React Ajax

안 화 수

React Ajax

❖ React Ajax

1. Ajax는 Asynchronous Javascript And XML의 약자이다.
   말 그대로, 비동기 자바스크립트와 XML이라는 뜻이다.
2. 2005년 Jesse James Garett가 처음 만들어낸 말로, HTML, XHTML, CSS, JavaScript, DOM, XMLHttpRequest객체 등을 비롯해 기존의 여러 기술을 사용하는 새로운 접근법을 설명하는 용어이다.

React Ajax

- **Ajax 사용하는 방법**

1. 순수 자바스크립트 문법 : fetch()

2. jQuery 설치 : $.ajax()

3. axios 설치 : axios.get()

React Ajax

- **프로젝트 생성**

```
npx  create-react-app  ajaxtest01
```

- axios 설치

```
npm  install  axios
```

axios

❖ axios 사용법

1. axios는 ajax를 구현하기 위한 대표적인 라이브러리이다.
2. jQuery ajax 라이브러리와 비슷한 기능을 지원한다.
3. 브라우저, node.js를 위한 Promise API를 활용하는 HTTP 비동기 통신 라이브러리이다.
4. Promise객체를 리턴하고, 크로스 브라우징 기반으로 브라우저 호환성이 좋다.
5. 자동으로 JSON 데이터 형식으로 변환, 요청 취소 및 타임아웃 설정이 가능하다.

axios

❖ axios의 request method

- GET : axios.get(url[, config]) - 조회
- POST : axios.post(url, data[, config]) - 생성
- PUT : axios.put(url, data[, config]) - 수정
- DELET : axios.delete(url[, config]) - 삭제

axios

❖ axios 사용법

- 데이터 요청 성공

  - 형식 : axios.get(데이터 요청할 URL)
    .then(()=>{ 요청 성공시 실행할 코드 })

  - then() 안에 있는 코드가 실행된다.

- 데이터 요청 실패

  -형식 : axios.get(데이터 요청할 URL)
    .catch(()=>{ 요청 실패시 실행할 코드 })

  - catch() 안에 있는 코드가 실행된다.

axios

- **axios 사용법**

```
<button onClick={()=>{
        axios.get('https://codingapple1.github.io/shop/data2.json')
            .then((result)=>{
                console.log('success')
                console.log(result)
            }).catch(()=>{
                console.log('fail')
            })
}}>요청</button>
```

axios

❖ axios 사용법

https://codingapple1.github.io/shop/data2.json


```
[
    {
        "id": 3,
        "title": "Flowey",
        "content": "only 5 inches",
        "price": 120000
    },
    {
        "id": 4,
        "title": "Baby shoes",
        "content": "for less than 6",
        "price": 120000
    },
    {
        "id": 5,
        "title": "Red Herring",
        "content": "Born in France",
        "price": 120000
    }
]
```

axios 예제

- src/App.js

```
import Main1 from './Main1';
import Main2 from './Main2';
import Main3 from './Main3';
import Main4 from './Main4'
import Main5 from './Main5';
import Main6 from './Main6';

function App() {
        return (
                <Main1 />
//              <Main2 />
//              <Main3 />
//              <Main4 />
//              <Main5 />
//              <Main6 />
    );
}
export default App;
```

axios 예제1

- src/Main1.js (1/2)

```
import axios from 'axios';
import './App.css';

function Main1() {
  return (
    <div className="App">

      <h1>react ajax연습</h1>
```

axios 예제1

- src/Main1.js (2/2)

```
        <button onClick={()=>{
            axios.get('https://codingapple1.github.io/shop/data2.json')
                .then((result)=>{
                    console.log('success')
                    console.log(result)
                    console.log(result.data)
                    alert('ok');
                }).catch(()=>{
                    console.log('fail')
                })
            }}>요청</button>
        </div>
    );
}
export default Main1;
```

axios 예제2

- src/Main2.js (1/2)

```
import React, { useState, useEffect } from "react";
import axios from 'axios';

function Main2(){
    const [ review, setReview ] = useState([]);

    // 화면이 실행 될때 한 번만 실행하기
    useEffect(()=>{
                axios.get('https://codingapple1.github.io/shop/data2.json')
               .then((result)=>{
                            console.log(result)
                            setReview(result.data)
                }).catch((err)=>{
                            console.log(err)
                })
    }, [ ] )
```

axios 예제2

- src/Main2.js (2/2)

```
return(
        <div>
            <h1>react ajax연습</h1>
            { review.map((item, index) => (
                <div key={index}>
                        <h3>id : {item.id}</h3>
                        <p>title : {item.title}</p>
                        <p>content : {item.content}</p>
                        <p>price : {item.price}</p>
                </div>
            ))}
        </div>
    )
}
export default Main2;
```

axios 예제

- **axios 사용법**

  로컬에 저장된 item.json 파일 읽어 오기

  1. item.json파일을 public 디렉토리에 저장한다.
  2. Item.js 컴포넌트 파일을 생성한다.
  3. App.js 파일에서 Item.js 파일을 import한다.

JSON

- JSON (JavaScript Object Notation)

```
[
  {
    "id": "1",
    "name": "레몬",
    "price": " 3000",
    "description": "레몬에 포함되어 있는 쿠엔산은 피로회복에 좋다.  비타민C도 풍부하다."
  },
  {
    "id": "2",
    "name": "키위",
    "price": " 2000",
    "description": "비타민C가 매우 풍부하다.  다이에트와 미용에도 매우 좋다."
  },
  {
    "id": "3",
    "name": "블루베리",
    "price": " 5000",
    "description": "블루베리에 포함된 anthocyanin(안토시아닌)은 눈피로에 효과가 있다."
  },
  {
    "id": "4",
    "name": "체리",
    "price": " 5000",
    "description": "체리는 맛이 단 성분이 많고 피로회복에 잘 듣는다."
  },
  {
    "id": "5",
    "name": "메론",
    "price": " 5000",
    "description": "메론에는 비타민A와 칼륨이 많이 포함되어 있다."
  },
  {
    "id": "6",
    "name": "수박",
    "price": "15000",
    "description": "수분이 풍부한 과일이다."
  }
]
```

axios 예제

src/Item.js (1/2)

```
import React, { useState, useEffect } from "react";
import axios from 'axios';

function Item(){
    const [ review, setReview ] = useState([]);

    // 화면이 실행 될때 한 번만 실행하기
    useEffect(()=>{
        axios.get('http://localhost:3000/item.json')
        // axios.get('item.json')
        .then((result)=>{
            console.log(result)
            setReview(result.data)
        }).catch((err)=>{
            console.log(err)
        })
    },[])
```

axios 예제

- src/Item.js (2/2)

```
return(
        <div>
            <h1>react ajax연습</h1>

            {review.map((item, index) => (
                <div key={index}>
                    <h3>id: {item.id}</h3>
                    <p>name: {item.name}</p>
                    <p>price: {item.price}</p>
                    <p>description: {item.description}</p>
                </div>
            ))}
        </div>
    )
}
export default Item;
```

Spring boot

- Spring boot로 생성한 서버에 요청하기
- restapi01 프로젝트 생성
- cotroller - HomeController.java
  - SampleVo.java

Spring boot 예제3

- restapi01 프로젝트 : HomeController (1/2)

```
@RestController
public class HomeController {

        @RequestMapping("/sample")
        public SampleVo sample() {
                SampleVo sv = new SampleVo();
                sv.setMno(23);
                sv.setFirstName("홍");
                sv.setLastName("길동");
                return sv;
        }
```

Spring boot 예제4

- restapi01 프로젝트 : HomeController (2/2)

```
    @RequestMapping("/list")
    public List<SampleVo> list() {
                    List<SampleVo> list = new ArrayList<SampleVo>();
                    for (int i = 1; i <= 10; i++) {
                                SampleVo sv = new SampleVo();
                                sv.setMno(i);
                                sv.setFirstName("홍");
                                sv.setLastName("길동" + i);
                                list.add(sv);
                    }
            return list;
    }
}
```

Spring boot

- restapi01 프로젝트 : SampleVo

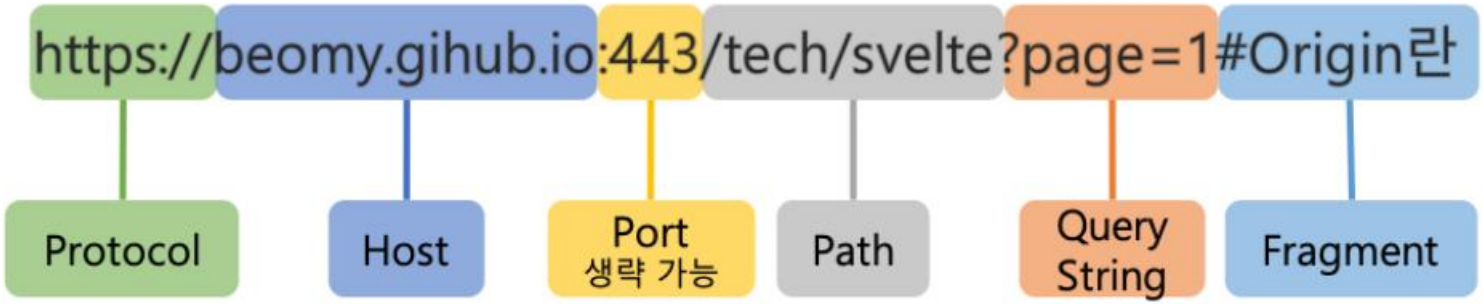
```
@Data
public class SampleVo {
        private Integer mno;
        private String firstName;
        private String lastName;
}
```

CORS

❖ CORS

교차 출처 리소스 공유 (Cross-origin Resource Sharing, Cors)는 추가 HTTP헤더를 사용하여, 한 출처에서 실행 중인 웹 애플리케이션이 다른 출처의 선택한 자원에 접근할 수 있는 권한을 부여하도록 브라우저에 알려주는 체제이다.

쉽게 말해서 도메인,프로토콜,포트번호가 하나라도 다를 경우에 출처가 다른 교차 출처(Cross Origin)라고 판단되며 브라우저에서는 보안 때문에 Cross-Origin HTTP 요청을 제한한다. 권한을 부여 받기 위한 Cross-Origin 요청은 서버에서 허가를 받아야 한다.



Protocol + Host + Port 3가지가 같으면 동일 출처(Origin)라고 한다.

CORS문제 해결

- React CORS문제 해결하기

1. Controller에 CrossOrigin 어노테이션을 사용하는 방법

2. WebConfig 클래스 생성해서 react의 3000번 포트허용하는 방법

CORS문제 해결

- React CORS문제 해결하기
  클래스나 메소드 위에 @CrossOrigin 어노테이션을 추가한다.

```
@RestController
@CrossOrigin("*")
public class HomeController {

        @RequestMapping("/sample")
        public SampleVo sample() {
                SampleVo sv = new SampleVo();
                sv.setMno(23);
                sv.setFirstName("홍");
                sv.setLastName("길동");
                return sv;
        }
}
```

CORS문제 해결

- React CORS문제 해결하기 : WebConfig 클래스 생성해서 react의 3000번 포트허용

```
package com.example.demo.config;

@Configuration
public class WebConfig implements WebMvcConfigurer{

	@Override
	public void addCorsMappings(CorsRegistry registry) {
		registry.addMapping("/**")	// react의 3000번 포트 허용
			.allowedOrigins("http://localhost:3000")
			.allowedMethods("GET", "POST", "PUT","DELETE")
			.allowedHeaders("*")
			.maxAge(3600);
	}
}
```

axios 예제3

- src/Main3.js (1/2)

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function Main3() {
    const [review, setReview] = useState([]);

    useEffect(() => {
        axios.get('http://localhost:80/sample')
            .then((response) => {
                console.log(response);
                console.log(response.data);
                setReview(response.data);
            })
            .catch((error) => {
                console.error('Error fetching data: ', error);
            });
    }, [] );  // 빈 배열을 전달하여 컴포넌트가 마운트될 때만 한 번 실행되도록 함
```

axios 예제3

- src/Main3.js (2/2)

```
    return (
        <div>
            <h1>react ajax 연습</h1>
            mno : {review.mno}<br/>
            First Name: {review.firstName}<br/>
            Last Name: {review.lastName}<br/>
        </div>
    );
}

export default Main3;
```

axios 예제4

- src/Main4.js (1/2)

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function Main4() {
    const [review, setReview] = useState([]);

    useEffect(() => {
        axios.get('http://localhost:80/list')
            .then((response) => {
                console.log(response);
                console.log(response.data);
                setReview(response.data);
            })
            .catch((error) => {
                console.error('Error fetching data: ', error);
            });
    }, [] );  // 빈 배열을 전달하여 컴포넌트가 마운트될 때만 한 번 실행되도록 함
```

axios 예제4

- src/Main4.js (2/2)

```
    return (
        <div>
            <h1>react ajax 연습</h1>

            { review.map((item, index) => (
                <div key={index}>
                    <h3>mno: {item.mno}</h3>
                    <p>First Name: {item.firstName}</p>
                    <p>Last Name: {item.lastName}</p>
                </div>
            ))}
        </div>
    );
}
export default Main4;
```

Spring boot 예제5

- restapi01 프로젝트 : HomeController - 추가

```
@RestController
@CrossOrigin("*")
public class HomeController {

    @RequestMapping("/register")
            public Integer register(@RequestBody SampleVo sv) {
                    System.out.println("mno:"+ sv.getMno());
                    System.out.println("firstName:"+ sv.getFirstName());
                    System.out.println("lastName:"+ sv.getLastName());

                    int result = 1;

            return result;
    }
}
```

axios 예제5

- src/Main5.js (1/2)

```
import axios from "axios";

function Main5(){
    const  loginaxios = (e) => {
            e.preventDefault();       // 창이 새로고침 되는 것을 막아준다.
            axios.post('http://localhost:80/register',{
                        mno : 10,
                        firstName : "hong",
                        lastName : "gildong",
            }).then((res)=>{
                        console.log(res.data);
                        alert(res.data);
                         if(res.status === 200){
                                    alert("회원가입 성공");
                        }
            }).catch((err)=>{
                        console.log(err);
             });
    };
```

axios 예제5

- src/Main5.js (2/2)

```
    return(
        <div>
            <button onClick={loginaxios}>회원가입</button>
        </div>
    )
}

export default Main5;
```

Spring boot 예제6

- restapi01 프로젝트 : HomeController - 추가

```
@RestController
@CrossOrigin("*")
public class HomeController {

    @RequestMapping("/register2")
            public Integer register2(@RequestBody SampleVo sv) {
                    System.out.println("mno:"+ sv.getMno());
                    System.out.println("firstName:"+ sv.getFirstName());
                    System.out.println("lastName:"+ sv.getLastName());

                    int result = 1;

            return result;
    }
}
```

axios 예제6

- src/Main6.js (1/3)

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

function Main6(){
        const [ mno, setMno ] = useState(0);
        const [ firstName, setFirstName ] = useState('');
        const [ lastName, setLastName ] = useState('');
```

axios 예제6

src/Main6.js (2/3)

```
const loginaxios = (e) => {
        e.preventDefault();              // 창이 새로고침되는 것을 막아준다.

        axios.post('http://localhost:80/register2',{
                mno : mno,
                firstName : firstName,
                lastName : lastName,
        }).then((res)=>{
                console.log(res.data);
                alert(res.data);
                if(res.status === 200){
                        alert("회원가입 성공");
                 }
        }).catch((err)=>{
                console.log(err);
         });
};     // loginaxios()  end
```

axios 예제6

- src/Main6.js (3/3)

```
    return(
        <div>
                mno : <input type="text" onChange={(e)=>{
                                setMno(e.target.value);
                        }} /> <br/><br/>
               firstName : <input type="text" onChange={(e)=>{
                                setFirstName(e.target.value);
                            }} /> <br/><br/>
               lastName :  <input type="text" onChange={(e)=>{
                                setLastName(e.target.value);
                            }} /> <br/><br/>
               <button onClick={loginaxios}>회원가입</button>
         </div>
     )
} // Main6 end
export default Main6;
```

axios

- **axios.get() : 데이터 조회**

  단순 데이터(페이지 요청, 지정된 요청) 요청을 수행할 경우

```
axios.get('https://test.com/user?userid=3')
.then( (response) => {
        // 성공 로직
}).catch( (error) => {
      // 에러 로직
}).finally( (e) => {
      // 항상 실행 로직
});
```

axios

- **axios.get() : 데이터 조회**

  파라미터 데이터를 포함시키는 경우 params 옵션을 이용

```
axios.get('https://test.com/user', {
  params : {
            userId : 3
          }
 }).then( (response) => {
        // 성공 로직
}).catch( (error) => {
        // 에러 로직
}).finally( (e) => {
        // 항상 실행 로직
});
```

axios

- **axios.post() : 데이터 생성**

post메서드에는 데이터를 Message Body에 포함시켜 보낸다.

```
axios.post("https://test.com/user", {
                    name : 'Fred',
                    userid : 100
  }).then( (response) => {

  }).catch( (error) => {

  })
```

axios

- **axios.put() : 데이터 수정**

put메서드는 서버에 있는 데이터베이스의 내용을 수정

```
axios.put("https://test.com/user", {
        name : 'Fred',
        userid : 100
  }).then( (response) => {

  }).catch( (error) => {

  })
```

axios

- **axios.delete() : 데이터 삭제**

  delete메서드는 서버에 있는 데이터베이스의 내용을 삭제

```
axios.delete('https://test.com/user?userid=3')
.then( (response) => {

}).catch( (error) => {

})
```

axios

- **axios.delete()**

  params가 많을 때는 두 번째 인자에 data를 추가한다.

```
axios.delete('https://test.com/user',{
  data : {
      userid : 3,
      username : "foo"
    }
}).then( (response) => {

}).catch( (error) => {

})
```