CORS

주차	29주차
스터디 일자	@2024/05/31

CORS 란?

> **교차 출처 리소스 공유(Cross-Origin Resource Sharing, CORS)**는 추가 HTTP 헤더를 사용하여, 한 출처에서 실행 중인 웹 애플리케이션이 다른 출처의 선택한 자원에 **접근할 수 있는 권한**을 부여하도록 브라우저에 알려주는 체제.

웹 애플리케이션은 **리소스가 자신의 출처(도메인, 프로토콜, 포트)와 다를 때** 교차 출처 HTTP 요청을 실행한다.

쉽게 말하면 CORS는 도메인이 다른 서버끼리 리소스를 주고 받는 정책

🤨 *그걸 왜 알아야 하죠?*

⇒ 일반적으로 웹 브라우저의 기본 정책이 *Same-Origin*으로, *origin*이 다른 서버와의 리소스 공유를 허용하고 있지 않기 때문에 이 CORS가 중요하다는 것!

- 예를 들어, 한 컴퓨터에서 React 서버(3000 포트)와 Springboot(8080 포트) 서버를 모두 띄워서 서로 리소스를 주고 받으려 한다면 **포트가 다르기 때문에** *origin*이 달라서 CORS 위반 문제가 발생하고, 개발자 도구 창에서는 CORS 위반 에러 메시지를 볼 수 있게 된다.

- 서버에서는 요청이 왔을 때 정상적으로 200번대 OK 상태 코드를 응답하고 리소스를 정상적으로 보내더라도, 응답을 받은 뒤 브라우저가 판단하기에 "같은 origin이 아니군" 이라고 판단해서 에러를 띄워버린다.

  결론적으로, **서버에서는 CORS 위반을 확인할 수 없다.**

CORS 작동 방식?

- **Preflight Request**
- **Simple Request**
- **Credentialed Request**

Preflight Request

기본적으로 브라우저는 HTTP 요청을 보낼 때, 사전에 *OPTIONS* 메서드를 통한 HTTP 요청을 보내서 요청을 보내기 **안전한지 확인해야 하는데,**

이를 미리 전송(*preflight*) 한다고 해서 **프리플라이트 요청(preflight request)**이라고 한다.

> **요청하려는 메서드 정보**는 `Access-Control-Request-Method` 헤더에
>
> **요청에 담길 헤더 정보**는 `Access-Control-Request-Headers` 헤더에 담아서 요청

```
OPTIONS /resources/post-here/ HTTP/1.1
Host: bar.other
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS
Accept: text/html,application/xhtml+xml,applicati
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Connection: keep-alive
Origin: http://foo.example
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Cont
```

> 그러면 서버는 응답으로 **어떤 것을 허용하는지**에 대한 정보를 담아서 돌려준다.

```
HTTP/1.1 204 No Content
Date: Mon, 01 Dec 2008 01:15:39 GMT
Server: Apache/2
Access-Control-Allow-Origin: https://foo.example
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER, Conter
Access-Control-Max-Age: 86400
Vary: Accept-Encoding, Origin
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
```

하지만 이런 Preflight Request 를 보내지 않는 경우도 있다.

먼저 브라우저를 사용하지 않으면 보내지 않는다.

⇒ Postman과 같은 기능을 사용하면 CORS 에러가 발생하지 않는다.

그리고 브라우저에서는 **Simple Request**일 경우 Preflight Request를 생략한다!

Simple Request

Simple Request 조건

- 본 요청 메서드가 `GET`, `HEAD`, `POST` 중 하나일 것

> - 클라이언트에서 자동으로 넣어주는 헤더와 Fetch 표준 정책에서 정의한 *CORS-safelisted request header* 라는 헤더 목록에 들어 있는 헤더 외에 **다른 헤더를 수동으로 넣어주지 않았을 것**
>   - Accept
>   - Accept-Language
>   - Content-Language
>   - Content-Type
>   - **Content-Type**의 경우 다음의 값들만 있을 것
>   - application/x-www-form-urlencoded
>   - multipart/form-data
>   - text/plain

위 조건을 모두 만족하는 경우에는 preflight 요청을 보내지 않는다.

⚠ 대부분의 HTTP API는 `text/xml` 이나 `application/json` 컨텐츠 타입을 가지도록 설계되기 때문에 사실상 이 조건들을 모두 만족시키는 상황을 만나기 쉽지 않아 Simple Request 경우는 흔히 보기 어렵다.

(= application/json은 Simple Request에 해당되지 않음 !)

Credentialed Request

Preflight Request, Simple Request 이외에 헤더에 인증과 관련된 정보를 담아서 보내는 Credentialed Request라는 경우도 있는데,

이 경우에는 credentials 옵션을 사용하며 CORS 정책 위반 여부를 검사하는 규칙에 **몇 가지 규칙이 더 들어가게 된다.**

CORS의 기본적인 방식이라기 보다는 다른 출처 간 통신에서 `좀 더 보안을 강화하고 싶을 때 사용하는 방법`으로,

헤더에 인증과 관련된 정보(쿠키, 토큰 등)를 담아서 보내는 식으로 사용한다.

서버에서 CORS 허용

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    public static final String ALLOWED_METHOD_NAMES = "GET,HEAD,

    @Override
    public void addCorsMappings(final CorsRegistry registry) {
        registry.addMapping("/**")
                    .allowedOrigins("*") // 허용할 origin 지정
                    .allowedOriginPatterns("*") // 보다 유연하게 o
                    .allowedMethods(ALLOWED_METHOD_NAMES.split("
                    .exposedHeaders(HttpHeaders.LOCATION); // 서
    }
}
```

```
@Bean
public WebMvcConfigurer corsConfigurer() {

    public static final String ALLOWED_METHOD_NAMES = "GET,POST,

    return new WebMvcConfigurer() {
```

```
        @Override
        public void addCorsMappings(CorsRegistry registry) {
            registry.addMapping("/**")
                    .allowedMethods(ALLOWED_METHOD_NAMES.split('
                    .exposedHeaders(HttpHeaders.LOCATION);
        }
    };
}
```

출처

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

https://velog.io/@effirin/CORS란-무엇인가

https://inpa.tistory.com/entry/WEB-📚-CORS-💯-정리-해결-방법-👏

https://velog.io/@ohzzi/CORS-허용-좀-해주세요