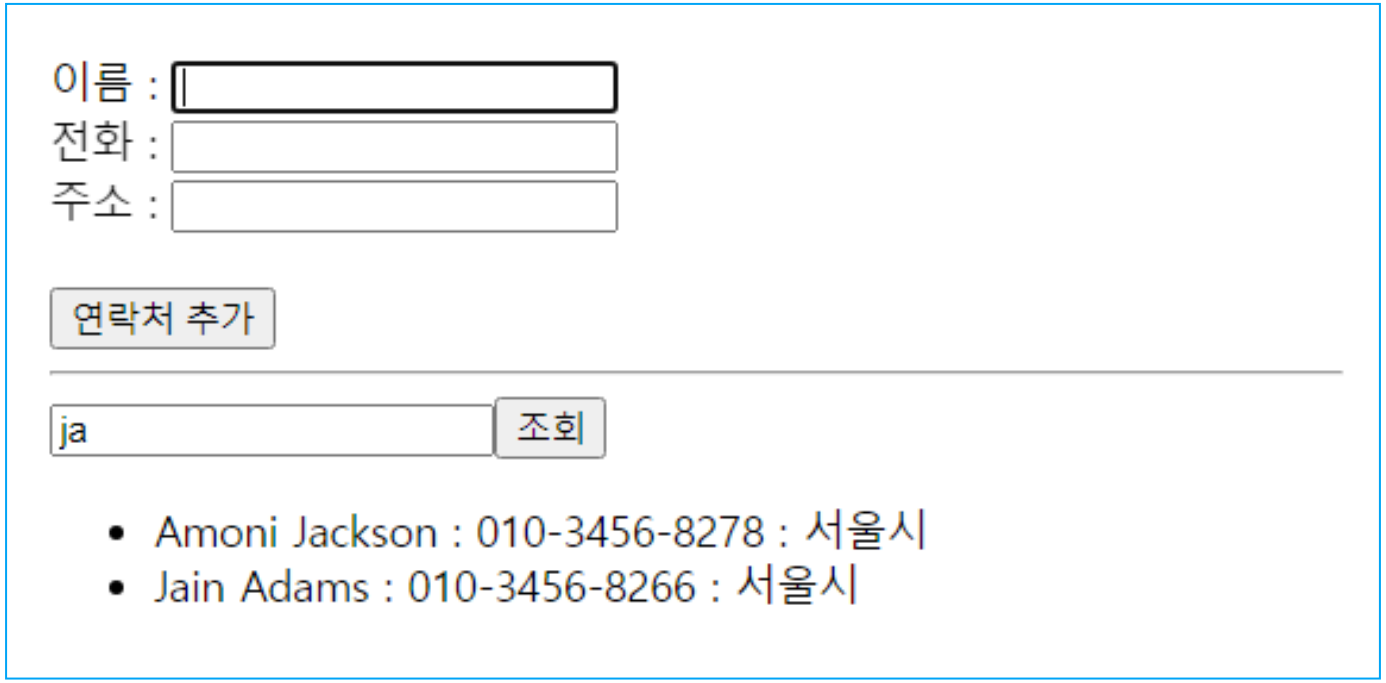
- axios + typescript

0. axios 예제 확인

❖es6, typescript 예제

- axios-app-es6
- axios-app-ts : 완성본
- contactsvc : 샘플 백엔드 API 예제

❖axios-app-es6를 기반으로 axios-app-ts와 같은 Typescript 코드로 리팩토링함



1. axios.get

❖사용 형태 : 많은 제네릭

```
axios.get<T, AxiosResponse<T, F>, D>(
    url: string,
    config?: AxiosRequestConfig<D> | undefined
): Promise<AxiosResponse<T, F>>
```

- T : 응답 데이터 형식
  - 사용하지 않으면 any 타입으로 처리
- F : 응답 데이터 형식 변환 타입
  - transformResponse 옵션 함수에 의해 변환된 응답데이터 형식 타입 지정
  - 사용하지 않으면 any 타입으로 처리
- D : 요청 파라미터 타입(AxiosRequestConfig)
  - 사용하지 않으면 any 타입으로 처리
- F, D는 사용하지 않는 경우가 대부분
  - F : ex) XML 응답데이터를 JSON으로 변
  - D : 요청 파라미터 형식은 문서화되어 정의되어 있으므로 any를 사용해도 무방하며, 대부분 Default Value 사용

1. axios.get

❖기존 코드 확인

- src/hooks/useFetch.js

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

const useFetch = (url, params) => {
  const [response, setResponse] = useState();
  const [error, setError] = useState();
  const [isLoading, setIsLoading] = useState(false);

  const fetchData = async () => {
    setResponse(undefined);
    setError(undefined);
    setIsLoading(true);
    try {
      const result = await axios.get(url, params);
      setResponse(result);
    } catch (err) {
      setError(err);
    } finally {
      setIsLoading(false);
    }
  };
```

```
  const requestFetch = () => {
    fetchData();
  };

  return { response, error, isLoading, requestFetch };
};

export { useFetch };
```

1. axios.get

❖기존 코드 확인(이어서)

- src/ContactList.jsx

```
import { useState } from "react";
import { useFetch } from "./hooks/useAxios";

const ContactList = () => {
  const [name, setName] = useState("ja");
  const { response, isLoading, error, requestFetch } = useFetch(
    `/contacts_long/search/${name}`,
    { timeout: 10000 }
  );

  const searchContacts = () => {
    if (!name || name.length < 2) {
      alert("조회를 위해 두글자 이상을 입력하세요");
      return;
    }
    requestFetch();
  };
```

```
  return (
    <div>
      <input type="text"value={name}
        onChange={(e) => setName(e.target.value)} />
      <button onClick={searchContacts}>조회</button>
      <br />
      <ul>
        {error ? (<h3>에러 발생 : {error.message}</h3>)
          : response ? (
          response.data.map((item) => {
            return (
              <li key={item.no}>
                {item.name} : {item.tel} : {item.address}{" "}
              </li>
            );
          })
        ) : ("")
        }
      </ul>
      {isLoading ? <h3>조회중</h3> : ""}
    </div>
  );
};
export default ContactList;
```

1. axios.get

❖typescript 버전의 코드

- src/hooks/useFetch.ts

```
import { useState } from "react";
import axios, { AxiosError, AxiosRequestConfig, AxiosResponse } from "axios";

const useFetch = <T>(url:string, params: AxiosRequestConfig) => {
  const [response, setResponse] = useState<AxiosResponse<T>>();
  const [error, setError] = useState<AxiosError>();
  const [isLoading, setIsLoading] = useState<boolean>(false);

  const fetchData = async () => {
    setResponse(undefined);
    setError(undefined);
    setIsLoading(true);
    try {
      const result: AxiosResponse<T> = await axios.get<T, AxiosResponse<T>>(url, params);
      setResponse(result);
    } catch (err) {
      setError(err as unknown as AxiosError);
    } finally {
      setIsLoading(false);
    }
  };
```

1. axios.get

❖typescript 버전의 코드

- src/hooks/useFetch.ts(이어서)

```
  const requestFetch = () => {
    fetchData();
  };

  return { response, error, isLoading, requestFetch };
};

export { useFetch };
```

1. axios.get

❖typescript 버전의 코드

- src/ContactList.tsx

```
import { useState } from "react";
import { useFetch } from "./hooks/useAxios";

type ContactItemType = {
  no: string; name: string; tel: string; address: string; photo?: string;
};

const ContactList = () => {
  const [name, setName] = useState<string>("ja");
  const { response, isLoading, error, requestFetch } = useFetch<ContactItemType[]>(
    `/contacts_long/search/${name}`,
    { timeout: 10000 }
  );

  const searchContacts = () => {
    if (!name || name.length < 2) {
      alert("조회를 위해 두글자 이상을 입력하세요");
      return;
    }
    requestFetch();
  };
```

1. axios.get

❖typescript 버전의 코드

- src/ContactList.tsx(이어서)

```
 return (
    <div>
      <input type="text" value={name} onChange={(e) => setName(e.target.value)} />
      <button onClick={searchContacts}>조회</button>
      <br />
      <ul>
      { error ? (<h3>에러 발생 : {error.message}</h3>) : response ?
         (response.data.map((item) => {
            return (
              <li key={item.no}>
                {item.name} : {item.tel} : {item.address}{" "}
              </li>
            );
          })) : ("")
       }
      </ul>
      {isLoading ? <h3>조회중</h3> : ""}
    </div>
  );
};

export default ContactList;
```

2. axios.post

❖사용 형태

```
axios.post<T, AxiosResponse<T, F>, any>(
  url: string,
  data?: P,
  config?: AxiosRequestConfig<D> | undefined
): Promise<AxiosResponse<T, F>>
```

- **T : 응답 데이터 타입**
  - 사용하지 않으면 any 타입으로 처리
- **P : Post 요청 데이터 타입**
- F : 응답 데이터 형식 변환 타입
  - transformResponse 옵션 함수에 의해 변환된 응답데이터 형식 타입 지정
  - 사용하지 않으면 any 타입으로 처리
- D : 요청 파라미터 타입(AxiosRequestConfig)
  - 사용하지 않으면 any 타입으로 처리

2. axios.post

❖기존 코드 확인

- src/hooks/usePost.js

```
……(생략)
const usePost = (url, params) => {
  const [response, setResponse] = useState();
  const [error, setError] = useState();
  const [isProcessing, setIsProcessing] = useState(false);
  const fetchData = async (data) => {
    setResponse(undefined);
    setError(undefined);
    setIsProcessing(true);
    try {
      const result = await axios.post(url, data, params);
      setResponse(result);
    } catch (err) {
      setError(err);
    } finally {
      setIsProcessing(false);
      setTimeout(()=>{
        setResponse(undefined);
        setError(undefined);
      }, 3000);
    }
  };
```

```
  const requestPost = (data) => {
    fetchData(data);
  };

  return { response, error, isProcessing, requestPost };
};

export { usePost };
```

2. axios.post

❖기존 코드 확인

- src/InputContact.jsx

```
import { useState } from "react";
import { usePost } from "./hooks/useAxios";

const InputContact = () => {
  const [name, setName] = useState("");
  const [tel, setTel] = useState("");
  const [address, setAddress] = useState("");

  const { response, isProcessing, error, requestPost } = usePost(`/contacts`, {
    timeout: 10000,
  });

  const addContact = () => {
    if (!name || name.length < 2 || !tel || !address) {
      alert("name은 두글자 이상, tel, address 를 반드시 입력해주세요.");
      return;
    }
    const data = { name, tel, address };
    requestPost(data);
  };
```

2. axios.post

❖기존 코드 확인

- src/InputContact.jsx(이어서)

```
const addContactHandler = () => {
   addContact();
   setName("");
   setTel("");
   setAddress("");
 };

 return (
   <div>
     이름 : <input type="text" value={name} onChange={(e) => setName(e.target.value)}
     />
     <br />
     전화 : <input type="text" value={tel} onChange={(e) => setTel(e.target.value)} />
     <br />
     주소 : <input type="text" value={address} onChange={(e) => setAddress(e.target.value)}
     />
     <br />
     <br />
     <button onClick={addContactHandler}>연락처 추가</button>
```

2. axios.post

❖기존 코드 확인

- src/InputContact.jsx(이어서)

```
      <br />
      {
       error ? (<h3>에러 발생 : {error.message}</h3>)
          : response ? (<h3>{response.data.message}</h3>) : ("")
      }
      {isProcessing ? <h3>처리중</h3> : ""}
    </div>
  );
};

export default InputContact;
```

2. axios.post

❖typescript 버전의 코드

- src/hooks/usePost.ts

```
import { useState } from "react";
import axios, { AxiosError, AxiosRequestConfig, AxiosResponse } from "axios";

//P : Request, T : Response
const usePost = <P,T>(url:string, params:AxiosRequestConfig) => {
  const [response, setResponse] = useState<AxiosResponse<T>>();
  const [error, setError] = useState<AxiosError>();
  const [isProcessing, setIsProcessing] = useState<boolean>(false);

  const fetchData = async (data:P) => {
    setResponse(undefined);
    setError(undefined);
    setIsProcessing(true);
```

2. axios.post

❖typescript 버전의 코드

- src/hooks/usePost.ts(이어서)

```
    try {
      const result = await axios.post<T,AxiosResponse<T>>(url, data, params);
      setResponse(result);
    } catch (err) {
      setError(err as unknown as AxiosError);
    } finally {
      setIsProcessing(false);
      setTimeout(()=>{
        setResponse(undefined);
        setError(undefined);
      }, 3000);
    }
  };

  const requestPost = (data:P) => {
    fetchData(data);
  };

  return { response, error, isProcessing, requestPost };
};

export { usePost };
```

2. axios.post

❖typescript 버전의 코드

- src/InputContacts.tsx

```
import { useState } from "react";
import { usePost } from "./hooks/useAxios";

type PostType = { name:string; tel:string; address:string; }
type ResponseType = { status: string; message: string; no?: string; }

const InputContact = () => {
  const [name, setName] = useState<string>("");
  const [tel, setTel] = useState<string>("");
  const [address, setAddress] = useState<string>("");

  const { response, isProcessing, error, requestPost } = usePost<PostType, ResponseType>(
    `/contacts`,
   {  timeout: 10000 }
  );
```

2. axios.post

❖typescript 버전의 코드

- src/InputContacts.tsx(이어서)

```
const addContact = () => {
  if (!name || name.length < 2 || !tel || !address) {
    alert("name은 두글자 이상, tel, address 를 반드시 입력해주세요.");
    return;
  }
  const data : PostType = { name, tel, address };
  requestPost(data);
};

const addContactHandler = () => {
  addContact();
  setName("");
  setTel("");
  setAddress("");
};

return (
  <div>
    이름 : <input type="text" value={name} onChange={(e) => setName(e.target.value)} /><br />
    전화 : <input type="text" value={tel} onChange={(e) => setTel(e.target.value)} /><br />
    주소 : <input type="text" value={address} onChange={(e) => setAddress(e.target.value)} /><br /><br />
    <button onClick={addContactHandler}>연락처 추가</button><br />
```

2. axios.post

❖typescript 버전의 코드

- src/InputContacts.tsx(이어서)

```
    {error ? (
      <h3>에러 발생 : {error.message}</h3>
    ) : response ? (
      <h3>{response.data.message}</h3>
    ) : (
      ""
    )}
    {isProcessing ? <h3>처리중</h3> : ""}
   </div>
  );
};

export default InputContact;
```