      element: <Suspense fallback={Loading}><ProductsList/></Suspense>
    },{
      path: "",
      element: <Navigate replace to="/products/list"/>
    },
  ]
}
export default productsRouter;
```

등록 페이지와 컴포넌트 처리

→ AddPage.js

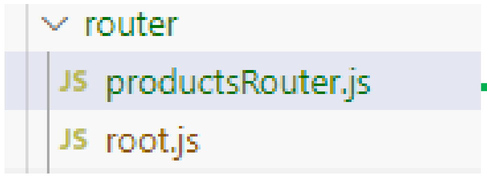


```
const AddPage = () => {

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

  </div>
   );
}

export default AddPage;
```

등록 페이지와 컴포넌트 처리

라우팅 설정

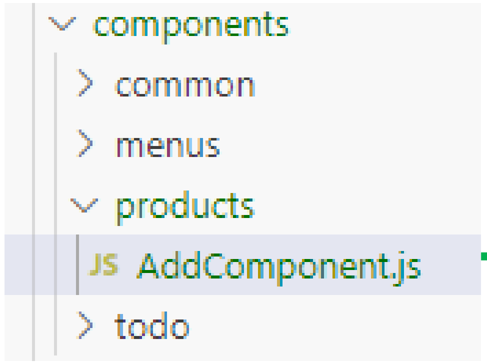
→ productsRouter.js



```
const productsRouter = () => {

  ...생략
  const ProductsAdd = lazy(() => import("../pages/products/AddPage"))

  return [
   ...생략
    {
      path: "",
      element: <Navigate replace to="/products/list"/>
    },
    {
      path: "add",
      element: <Suspense fallback={Loading}><ProductsAdd/></Suspense>
    }
  ]
}
export default productsRouter;
```

상품의 AddComponent와 API 호출

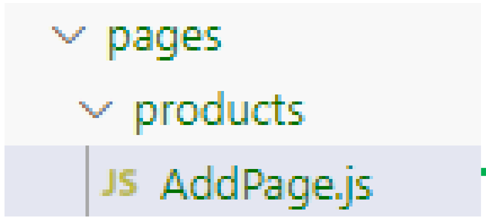
→ components 폴더에 products 폴더를 생성하고 AddComponent 생성



```
const AddComponent = () => {

  return (
    <div className = "border-2 border-sky-200 mt-10 m-2 p-4">
      <div className="flex justify-center">
        <h1>Add Component</h1>
      </div>
    </div>
  );
}

export default AddComponent;
```

상품의 AddComponent와 API 호출

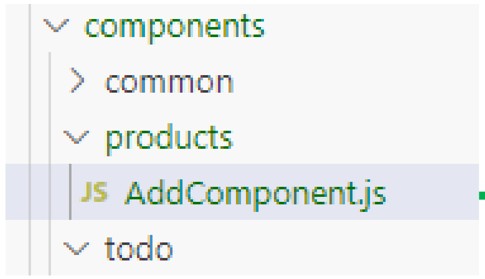
→ /pages/products/AddPage.js 내 AddComponent를 추가



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

const AddPage = () => {

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

    <AddComponent/>

  </div>
   );
}

export default AddPage;
```

상품의 AddComponent와 API 호출

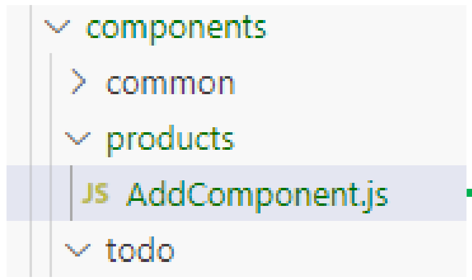
→ AddComponent에 /todo/AddComponents를 추가하여 첨부파일 기능 추가



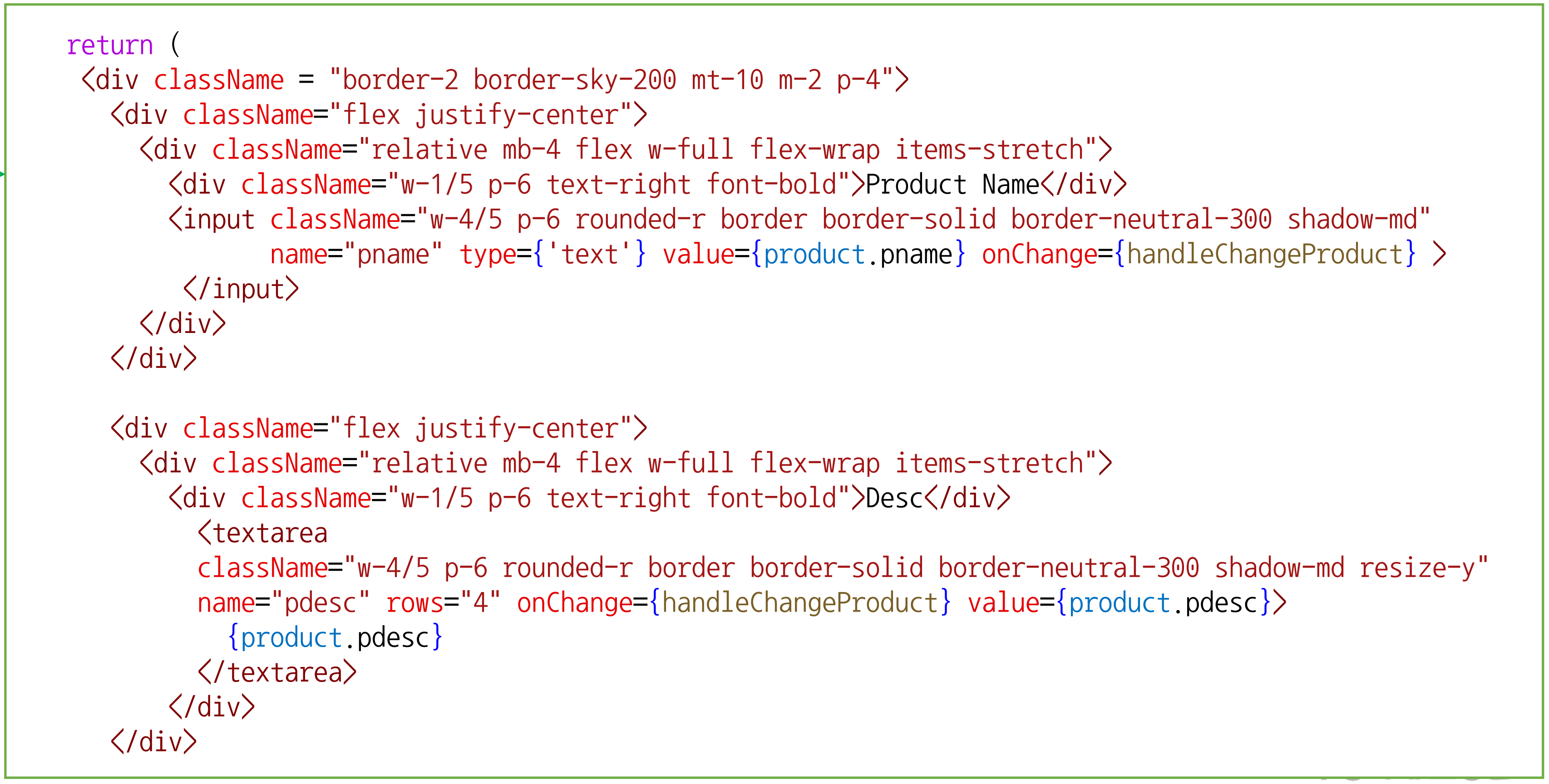
```
import { useRef, useState } from "react";

const initState = { pname: '',  pdesc: '',  price: 0,  files: []  }

const AddComponent = () => {

  const [product,setProduct] = useState({...initState})

  const uploadRef = useRef()

  const handleChangeProduct = (e) => {
    product[e.target.name] = e.target.value
    setProduct({...product})
  }

  const handleClickAdd = (e) => {
    console.log(product)
  }
```

상품의 AddComponent와 API 호출

→ AddComponent에 첨부파일 기능 추가



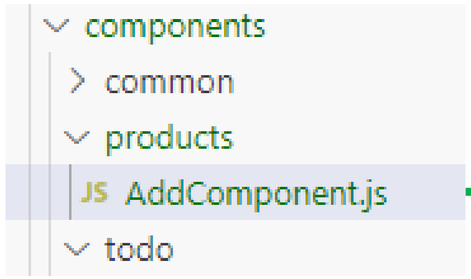
```
components
  common
  products
    AddComponent.js
  todo
```



```
return (
 <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">Product Name</div>
       <input className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md"
              name="pname" type={'text'} value={product.pname} onChange={handleChangeProduct} >
        </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">Desc</div>
         <textarea
         className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md resize-y"
         name="pdesc" rows="4" onChange={handleChangeProduct} value={product.pdesc}>
           {product.pdesc}
         </textarea>
       </div>
   </div>
```

상품의 AddComponent와 API 호출

→ AddComponent에 첨부파일 기능 추가

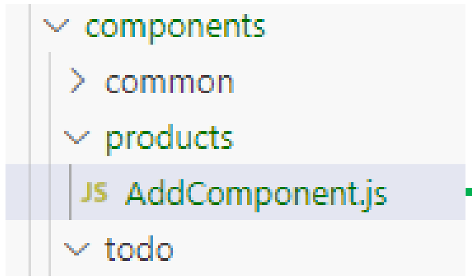


```
<div className="flex justify-center">
  <div className="relative mb-4 flex w-full flex-wrap items-stretch">
    <div className="w-1/5 p-6 text-right font-bold">Price</div>
    <input
           className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md"
           name="price" type={'number'} value={product.price} onChange={handleChangeProduct}>
     </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">Files</div>
    <input
      ref={uploadRef}
      className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md"
      type={'file'} multiple={true}>
    </input>
  </div>
</div>
```

YouTube NAVER 카페

상품의 AddComponent와 API 호출

→ AddComponent에 첨부파일 기능 추가



```
      <div className="flex justify-end">
        <div className="relative mb-4 flex p-4 flex-wrap items-stretch">
          <button type="button"
                  className="rounded p-4 w-36 bg-blue-500 text-xl  text-white "
                  onClick={handleClickAdd} >
            ADD
          </button>
        </div>
      </div>

    </div>
  );
}

export default AddComponent;
```

구멍가게 코딩단

등록 페이지와 컴포넌트 처리

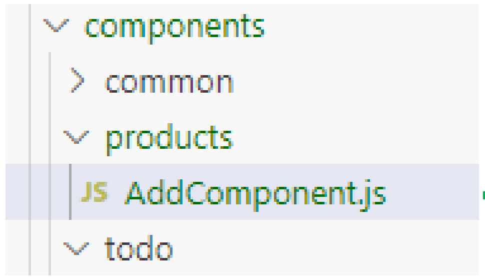
상품의 AddComponent와 API 호출

→ useRef( )와 FormData

- useRef() : <input type='file'>의 value 속성값을 읽어 옴
- FormData 객체 : 파일 정보를 읽어와 FormData 객체로 구성하고 Axios로 서버 호출 시 사용

상품의 AddComponent와 API 호출

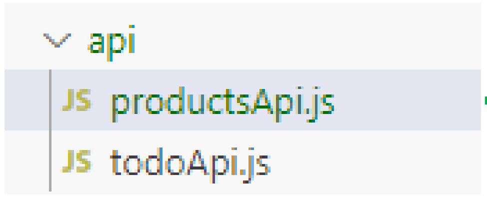
→ useRef( )와 FormData



```
const handleClickAdd = (e) => {

  const files = uploadRef.current.files

  const formData = new FormData()

  for (let i = 0; i < files.length; i++) {
    formData.append("files", files[i]);
  }

  //other data
  formData.append("pname", product.pname)
  formData.append("pdesc", product.pdesc)
  formData.append("price", product.price)

  console.log(formData)

}
```

상품의 AddComponent와 API 호출

→ productsAPI의 개발

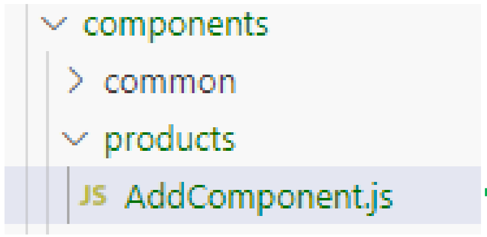


```
import axios from "axios"
import { API_SERVER_HOST } from "./todoApi"

const host = `${API_SERVER_HOST}/api/products`

export const postAdd = async (product) => {

  const header = {headers: {"Content-Type": "multipart/form-data"}}

  // 경로 뒤 '/' 주의
  const res = await axios.post(`${host}/`, product, header)

  return res.data

}
```

YouTube NAVER 카페

상품의 AddComponent와 API 호출

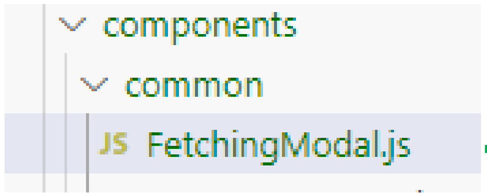
→ productsAPI의 개발



```
import { useRef, useState } from "react";
import { postProduct } from "../../api/productsApi";
…생략

const AddComponent = () => {
 …생략

  const handleClickAdd = (e) => {
   const files = uploadRef.current.files
   const formData = new FormData()
   for (let i = 0; i < files.length; i++) {
     formData.append("files", files[i]);
   }
   //other data
   formData.append("pname", product.pname)
   formData.append("pdesc", product.pdesc)
   formData.append("price", product.price)

   //console.log(formData)
   postProduct(formData)
  }
…생략
```

상품의 AddComponent와 API 호출

→ 진행 모달창과 결과 모달창 : components/common 폴더의 ResultModal 컴포넌트를 추가

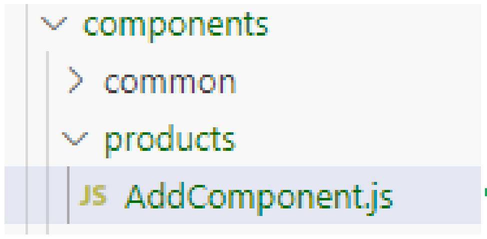


```
const FetchingModal = (  ) => {
  return (
    <div
    className={`fixed top-0 left-0 z-[1055] flex h-full w-full  place-items-
center justify-center bg-black bg-opacity-20`}>
      <div
      className=" bg-white rounded-3xl opacity-100 min-w-min h-1/4  min-w-
[600px] flex justify-center items-center ">

        <div className="text-4xl font-extrabold text-orange-400 m-20">
          Loading.....
        </div>
      </div>
    </div>
   );
}

export default FetchingModal;
```

YouTube NAVER 카페

상품의 AddComponent와 API 호출

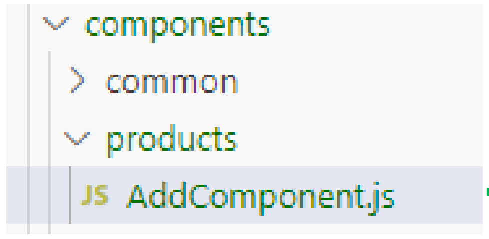
→ 모달창 처리 : 서버와의 통신 상태를 fetching이라는 useState( )를 통해서 제어



```
import FetchingModal from "../common/FetchingModal";

  const [product,setProduct] = useState({...initState})
  const uploadRef = useRef()
  const [fetching, setFetching] = useState(false)
  const handleChangeProduct = (e) => {…}
  const handleClickAdd = (e) => {

   …생략

    setFetching(true)
    postAdd(formData).then(data=> {
      setFetching(false)
    })
  }

  return (
    <div className = "border-2 border-sky-200 mt-10 m-2 p-4">
       {fetching? <FetchingModal/> :<></>}
 …이하 생략
```

상품의 AddComponent와 API 호출

→ 모달창 처리 : 결과 모달창 처리

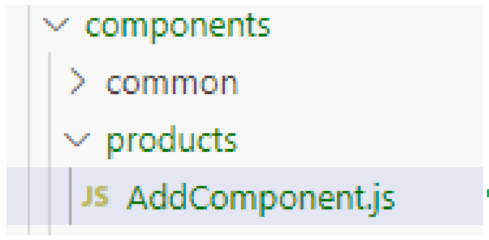


```
import { useRef, useState } from "react";
import { postAdd } from "../../api/productsApi";
import FetchingModal from "../common/FetchingModal";
import ResultModal from "../common/ResultModal";

const initState = {  pname: '',  pdesc: '',  price: 0,  files: []  }

const AddComponent = () => {

  const [product,setProduct] = useState({...initState})

  const uploadRef = useRef()

  const [fetching, setFetching] = useState(false)
  const [result, setResult] = useState(null)

  const handleChangeProduct = (e) => {…}
```

YouTube NAVER 카페

상품의 AddComponent와 API 호출

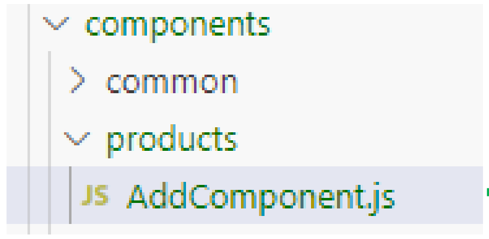
→ 모달창 처리 : 결과 모달창 처리



```
const handleClickAdd = (e) => {

  const files = uploadRef.current.files
  const formData = new FormData()
  for (let i = 0; i < files.length; i++) {
    formData.append("files", files[i]);
  }
  //other data
  formData.append("pname", product.pname)
  formData.append("pdesc", product.pdesc)
  formData.append("price", product.price)
  console.log(formData)
  setFetching(true)
  postAdd(formData).then(data => {
    setFetching(false)
    setResult(data.result)
  })
}
```

상품의 AddComponent와 API 호출

→ 모달창 처리 : 결과 모달창 처리



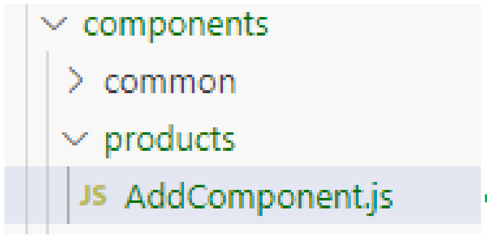
```
const closeModal = () => { //ResultModal 종료
  setResult(null)
}

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

    {fetching? <FetchingModal/> :<></>}

    {result?
      <ResultModal
      title={'Product Add Result'}
      content={`${result}번 등록 완료`}
      callbackFn ={closeModal}
      />
      : <></>
    }
```

…이하 생략

YouTube NAVER 카페

상품의 AddComponent와 API 호출

→ 등록 후 목록 페이지 이동

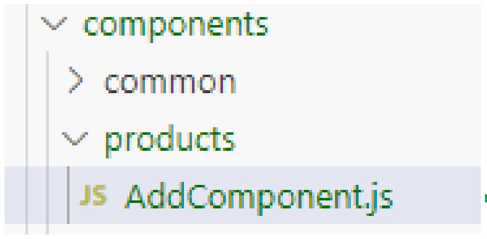


```
import { useRef, useState } from "react";
import { postAdd } from "../../api/productsApi";
import FetchingModal from "../common/FetchingModal";
import ResultModal from "../common/ResultModal";
import useCustomMove from "../../hooks/useCustomMove";

const initState = { … }

const AddComponent = () => {
  const [product,setProduct] = useState({...initState})
  const uploadRef = useRef()
  //for FetchingModal
  const [fetching, setFetching] = useState(false)
  //for ResultModal
  const [result, setResult] = useState(null)
  const {moveToList} = useCustomMove() //이동을 위한 함수
  const handleChangeProduct = (e) => {…}
```

02 등록 페이지와 컴포넌트 처리

상품의 AddComponent와 API 호출

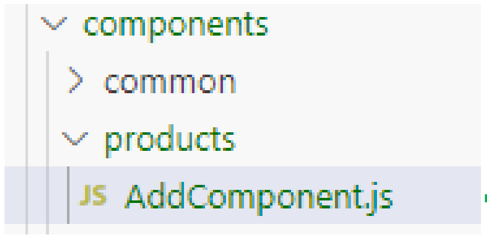
→ 등록 후 목록 페이지 이동



```
const handleClickAdd = (e) => {
  const files = uploadRef.current.files
  const formData = new FormData()
  for (let i = 0; i < files.length; i++) {
    formData.append("files", files[i]);
  }
  //other data
  formData.append("pname", product.pname)
  formData.append("pdesc", product.pdesc)
  formData.append("price", product.price)
  console.log(formData)
  setFetching(true)
  postAdd(formData).then(data => {
    setFetching(false)
    setResult(data.result)
  })
}
```

상품의 AddComponent와 API 호출

→ 등록 후 목록 페이지 이동



```
const closeModal = () => {
  setResult(null)
  moveToList({page:1}) //모달 창이 닫히면 이동
}
…이하 생략
```

목록 페이지와 목록 컴포넌트 처리

공통 코드를 커스텀 훅으로 만들기

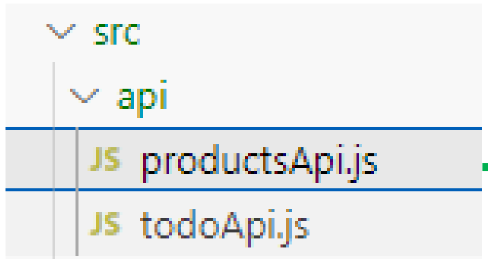
→ 코드 분리 및 재사용 목적으로 커스텀 훅(Custom Hook) 활용.

→ 커스텀 훅은 공통 로직이나 상태 재사용을 위한 함수로 작성.

→ 커스텀 훅 함수명은 'use-'로 시작하는 규칙을 따르며, 사용 방식은 다른 훅과 유사.

공통 코드를 커스텀 훅으로 만들기

→ api/productsApi.js에는 서버에서 목록 데이터를 가져오기 위한 함수 추가

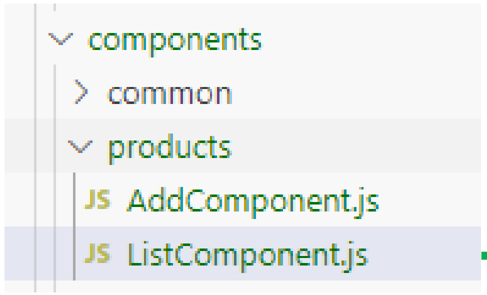


```
import axios from "axios"
import { API_SERVER_HOST } from "./todoApi"

const host = `${API_SERVER_HOST}/api/products`

export const postAdd = async (product) => {…}

export const getList = async ( pageParam ) => {

  const {page,size} = pageParam

  const res = await axios.get(`${host}/list`, {params:
{page:page,size:size }})

  return res.data

}
```

목록 페이지와 목록 컴포넌트 처리

ListComponent 처리

→ components/products/ListComponent.js

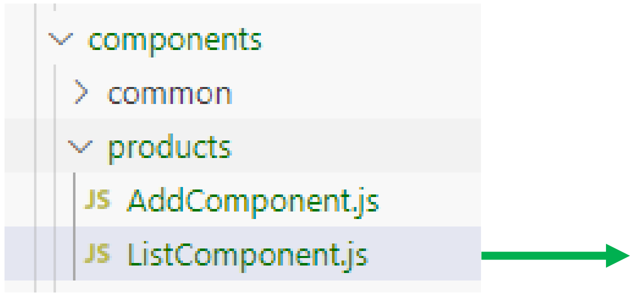


```
import { useEffect, useState } from "react";
import { getList } from "../../api/productsApi";
import usePageMove from "../../hooks/usePageMove";

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

목록 페이지와 목록 컴포넌트 처리

ListComponent 처리

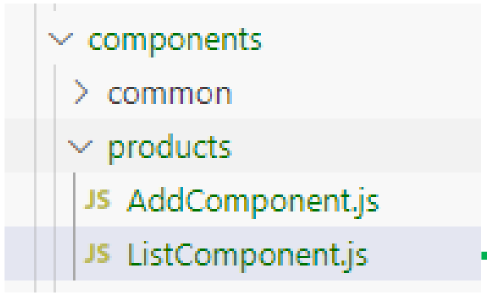
→ components/products/ListComponent.js



```
const ListComponent = () => {

  const {page, size, refresh, moveToList, moveToRead} = useCustomMove()

  //serverData는 나중에 사용
  const [serverData, setServerData] = useState(initState)

  //for FetchingModal
  const [fetching, setFetching] = useState(false)

  useEffect(() => {
    setFetching(true)
    getList({page,size}).then(data => {
      console.log(data)
      setServerData(data)
      setFetching(false)
    })
  }, [page,size, refresh])
```

ListComponent 처리

→ components/products/ListComponent.js

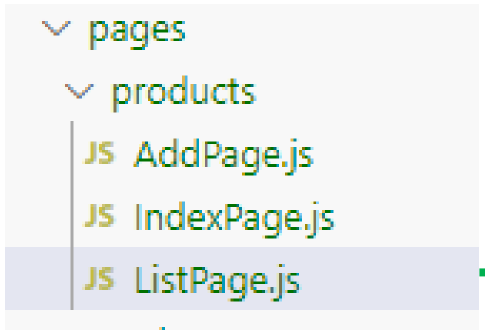


```
  return (
  <div className="border-2 border-blue-100 mt-10 mr-2 ml-2">

    <h1>Products List Component</h1>


    {fetching? <FetchingModal/> :<></>}


  </div>

  );
}

export default ListComponent;
```

목록 페이지와 목록 컴포넌트 처리

ListComponent 처리

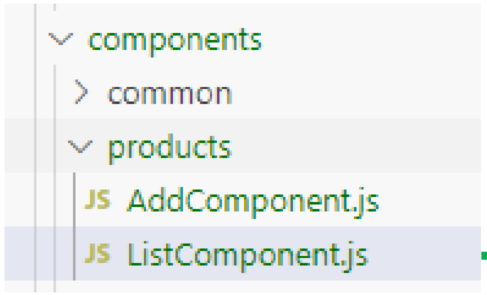
→ ListPage에서 ListComponent를 import해서 사용



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

const ListPage = () => {

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

    <ListComponent/>

  </div>
   );
}

export default ListPage;
```

ListComponent 처리

→ 목록 데이터의 출력



```
import { API_SERVER_HOST } from "../../api/todoApi";

const host = API_SERVER_HOST
```

목록 페이지와 목록 컴포넌트 처리

ListComponent 처리

```
return (
<div className="border-2 border-blue-100 mt-10 mr-2 ml-2">

  {fetching? <FetchingModal/> :<></>}

  <div className="flex flex-wrap mx-auto p-6">

    {serverData.dtoList.map(product =>

    <div key= {product.pno} className="w-1/2 p-1 rounded shadow-md border-2" onClick={() => moveToRead(product.pno)}>
      <div className="flex flex-col  h-full">
        <div className="font-extrabold text-2xl p-2 w-full ">{product.pno}</div>
        <div className="text-1xl m-1 p-2 w-full flex flex-col">
          <div className="w-full overflow-hidden ">
            <img alt="product" className="m-auto rounded-md w-60"
                src={`${host}/api/products/view/s_${product.uploadFileNames[0]}`}/>
          </div>
```

목록 페이지와 목록 컴포넌트 처리

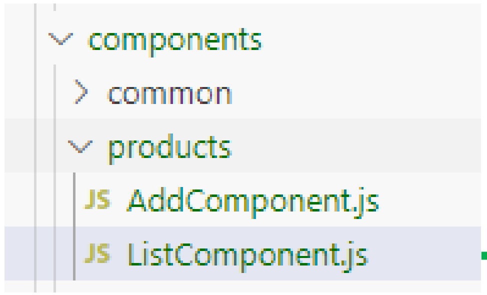
ListComponent 처리

```
          <div className="bottom-0 font-extrabold bg-white">
            <div className="text-center p-1">
              이름: {product.pname}
            </div>
            <div className="text-center p-1">
              가격: {product.price}
            </div>
          </div>

        </div>
      </div>
    </div>
    )}
  </div>
</div>

);
```

목록 페이지와 목록 컴포넌트 처리

ListComponent 처리

→ 페이지의 이동

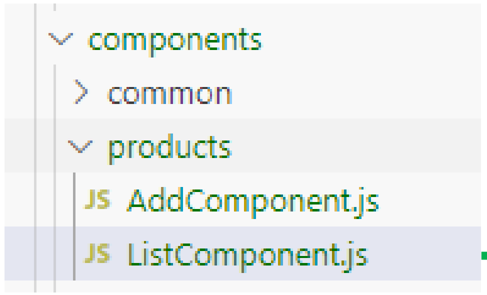


```
import PageComponent from "../common/PageComponent";

…생략

const ListComponent = () => {
  const {page, size, refresh, moveToList, moveToRead} = useCustomMove()
  //serverData는 나중에 사용
  const [serverData, setServerData] = useState(initState)
  //for FetchingModal
  const [fetching, setFetching] = useState(false)

  useEffect(() => {
    setFetching(true)
    getList({page,size}).then(data => {
      console.log(data)
      setServerData(data)
      setFetching(false)
    })  }, [page,size, refresh])
```

목록 페이지와 목록 컴포넌트 처리

ListComponent 처리

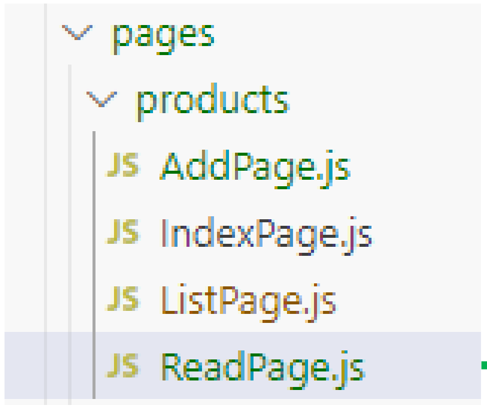
→ 페이지의 이동



```
  return (
  <div className="border-2 border-blue-100 mt-10 mr-2 ml-2">

    …생략

    <PageComponent serverData={serverData} movePage={moveToList}></PageComponent>

  </div>

  );
}

export default ListComponent;
```

조회 페이지와 조회 컴포넌트

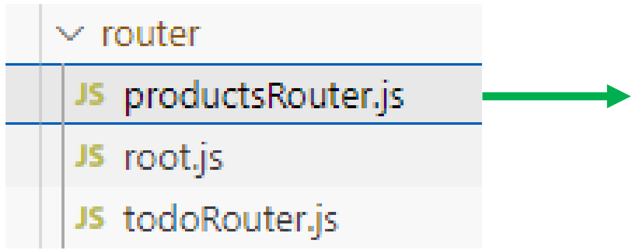
→ pages/products/ReadPage.js



```
const ReadPage = () => {
  return (
  <div className="p-4 w-full bg-white">
    <div className="text-3xl font-extrabold">
      Products Read Page
    </div>

  </div>
  );
}

export default ReadPage;
```

라우터 설정

→ router/productsRouter.js에 ReadPage에 대한 설정

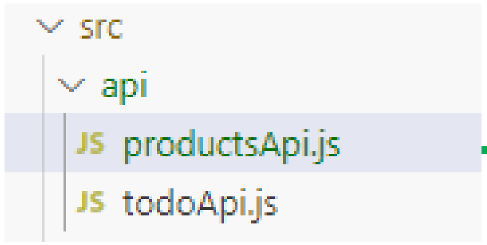


```
import { Suspense, lazy } from "react";
import { Navigate } from "react-router-dom";

…생략

const ProductRead = lazy(() => import("../pages/products/ReadPage"))

const productsRouter = () => {

  return [
   …생략
    {
      path: "read/:pno",
      element: <Suspense fallback={Loading}><ProductRead/></Suspense>
    }
  ]
}

export default productsRouter;
```

조회 페이지와 조회 컴포넌트

ReadComponent 처리

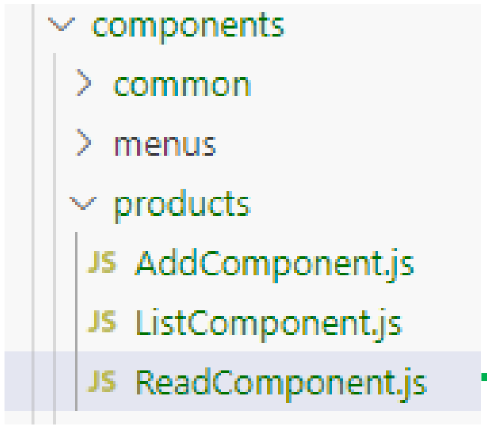
→ api/productsAPi.js



```
import axios from "axios"
import { API_SERVER_HOST } from "./todoApi"

const host = `${API_SERVER_HOST}/api/products`

…생략

export const getOne = async (tno) => {

  const res = await axios.get(`${host}/${tno}` )

  return res.data

}
```

ReadComponent 처리

→ components/products 폴더에 ReadComponent.js 추가

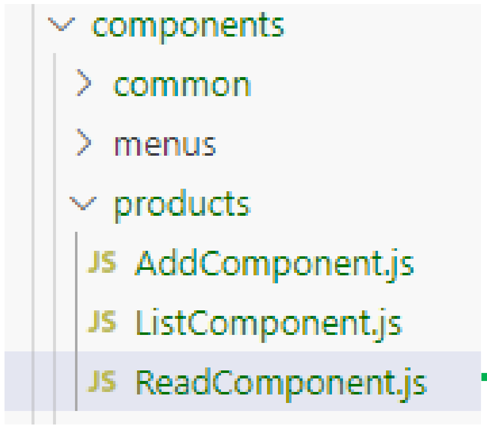


```
import { useEffect, useState } from "react"
import  {getOne} from "../../api/productsApi"
import { API_SERVER_HOST } from "../../api/todoApi"
import useCustomMove from "../../hooks/useCustomMove"
import FetchingModal from "../common/FetchingModal"

const initState = { pno:0, pname: '', pdesc: '', price: 0,  uploadFileNames:[] }

const host = API_SERVER_HOST

const ReadComponent = ({pno }) => {

  const [product, setProduct] = useState(initState)
  //화면 이동용 함수
  const {moveToList, moveToModify} = useCustomMove()
  //fetching
  const [fetching, setFetching] = useState(false)
```

YouTube NAVER 카페

ReadComponent 처리

→ components/products 폴더에 ReadComponent.js 추가

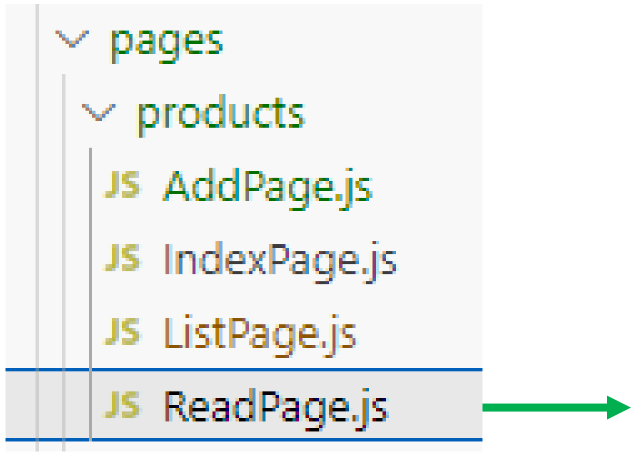


```
useEffect(() => {
  setFetching(true)
  getOne(pno).then(data => {
    setProduct(data)
    setFetching(false)
  })}, [pno])
return (
  <div className = "border-2 border-sky-200 mt-10 m-2 p-4">
  {fetching? <FetchingModal/> :<></>}
  <div className="flex justify-center mt-10">
    <div className="relative mb-4 flex w-full flex-wrap items-stretch">
      <div className="w-1/5 p-6 text-right font-bold">PNO</div>
      <div className="w-4/5 p-6 rounded-r border border-solid shadow-md">
        {product.pno}
      </div>
    </div>
  </div>
  </div>
)}
```

조회 페이지와 조회 컴포넌트

ReadComponent 처리

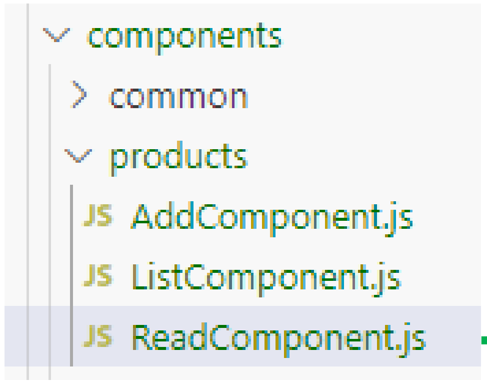
→ ReadPage.js



```
import { useParams } from "react-router-dom";
import ReadComponent from "../../components/products/ReadComponent";

const ReadPage = () => {
  const {pno} = useParams()
  return (
  <div className="p-4 w-full bg-white">
    <div className="text-3xl font-extrabold">
      Products Read Page
    </div>
    <ReadComponent pno={pno}></ReadComponent>
  </div>
  );
}

export default ReadPage;
```

ReadComponent 처리

→ 데이터 출력과 이동

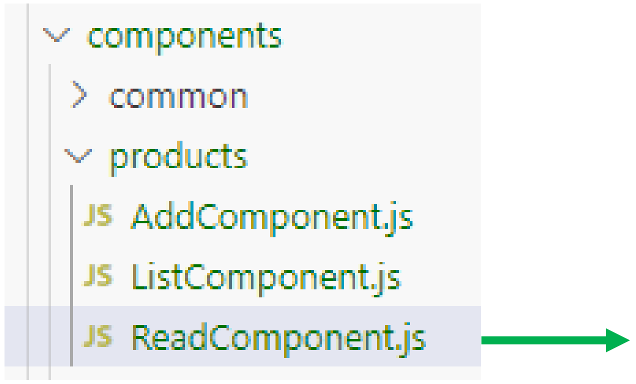


```
import { useEffect, useState } from "react"
import  {getOne} from "../../api/productsApi"
import { API_SERVER_HOST } from "../../api/todoApi"
import useCustomMove from "../../hooks/useCustomMove"
import FetchingModal from "../common/FetchingModal"

const initState = {
  pno:0,
  pname: '',
  pdesc: '',
  price: 0,
  uploadFileNames:[]
}

const host = API_SERVER_HOST
```

구멍가게 코딩단

ReadComponent 처리

→ 데이터 출력과 이동



```
const ReadComponent = ({pno }) => {

  const [product, setProduct] = useState(initState)
  //화면 이동용 함수
  const {moveToList, moveToModify} = useCustomMove()
  //fetching
  const [fetching, setFetching] = useState(false)

  useEffect(() => {
    setFetching(true)
    getOne(pno).then(data => {
      setProduct(data)
      setFetching(false)

    })
  }, [pno])
```

ReadComponent 처리

```
return (

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

  {fetching? <FetchingModal/> :<></>}

    <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">PNO</div>
        <div className="w-4/5 p-6 rounded-r border border-solid shadow-md">{product.pno}</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">PNAME</div>
        <div className="w-4/5 p-6 rounded-r border border-solid shadow-md">{product.pname}</div>
      </div>
    </div>
```

ReadComponent 처리

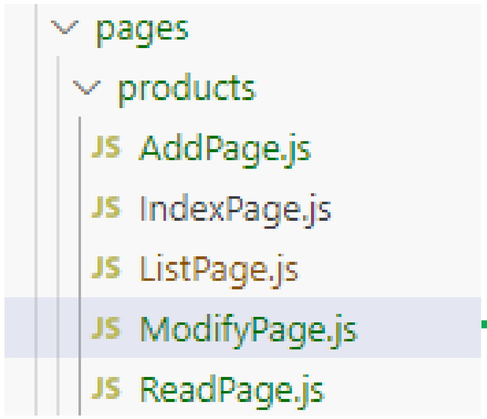
```
<div className="flex justify-center">
  <div className="relative mb-4 flex w-full flex-wrap items-stretch">
    <div className="w-1/5 p-6 text-right font-bold">PRICE</div>
    <div className="w-4/5 p-6 rounded-r border border-solid shadow-md">{product.price}</div>
  </div>
</div>
<div className="flex 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">PDESC</div>
    <div className="w-4/5 p-6 rounded-r border border-solid shadow-md">{product.pdesc}</div>
  </div>
</div>
<div className="w-full justify-center flex  flex-col m-auto items-center">
  {product.uploadFileNames.map( (imgFile, i) =>
    <img alt ="product" key={i} className="p-4 w-1/2" src={`${host}/api/products/view/${imgFile}`}/>
  )}
</div>
```

ReadComponent 처리

```
      <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={() => moveToModify(pno)}
        >
          Modify
        </button>
        <button type="button" className="rounded p-4 m-2 text-xl w-32 text-white bg-blue-500" onClick={moveToList}>
          List
        </button>
      </div>
    </div>
  )
}

export default ReadComponent
```

수정/삭제 페이지와 컴포넌트 처리

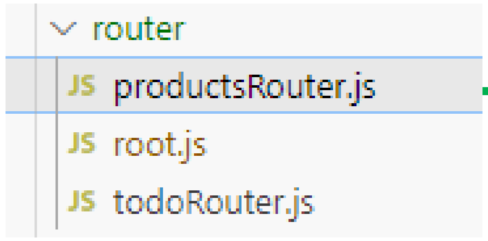
→ pages/products/ModifyPage.js



```
const ModifyPage = () => {
  return (
  <div className="p-4 w-full bg-white">
    <div className="text-3xl font-extrabold">
      Products Modify Page
    </div>
  </div>
   );
}

export default ModifyPage;
```

라우팅 설정

→ router/productsRouter.js 내 ModifyPage 설정 추가

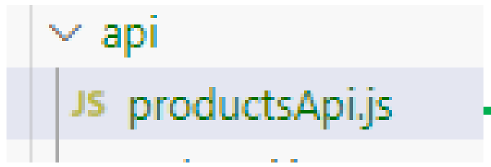


```
import { Suspense, lazy } from "react";
import { Navigate } from "react-router-dom";

…

const ProductModify = lazy(() => import("../pages/products/ModifyPage"))

const productsRouter = () => {
  return [
   …생략
    {
      path: "modify/:pno",
      element: <Suspense fallback={Loading}><ProductModify/></Suspense>
    }
   ]
}

export default productsRouter;
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

→ api/productsApi.js 파일에 수정/삭제를 위한 함수 추가

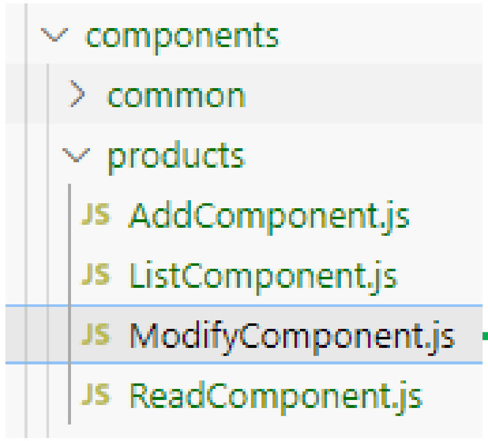


```
export const putProduct = async (pno, product) => {

  const header = {headers: {"Content-Type": "multipart/form-data"}}

  const res = await axios.put(`${host}${pno}`, product, header)

  return res.data

}

export const deleteProduct = async (pno) => {

  const res = await axios.delete(`${host}${pno}`)

  return res.data

}
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

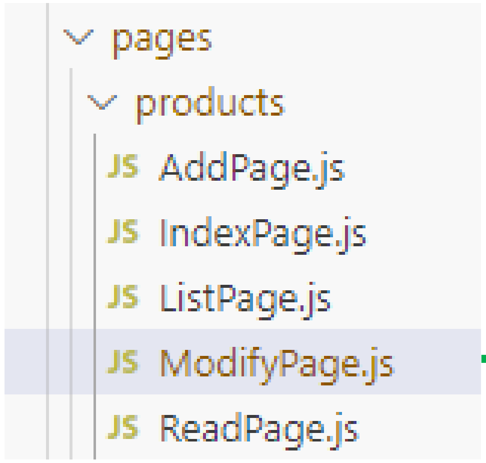
→ components/products 폴더에 ModifyComponent 추가


components
common
products
AddComponent.js
ListComponent.js
ModifyComponent.js
ReadComponent.js

```
const ModifyComponent = ({pno}) => {

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

    </div>
   );
}

export default ModifyComponent;
```

ModifyComponent 처리

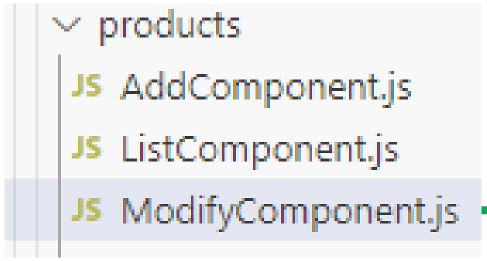
→ ModifyPage에 ModifyComponent import



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

const ModifyPage = () => {

  const {pno} = useParams()

  return (
  <div className="p-4 w-full bg-white">
    <div className="text-3xl font-extrabold">
      Products Modify Page
    </div>
    <ModifyComponent pno={pno}/>
  </div>
   );
}

export default ModifyPage;
```

ModifyComponent 처리

→ 데이터 출력

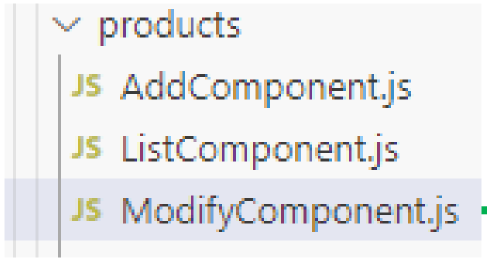


```
import { useEffect, useState } from "react";
import { getOne } from "../../api/productsApi";
import FetchingModal from "../common/FetchingModal";

const initState = {
  pno:0,
  pname: '',
  pdesc: '',
  price: 0,
  delFlag:false,
  uploadFileNames:[]
}

const ModifyComponent = ({pno}) => {

  const [product, setProduct] = useState(initState)
  const [fetching, setFetching] = useState(false)
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

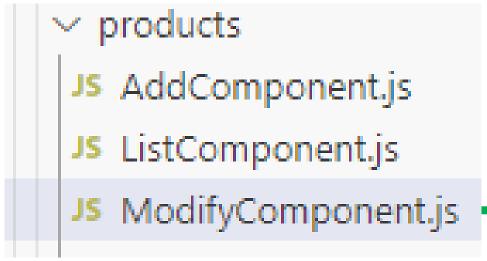
→ 데이터 출력



```
  useEffect(() => {
    setFetching(true)
    getOne(pno).then(data => {
      setProduct(data)
      setFetching(false)
    } )
  },[pno])

  return (
    <div className = "border-2 border-sky-200 mt-10 m-2 p-4">
       Product Modify Component
       {fetching? <FetchingModal/> :<></>}
    </div>
   );
}

export default ModifyComponent;
```

ModifyComponent 처리

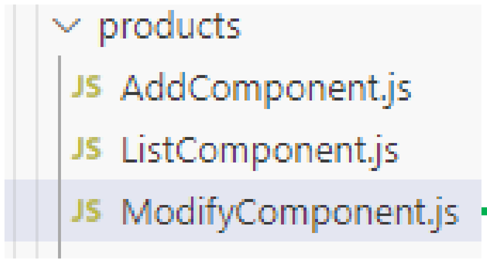
→ 변경 가능한 데이터를 input 으로 변경, 변경 기능 구성



```
import { useEffect, useRef, useState } from "react";
import { getOne } from "../../api/productsApi";
import FetchingModal from "../common/FetchingModal";
import { API_SERVER_HOST } from "../../api/todoApi";

const initState = {
  pno:0,
  pname: '',
  pdesc: '',
  price: 0,
  delFlag:false,
  uploadFileNames:[]
}

const host = API_SERVER_HOST
```

ModifyComponent 처리

→ 변경 가능한 데이터를 input 으로 변경, 변경 기능 구성



```
const ModifyComponent = ({pno}) => {
  const [product, setProduct] = useState(initState)
  const [fetching, setFetching] = useState(false)
  const uploadRef = useRef()
  useEffect(() => {
    setFetching(true)
    getOne(pno).then(data => {
      setProduct(data)
      setFetching(false)
    } )
  },[pno])

  const handleChangeProduct = (e) => {
    product[e.target.name] = e.target.value
    setProduct({...product})
  }

  const deleteOldImages = (imageName) => { }
```

ModifyComponent 처리

→ 변경 가능한 데이터를 input 으로 변경, 변경 기능 구성

```
return (
<div className = "border-2 border-sky-200 mt-10 m-2 p-4">
  {fetching? <FetchingModal/> :<></>}
  <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">Product Name</div>
      <input className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md"
        name="pname" type={'text'} value={product.pname} onChange={handleChangeProduct} ></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">Desc</div>
        <textarea className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md resize-y"
        name="pdesc" rows="4" onChange={handleChangeProduct} value={product.pdesc}> {product.pdesc} </textarea>
      </div>
  </div>
```

ModifyComponent 처리

→ 변경 가능한 데이터를 input 으로 변경, 변경 기능 구성

```
    <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">Price</div>
        <input className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md"
               name="price" type={'number'} value={product.price} onChange={handleChangeProduct}></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">DELETE</div>
          <select name="delFlag" value={product.delFlag} onChange={handleChangeProduct}
                 className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md">
            <option value={false}>사용</option>
            <option value={true}>삭제</option>
          </select>
      </div>
    </div>
```

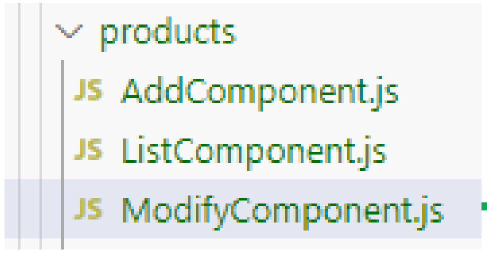
ModifyComponent 처리

```
     <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">Files</div>
        <input ref={uploadRef} className="w-4/5 p-6 rounded-r border border-solid border-neutral-300 shadow-md"
               type={'file'} multiple={true}></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">Images</div>
        <div className="w-4/5 justify-center flex flex-wrap items-start">
        {product.uploadFileNames.map( (imgFile, i) =>
            <div className="flex justify-center flex-col w-1/3 m-1 align-baseline" key = {i}>
              <button className="bg-blue-500 text-3xl text-white">DELETE</button>
              <img alt ="img" src={`${host}/api/products/view/s_${imgFile}`}/>
            </div>
          )}
        </div>
      </div>
    </div>
  </div>
  );
}

export default ModifyComponent;
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

→ 기존 이미지의 삭제

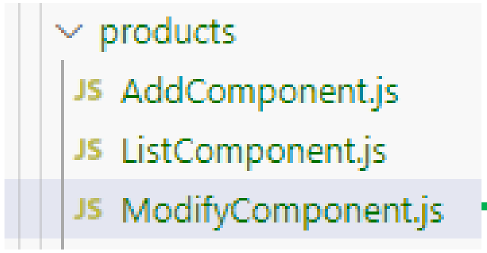


```
const deleteOldImages = (imageName) => {
  const resultFileNames = product.uploadFileNames.filter(fileName => fileName !== imageName)
  product.uploadFileNames = resultFileNames
  setProduct({...product})
}
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

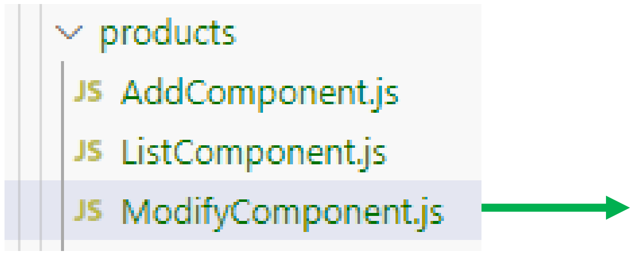
→ 기존 이미지의 삭제



```
{product.uploadFileNames.map( (imgFile, i) =>
    <div
      className="flex justify-center flex-col w-1/3"
      key = {i}>
      <button className="bg-blue-500 text-3xl text-white"
      onClick={() => deleteOldImages(imgFile)}
      >DELETE</button>
      <img
      alt ="img"
      src={`${host}/api/products/view/s_${imgFile}`}/>

    </div>
  )}
```

ModifyComponent 처리

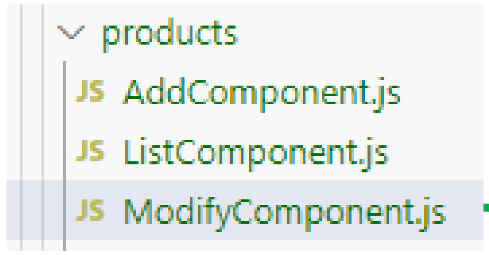
→ 새로운 이미지 파일의 추가와 수정



```
import { getOne, putOne } from "../../api/productsApi";

…생략

  const handleClickModify = () => {
    const files = uploadRef.current.files
    const formData = new FormData()
    for (let i = 0; i < files.length; i++) {
      formData.append("files", files[i]);
    }
    //other data
    formData.append("pname", product.pname)
    formData.append("pdesc", product.pdesc)
    formData.append("price", product.price)
    formData.append("delFlag", product.delFlag)
    for( let i = 0; i < product.uploadFileNames.length ; i++){
      formData.append("uploadFileNames", product.uploadFileNames[i])
    }
    putOne(pno, formData)
  }
```

ModifyComponent 처리

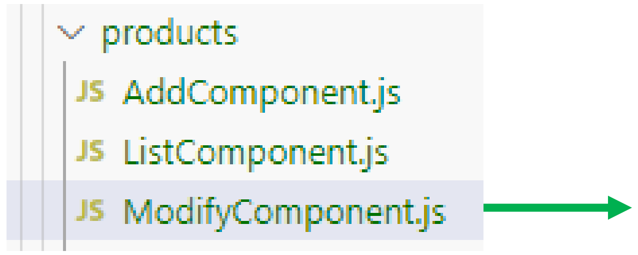
→ 새로운 이미지 파일의 추가와 수정



```
return ( …생략…
     <div className="flex justify-end p-4">
      <button type="button"
              className="rounded p-4 m-2 text-xl w-32 text-white bg-red-500">
        Delete
      </button>
      <button type="button" onClick={handleClickModify}
      className="inline-block rounded p-4 m-2 text-xl w-32  text-white bg-
orange-500">
        Modify </button>
      <button type="button"
      className="rounded p-4 m-2 text-xl w-32 text-white bg-blue-500">
        List
      </button>
    </div>

  </div>
  );
```

ModifyComponent 처리

→ 수정 작업 후 모달창

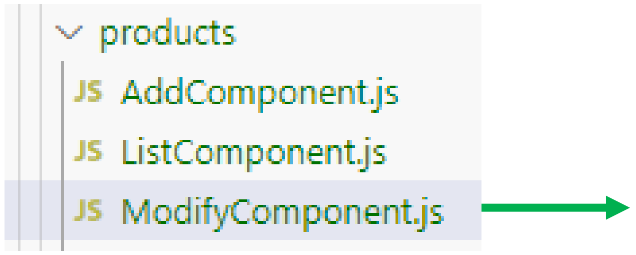


```
import { useEffect, useRef, useState } from "react";
import { getOne, putOne } from "../../api/productsApi";
import FetchingModal from "../common/FetchingModal";
import { API_SERVER_HOST } from "../../api/todoApi";
import useCustomMove from "../../hooks/useCustomMove";
import ResultModal from "../common/ResultModal";


  //결과 모달
  const [result, setResult] = useState(null)

  //이동용 함수
  const {moveToRead, moveToList} = useCustomMove()

  const [fetching, setFetching] = useState(false)
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

→ 수정 작업 후 모달창

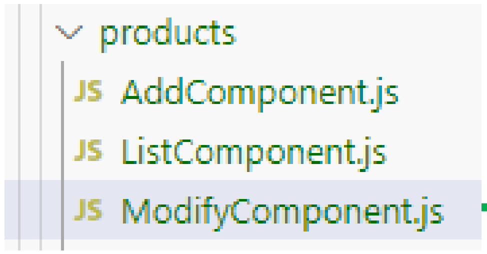


```
const handleClickModify = () => {
  const files = uploadRef.current.files
  const formData = new FormData()
  for (let i = 0; i < files.length; i++) {
    formData.append("files", files[i]);
  }
  //other data
  formData.append("pname", product.pname)
  formData.append("pdesc", product.pdesc)
  formData.append("price", product.price)
  formData.append("delFlag", product.delFlag)
  for( let i = 0; i < product.uploadFileNames.length ; i++){
    formData.append("uploadFileNames", product.uploadFileNames[i])
  }
  //fetching
  setFetching(true)
  putOne(pno, formData).then(data => { //수정 처리
    setResult('Modified')
    setFetching(false)
  })
}
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

→ 수정 작업 후 모달창

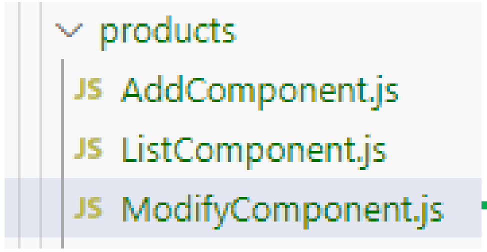


```
const closeModal = () => {
  if(result ==='Modified') {
    moveToRead(pno)  //조회 화면으로 이동
  }
  setResult(null)
}

return (
<div className = "border-2 border-sky-200 mt-10 m-2 p-4">
  {fetching? <FetchingModal/> :<></>}
  {result?
    <ResultModal
    title={`${result}`}
    content={'정상적으로 처리되었습니다.'}  //결과 모달창
    callbackFn={closeModal}/>
  :
  <></>
  }
…생략
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

→ 삭제 버튼의 동작 처리

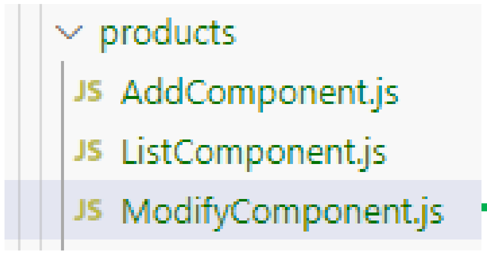


```
import { getOne, putOne, deleteOne } from "../../api/productsApi";

…생략…
  const handleClickDelete = () => {
    setFetching(true)
    deleteOne(pno).then(data => {
      setResult("Deleted")
      setFetching(false)
    })
  }

  const closeModal = () => {
    if(result ==='Modified') {
      moveToRead(pno)
    }else if(result === 'Deleted') {
      moveToList({page:1})
    }
    setResult(null)
  }
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

→ 삭제 버튼의 동작 처리

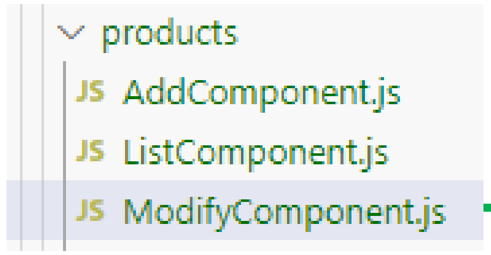


```
<div className="flex justify-end p-4">
  <button type="button"
  className="rounded p-4 m-2 text-xl w-32 text-white bg-red-500"
  onClick={handleClickDelete}
  >
    Delete
  </button>

  …생략


</div>
```

수정/삭제 페이지와 컴포넌트 처리

ModifyComponent 처리

→ 목록 화면 이동



```
<button type="button"
className="rounded p-4 m-2 text-xl w-32 text-white bg-blue-500"
onClick={moveToList}
>
  List
</button>
```

감사합니다.