브라우저의 출처 정책 이해하기

SOP*(Same Origin Policy, 동일 출처 정책)*와 CORS*(Cross-Origin Resource Sharing, 교차 출처 리소스 공유)*

박연종

2026.01.28.

System Software Lab.

목차

백엔드 개발 끝! 프론트엔드 구현을 하려니…

CORS의 등장까지

1996년 3월, 브라우저에 JavaScript를 도입

Company Press Relations

NETSCAPE

NETSCAPE AND SUN ANNOUNCE JAVASCRIPT, THE OPEN, CROSS-PLATFORM OBJECT SCRIPTING LANGUAGE FOR ENTERPRISE NETWORKS AND THE INTERNET

28 INDUSTRY-LEADING COMPANIES TO ENDORSE JAVASCRIPT AS A COMPLEMENT TO JAVA FOR EASY ONLINE APPLICATION DEVELOPMENT

MOUNTAIN VIEW, Calif. (December 4, 1995) -- Netscape Communications Corporation (NASDAQ: NSCP) and Sun Microsystems, Inc. (NASDAQ:SUNW), today announced JavaScript, an open, cross-platform object scripting language for the creation and customization of applications on enterprise networks and the Internet. The JavaScript language complements Java, Sun's industry-leading object-oriented, cross-platform programming language. The initial version of JavaScript is available now as part of the beta version of Netscape Navigator 2.0, which is currently available for downloading from Netscape's web site.

JavaScript의 초기 버전은 Netscape Navigator 2.0 베타 버전부터 이용할 수 있다.

In addition, 28 industry-leading companies, including Ar
Attachmate Corporation, AT&T;, Borland International,
Corporation, Hewlett-Packard Company, Iconovex Corp
Inc., Intuit, Inc., Macromedia, Metrowerks, Inc., Novell, Inc., Oracle Corporation, Paper Software, Inc., Precept Software, Inc., RAD Technologies, Inc., The Santa Cruz Operation, Inc., Silicon Graphics, Inc., Spider Technologies, Sybase, Inc., Toshiba Corporation, Verity, Inc., and Vermeer Technologies, Inc., have endorsed JavaScript as an open standard object scripting language and intend to provide it in future products. The draft specification of JavaScript, as well as the final draft specification of Java, is planned for publishing and submission to appropriate standards bodies for industry review and comment this month.

JavaScript is an easy-to-use object scripting language designed for creating live online applications that link together objects and resources on both clients and servers. While Java is used by programmers to create new objects and applets, JavaScript is designed for use by HTML page authors and enterprise application developers to dynamically script the behavior of objects running on either the client or the server. JavaScript is analogous to Visual Basic in that it can be used by people with little or no programming experience to quickly construct complex applications. JavaScript's design represents the next generation of software designed specifically for the Internet and is:

- designed for creating network-centric applications
- complementary to and integrated with Java
- complementary to and integrated with HTML
- open and cross-platform.

출처 Web Design Museum

RELEASE NOTES

NETSCAPE NAVIGATOR 2.0 FOR WINDOWS

WHAT'S NEW?

Java Applets

Support for Java Applets:

- Java is currently supported on the 32 bit Navigator for both Windows 95 and Windows NT.
- There is a preference to disable Java applets located in the General panel of the security preferences. Java support is enabled by default.
- The version of Java supported by Netscape Navigator is compatible with the 1.0 version of the Java(tm) Developers Kit(JDK) from Sun Microsystems. Please see http://www.javasoft.com/JDK-1.0 for more info.
- Applets written for the ALPHA release of the HotJava browser are not compatible with this later version of Java.
- There is an index of Java applets at http://www.gamelan.com/
- For more info about Java, please see the Java Applets Page.

JavaScript

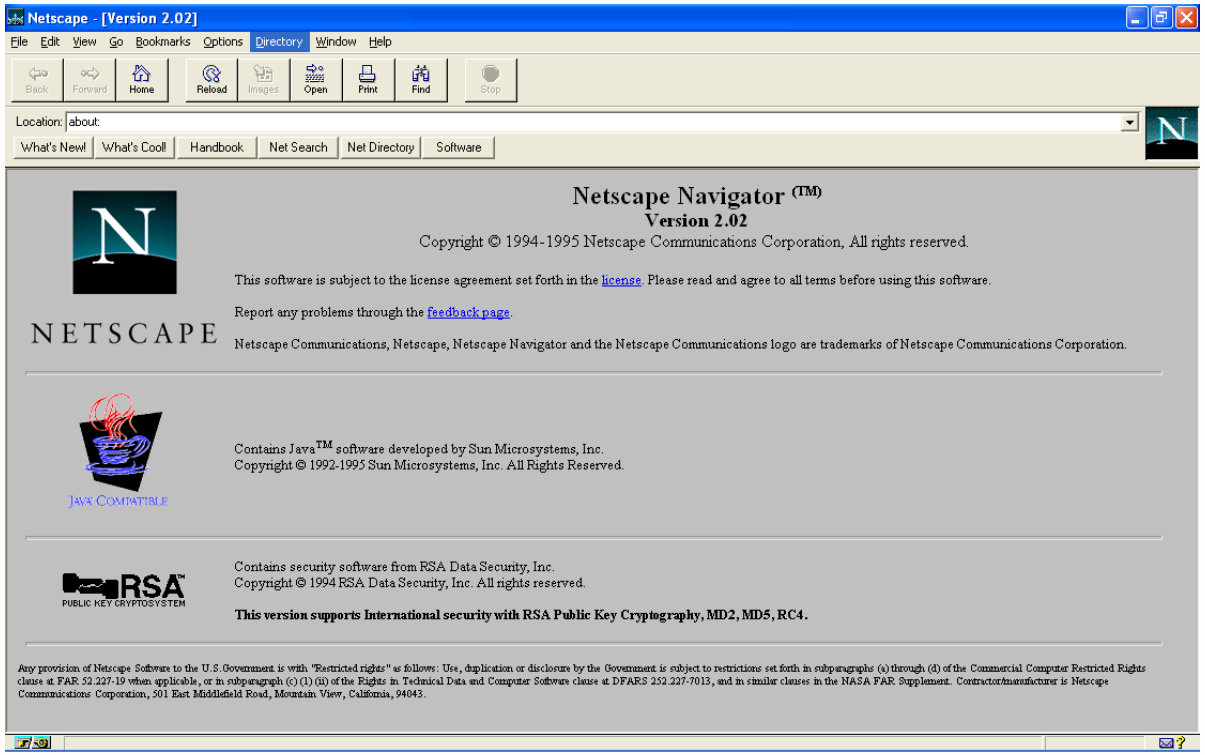
- **Built-in JavaScript:** Netscape Navigator now includes a built-in scripting language, called JavaScript. JavaScript development is based on the JAVA programming language, which extends and enhances the capability of HTML documents. JavaScript supports most of JAVA's expre
and s
no co

Netscape Navigator는 이제 JavaScript라고 부르는 스크립트 언어를 포함한다.

See the on-line JavaScript Documentation.

출처 Wayback Machine Netscape

1996년 5월, 다른 출처의 스크립트 실행을 방지하기 시작



Using data tainting for security

Navigator version 2.02 and later automatically prevents scripts on one server from accessing properties of documents on a diffe
fetching private information such a

Navigator 버전 2.02+에서는 다른 서버의 문서 요소에 접근하는 것을 자동으로 막는다.
이 제한은 스크립트가 민감한 정보를 가져오는 것을 방지한다.

출처 Wikipedia

1996년 5월 출시한 Netscape Navigator 버전 2.02 릴리즈 노트에 따르면,
다른 출처에 위치한 문서 속성으로의 접근을 자동으로 차단하여
다른 출처의 스크립트가 DOM *(Document Object Model)* 이나 민감한 정보에
접근하지 못하게 함.

1999년 3월, XMLHTTP 소개

1998년, Microsoft의 Exchange 2000 팀에 있던 Alex Hopmann은 Outlook Web Access를 더 좋은 사용자 경험을 만들고자 함

새로운 페이지를 띄워주거나, 페이지를 새로 로드하지 않고 서버와 통신하기를 원함

그래서 그는 XMLHTTP를 만들었고 이를 어떻게 배포할지 궁리함

> OWA는 아무 컴퓨터에서나 브라우저에서 접근할 수 있어야 함을 가치로 하였음
>
> 그래서 ActiveX로 배포하는 생각을 접고, 다른 대안을 고민

고민하던 끝에 Internet Explorer 5에 기본으로 탑재하게 되고, 1999년 IE 5가 출시

다른 브라우저에서는…

Mozilla는 2000년에, Apple Safari는 2004년에 `XMLHttpRequest`라는 이름으로 브라우저에 도입을 시작

> Netscape는 1997년 출시된 IE 3.0부터 Microsoft와의 경쟁에서 밀리기 시작하자
>
> 1998년 Navigator와 Gecko 엔진 소스 코드를 공개 및 오픈소스 개발 활동을 위해 Mozilla 팀을 만듦.

출처 Alex Hopmann, HamoniKR

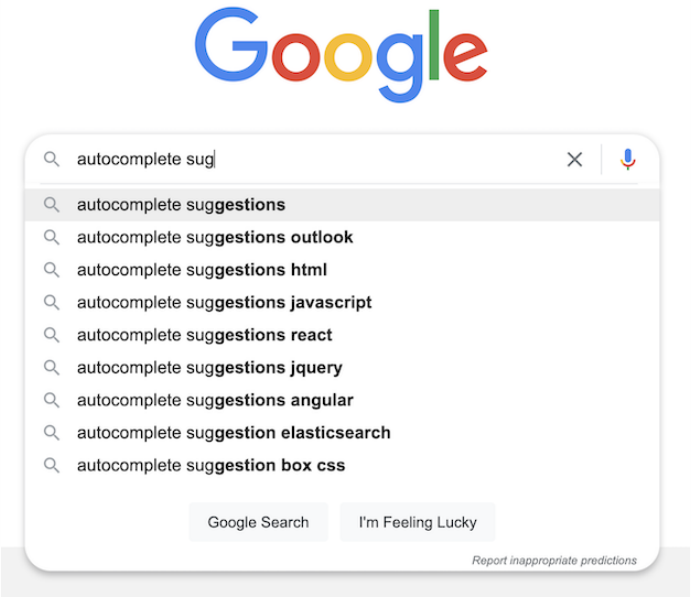
XMLHttpRequest의 인기

2004년 발표된 Gmail 등에도 XMLHTTP가 사용되었지만, 사람들이 소스 코드가 너무 복잡해 눈치채지 못했다고

2007년 Alex Hopmann이 자신의 회고록에서 설명

Google 검색 창에 검색어를 입력하면 검색어를 자동 추천하는 Google Suggest를 여러 사람이 소스 코드를 보며

어떻게 동작하는지 알게 되었다고 서술

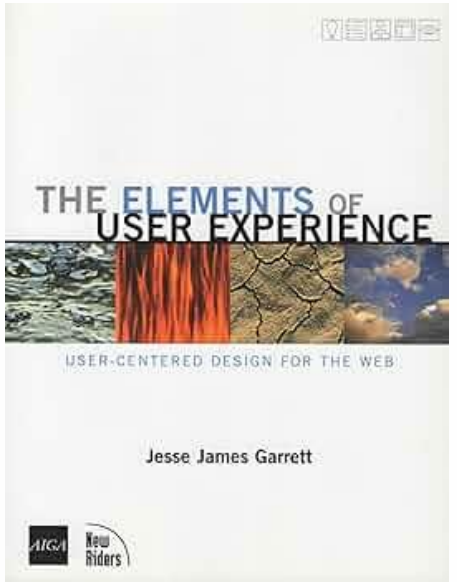


XMLHttpRequest의 인기

2004년 발표된 Gmail 등에도 XMLHTTP가 사용되었지만, 사람들이 소스 코드가 너무 복잡해 눈치채지 못했다고
2007년 Alex Hopmann이 자신의 회고록에서 설명

Google 검색 창에 검색어를 입력하면 검색어를 자동 추천하는 Google Suggest를 여러 사람이 소스 코드를 보며
어떻게 동작하는지 알게 되었다고 서술

2005년 2월 18일, Jesse James Garrett가 Google이 사용하는 비동기 통신을 자신의 논문에서
AJAX*(Asynchronous JavaScript And XML)*라고 명명한 이후 사람들이 이 용어를 사용하기 시작



Jesse James Garrett은 The Elements of User Experience의 저자로,
이 책으로 웹 디자인 커뮤니티에서 인기를 얻음.

XMLHttpRequest의 인기

2005년 2월 8일에 출시된 Google Maps는 AJAX로 구현돼 ActiveX를 사용하던 당시 한국의 지도 서비스를 사용하던 사람들에게 큰 충격을 주었음

홈 ›› Forums ›› 이슈 ›› 뉴스, 새소식

구글 맵~! 구글의 새 베타 서비스(비록 한국은 안되겠찌만..)

글쓴이: 정태영 / 작성시간: 화, 2005/02/15 - 10:42오후

http://maps.google.com/

역시 구글.. activeX 따위를 쓰지 않고도.. 크로스 브라우져에서..
이런 멋진 서비스가 가능하단 것을 보여주는군요 :)

google suggest 에서 한자한자 타이핑 할때마다.. 높은 순위의 검색단어들을.. 실시간 갱신해줄 때처럼.. xmlHTTP 를 사용하고 있습니다.. 요번에도 충분히 멋지군요 :)

역시 구글이란 말밖에 안나옵니다.. 꺄홋

Forums:
뉴스, 새소식

마치 종이지도를 보는듯한 느낌이 나도록 만들었군요..

글쓴이: s9204 / 작성시간: 화, 2005/02/15 - 11:14오후

마치 종이지도를 보는듯한 느낌이 나도록 만들었군요..

Good~

글쓴이: bigpooh / 작성시간: 수, 2005/02/16 - 1:23오전

정말 잘 만드네요. 감탄했습니다.

출처 KLDP


Web 2.0
Aggregators
Folksonomy
Wikis
Blogs
User Centered
Joy of Use
Participation
Six Degrees
Usability
Pagerank
XFN
Widgets
Recommendation
Social Software
FOAF
Browser
Simplicity
Sharing
Collaboration
Perpetual Beta
AJAX
Videocasting
Podcasting
Audio
IM
Video
Design
Convergence
CSS
Pay Per Click
UMTS
Mobility
Atom
XHTML
SVG
Ruby on Rails
VC
Trust
Affiliation
OpenAPIs
RSS
Semantic
Web Standards
SEO
Economy
OpenID
Remixability
REST
Standardization
The Long Tail
DataDriven
Accessibility
XML
Microformats
Syndication
Modularity
SOAP

JSONP

XMLHttpRequest를 사용해 비동기 통신을 만들 수 있었지만 다른 출처에서의 응답을 확인할 수 없었음

동일 출처 정책으로 인해 스크립트가 HTTP 통신으로 교차 출처의 데이터에 접근하여 읽기를 할 수 없었기 때문

오직 동일 출처에서만 XMLHttpRequest를 사용할 수 있었음

그러나 동일 출처가 아닌 다른 출처에서 응답을 받아와야 하는 경우가 빈번

이런 경우 사람들은 JSONP라는 우회 방법으로 다른 출처에서 응답을 가져옴

JSONP

서버 측

```
{ name: 'Foo', id: 1234, rank: 7 }
```

클라이언트 측

```
<script type="application/javascript" src="http://server.example.com/Users/1234"></script>
```

서버 측

```
parseResponse({ name: 'Foo', id: 1234, rank: 7 });
```

클라이언트 측

```
<script type="application/javascript"
src="http://server.example.com/Users/1234?callback=parseResponse"></script>
```

다른 출처의 JavaScript는 외부 라이브러리 실행을 위해 브라우저가 허용했음

사람들은 이를 이용해 동일 출처 정책을 우회해 다른 출처의 데이터를 가져왔지만 GET 메서드 등 제약이 존재

2009년, CORS의 등장

JSONP는 출처가 보내는 코드를 내 페이지 권한으로 실행해 안전한 데이터 접근이라 보기 어려움

또, GET 메서드로만 요청을 보낼 수 있고 헤더 등을 지정하기 힘듦

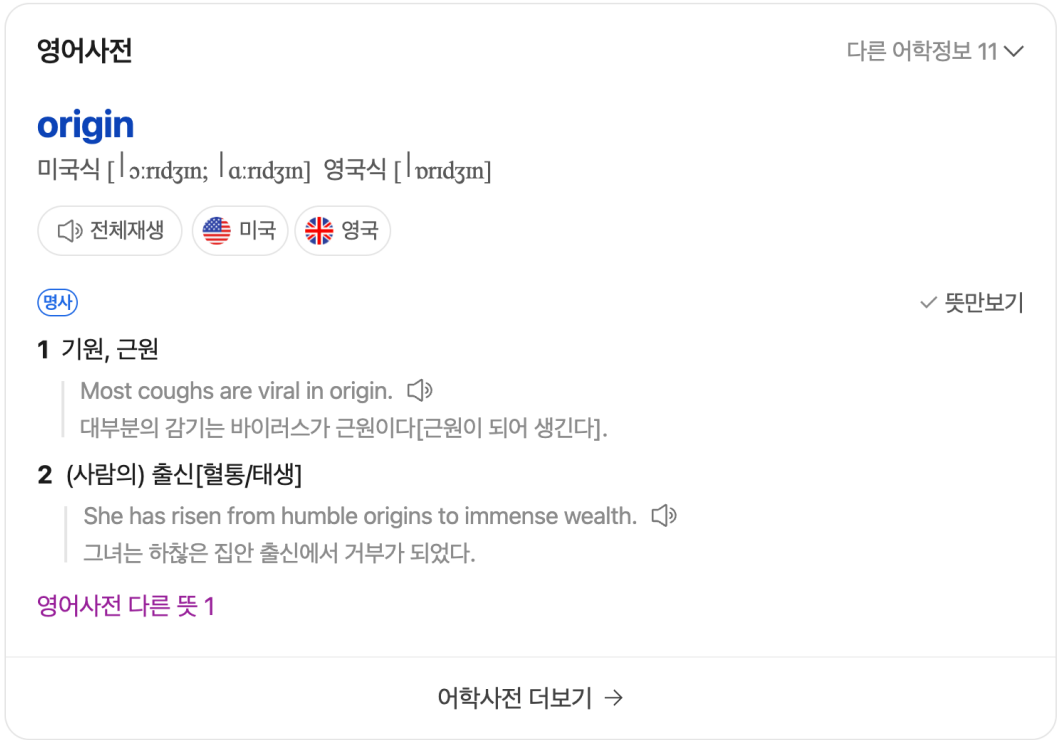
브라우저 개발 측은 서버가 허용한 경우에만 브라우저가 다른 출처의 응답을 읽을 수 있도록 CORS를 선보임

SOP *(Same Origin Policy, 동일 출처 정책)*

Origin

SOP *(Same Origin Policy, 동일 출처 정책)*
CORS *(Cross-Origin Resource Sharing, 교차 출처 리소스 공유)*

```
Access to fetch at 'http://localhost:3000/' (redirected from 'https://api.[redacted]') from origin '   test:1
http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
```



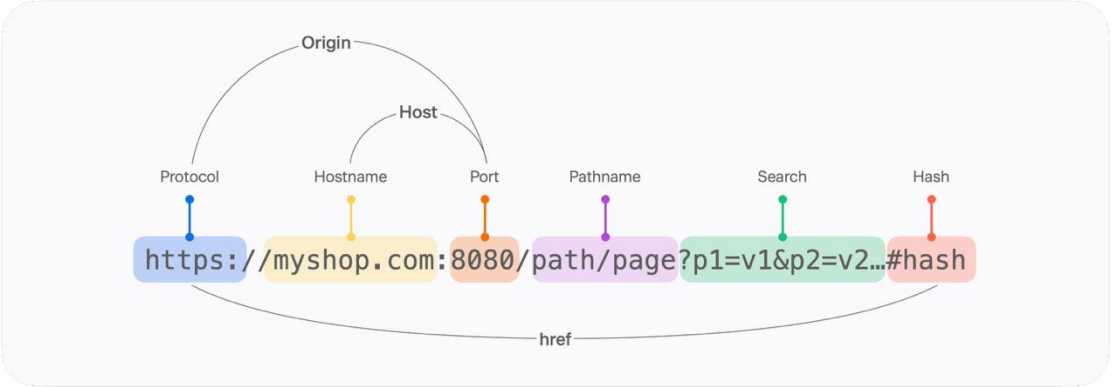
Origin

SOP *(Same Origin Policy, 동일 출처 정책)*
CORS *(Cross-Origin Resource Sharing, 교차 출처 리소스 공유)*

```
Access to fetch at 'http://localhost:3000/' (redirected from 'https://api.[redacted]') from origin '   test:1
http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
```

브라우저는 URI를 Origin으로 묶어 자원을 관리

Origin은 스킴(프로토콜)과 호스트 네임, 포트로 구성



왜 호스트 네임만이 아닐까? (RFC 6454)

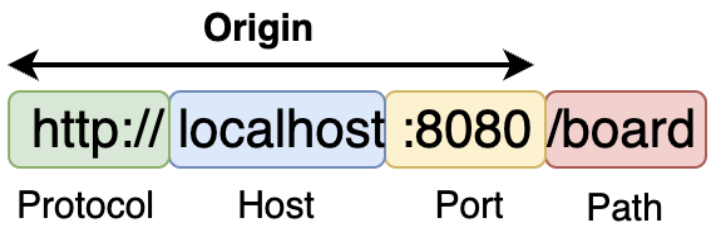
TLS로 받은 민감 데이터에 접근할 수 있는 위험이 있어 프로토콜을 포함해야 함

포트별로 서로 다른 서비스를 운용하는 경우가 흔해 서비스별 분리를 위해 포트를 포함해야 함

Same Origin Policy *(동일 출처 정책)*

어떤 출처에서 불러온 문서나 스크립트가 다른 출처에서 가져온 리소스와 상호 작용할 수 있는 방법을 제한하는 중요한 보안 메커니즘

프로토콜, 호스트 네임, 포트가 일치하면 동일한 출처라고 간주함



동일 출처 정책이 없었다면 다음의 끔찍한 시나리오가 성공적으로 발생할 수 있음

검찰청에서 수사를 받아야 한다며 링크에 접속하기를 유도

검찰청을 가장한 악성 사이트에 접속

해당 사이트는 실제 검찰청 사이트를 iframe으로 로드

사용자는 '나의 사건 조회'를 눌러 실제로 사건이 있는지 확인

이때 다른 출처의 요소인 iframe 내 DOM에 접근해 사용자의 개인정보를 탈취할 수 있음

CORS*(Cross-Origin Resource Sharing, 교차 출처 리소스 공유)*

Cross-Origin Resource Sharing *(교차 출처 리소스 공유)*

브라우저가 자신의 출처가 아닌 다른 어떤 출처로부터 자원을 로딩하는 것을 허용하도록 서버가 허가 해주는 HTTP 헤더 기반 메커니즘

서버가 실제 요청을 허가할 것인지 확인하기 위해 브라우저가 보내는 Preflight 메커니즘에 의존

요청이 GET, HEAD, POST(일부)일 때

→ 서버 상태를 변경하지 않는 요청이라고 판단하는 경우

브라우저가 다른 출처로 요청을 보낼 때 `Origin` 헤더를 추가해 요청을 전송
다른 출처는 `Access-Control-Allow-Origin` 헤더에 허용 출처를 담아 응답과 함께 전송

브라우저는 `Origin`과 `Access-Control-Allow-Origin` 헤더를 비교해 응답을 로딩할지 결정

Cross-Origin Resource Sharing *(교차 출처 리소스 공유)*

브라우저가 자신의 출처가 아닌 다른 어떤 출처로부터 자원을 로딩하는 것을 허용하도록 서버가 허가 해주는 HTTP 헤더 기반 메커니즘

서버가 실제 요청을 허가할 것인지 확인하기 위해 브라우저가 보내는 Preflight 메커니즘에 의존

그 외 메서드로 요청할 때

→ 서버 상태를 변경한다고 판단하는 경우

브라우저가 다른 출처로 요청을 보내기 전, OPTIONS 메서드와

`Origin`, `Access-Control-Request-Method`, `-Headers` 헤더를 추가해 요청을 전송

다른 출처는 `Access-Control-Allow-Origin`, `-Method`, `-Headers`,

`Access-Control-Max-Age` 헤더를 담아 응답

브라우저는 요청 정보와 일치하는지 확인해 실제 메서드의 요청을 전송

같은 출처로 요청할 때마다 사전 요청을 보내지 않기 위해

`Access-Control-Max-Age` 값(초)만큼 캐싱

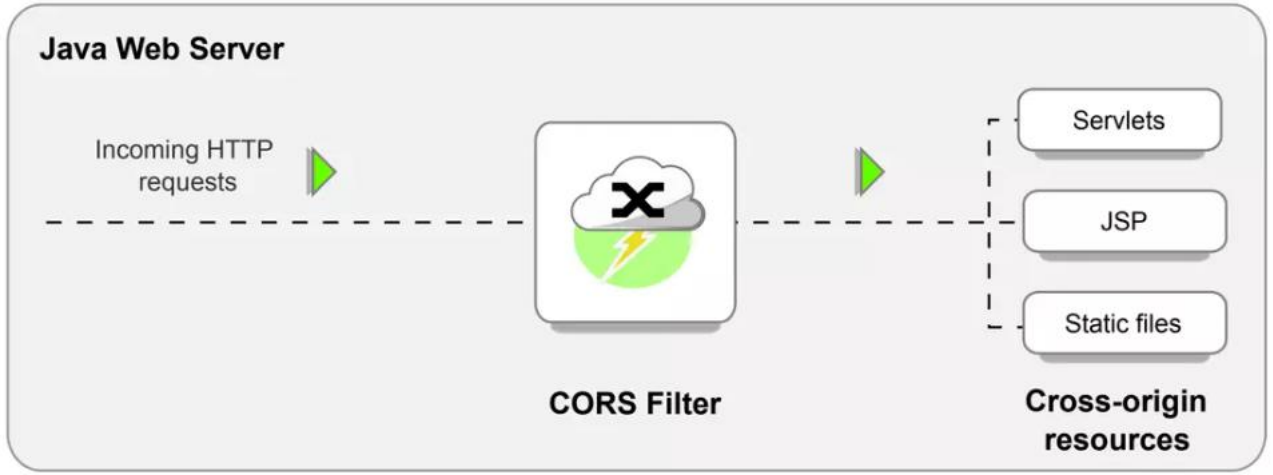
Cross-Origin Resource Sharing *(교차 출처 리소스 공유)*

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS
`Access-Control-Allow-Origin`	✓ 4 (2010)	✓ 12 (2015)	✓ 3.5 (2009)	✓ 12 (2012)	✓ 4 (2009)	✓ 18 (2012)	✓ 4 (2011)	✓ 12 (2012)	✓ 3.2 (2010)	✓ 1 (2013)	✓ 2 (2009)	✓ 3.2 (2010)

Spring Security를 사용해 CORS 설정

Spring Web에서는 `org.springframework.web.filter.CorsFilter`가 요청 및 응답을 가로채 CORS 헤더를 검증 및 추가함

개발자는 `CorsConfigurationSource`를 제공해 이 필터에 설정을 변경할 수 있음



Spring Security를 사용해 CORS 설정

Spring Web에서는 `org.springframework.web.filter.CorsFilter`가 요청 및 응답을 가로채 CORS 헤더를 검증 및 추가함

개발자는 `CorsConfigurationSource`를 제공해 이 필터에 설정을 변경할 수 있음

```java
@Configuration
@EnableWebSecurity
public class WebSecurityConfig {

    @Bean
    public SecurityFilterChain apiFilterChain(HttpSecurity http) throws Exception {
        http
            .securityMatcher("/api/**")
            .cors((cors) -> cors
                .configurationSource(apiConfigurationSource())
            )
            ...
        return http.build();
    }


    UrlBasedCorsConfigurationSource apiConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOrigins(Arrays.asList("https://frontend-url.example.com"));
        configuration.setAllowedMethods(Arrays.asList("GET","POST"));
        configuration.setAllowedHeaders(Arrays.asList("Authorization", "Cache-Control", "Content-Type"));
        configuration.setExposedHeaders(Arrays.asList("Authorization","Location"));

        configuration.setMaxAge(Arrays.asList(300L));

        configuration.setAllowCredentials(true));

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);

        return source;
    }
}
```

정리

SOP (Same Origin Policy)

JavaScript가 서로 다른 출처의 응답에 접근하는 것을 제한하는 브라우저 기본 정책

CORS (Cross-Origin Resource Sharing)

허용된 출처에 한해서만 서로 다른 출처임에도 JavaScript가 응답에 접근할 수 있도록 하는 브라우저 정책

Java 17의 sealed 클래스

박연종

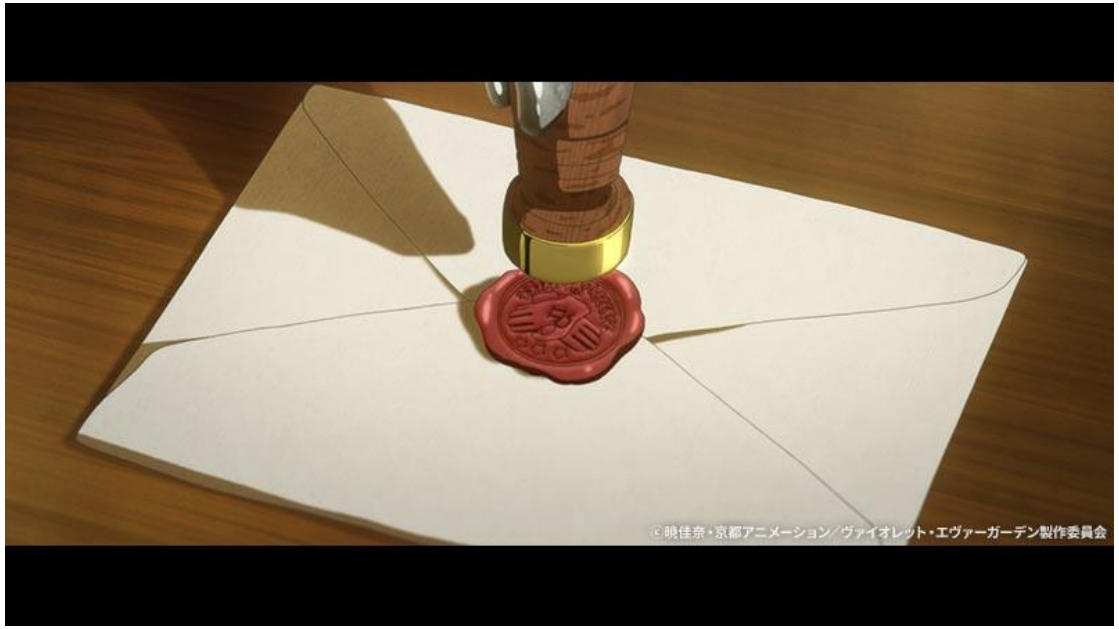
2026.01.28.

목차

sealed? permits?

```
public sealed abstract class ResponseBody<T> permits SuccessResponseBody, FailedResponseBody {
    @JsonInclude(JsonInclude.Include.NON_NULL)
    private String code;
}
```

seal



영어사전 다른 어학정보 10

seal

미국・영국 [si:l]

전체재생 미국 영국

동사 뜻만보기

1 (봉투 등을) 봉[봉인]하다

Make sure you've signed the cheque before sealing the envelope.
봉투를 봉하기 전에 수표에 서명을 했는지 확인하세요.

2 밀봉[밀폐]하다

The organs are kept in sealed plastic bags.
그 장기들은 밀봉된 비닐봉지 속에 보관된다.

명사

1 (특정 개인·기관을 나타내는) 직인[도장/인장]

The letter bore the president's **seal**.
그 편지에는 회장의 직인이 찍혀 있었다.

영어사전 다른 뜻 4

어학사전 더보기 →

permit

영어사전 다른 어학정보 5

permit

동사 [pə|mɪt; 美 pər|mɪt]

전체재생 미국 영국

동사 뜻만보기

1 허용[허락]하다

> Radios are not permitted in the library.
> 도서관 안에서는 라디오가 허용되지 않는다.

2 (어떤 일을) 가능하게 하다[허락하다]

> I'll come tomorrow, weather permitting.
> 내일 날씨가 괜찮으면 제가 오겠어요.

명사

1 (특히 한정된 기간 동안 유효한) 허가증

> a fishing/residence/parking, etc. permit
> 낚시/거주/주차 허가증 등

영어사전 다른 뜻 1

활용형

3인칭 단수 현재 permits
과거형 permitted
과거 분사 permitted
현재 분사 permitting
명사 permitter, permission
형용사 permissive

sealed? permits?

```
public sealed abstract class ResponseBody<T> permits SuccessResponseBody, FailedResponseBody {
    @JsonInclude(JsonInclude.Include.NON_NULL)
    private String code;
}
```

sealed classes

sealed로 선언된 클래스/인터페이스는 상속(extends)/구현(implements)할 수 있는 타입을 제한

누가 상속할 수 있는지를 언어 차원에서 명시하는 기능

```
public sealed class Shape permits Circle, Rectangle, Square {
    // 공통 속성 및 메서드
}

public final class Circle extends Shape {
    private final double radius;
    // Circle 구현
}

public final class Rectangle extends Shape {
    private final double width;
    private final double height;
    // Rectangle 구현
}

public final class Square extends Shape {
    private final double side;
    // Square 구현
}
```

sealed classes

permit한 클래스가 sealed 클래스를 상속 또는 구현하지 않으면 컴파일러가 오류를 발생시킴

```
public sealed class Shape permits Circle, Rectangle, Square { // 오류!
    // 공통 속성 및 메서드
}

public class Circle {
    private final double radius;
    // Circle 구현
}
```

permit한 클래스 외 클래스가 sealed 클래스를 상속 또는 구현하면 컴파일러가 오류를 발생시킴

```
public sealed class Shape permits Circle, Rectangle, Square {
    // 공통 속성 및 메서드
}

public final class Triangle extends Shape { // 오류!

}
```

sealed classes

sealed 클래스를 상속 또는 구현한 permit한 클래스가 서브 클래스를 만들 수 있도록 허용할 수 있음

```
public sealed class Shape permits Circle, Polygon {

}

public non-sealed class Polygon extends Shape {

}

public final class Rectangle extends Polygon {

}

public final class Square extends Polygon {

}
```

왜 sealed 클래스가 필요할까?

코드 재사용을 위한 상속만 있지 않고, 타입 계층을 구조화하기 위해 사용

라이브러리/프레임워크 개발자가 도형 종류, 대출 종류, 결제 방식 등의 도메인을 다룰 때

알려진 몇 가지 타입만 존재하기를 원함

```
public abstract class PaymentMethod { }

public final class Card extends PaymentMethod { }

public final class BankTransfer extends PaymentMethod { }

public final class PaymentProcessor {
    public static void pay(PaymentMethod m, long amountWon) {
        if (m instanceof Card c) {
            chargeCard(c, amountWon);
        } else if (m instanceof BankTransfer t) {
            transfer(t, amountWon);
        } else {
            throw new IllegalArgumentException("Unsupported payment method: " + m.getClass());
        }
    }
}
```

```
public final class Crypto extends PaymentMethod { }

public class App {
    public static void main(String[] args) {
        PaymentMethod m = new Crypto();
        PaymentProcessor.pay(m, 10_000); // 오류!
    }
}
```

왜 sealed 클래스가 필요할까?

코드 재사용을 위한 상속만 있지 않고, 타입 계층을 구조화하기 위해 사용

개발자들은 다음 두 가지 방법을 택해 외부에서 상속을 제한할 수 있음

`final` 키워드 사용

상속 자체를 할 수 없도록 하여 서브 클래스를 만들 수 없도록 함

```
public final class PaymentMethod { }

public final class Card extends PaymentMethod { } // 불가

public final class BankTransfer extends PaymentMethod { } // 불가
```

왜 sealed 클래스가 필요할까?

코드 재사용을 위한 상속만 있지 않고, 타입 계층을 구조화하기 위해 사용

개발자들은 다음 두 가지 방법을 택해 외부에서 상속을 제한할 수 있음

package-private

같은 패키지에서만 상속 가능하게 하여 외부에서 서브 클래스를 만들 수 없도록 함

이 방법은 외부에서 슈퍼 클래스를 볼 수 없게 하여 분기 등의 처리가 불가능함

그렇다고 해서 슈퍼 클래스를 public으로 하면 누구나 상속 가능한 문제가 발생

```
abstract class PaymentMethod { }

public final class Card extends PaymentMethod { }

public final class BankTransfer extends PaymentMethod { }
```

```
public class App {
    public static void main(String[] args) {
        PaymentMethod method = new Card();
    }
}
```

sealed 클래스를 사용해야 하는 이유

상속을 할 수 있는 타입을 제한해 설계 의도를 명확화

```
public abstract class PaymentMethod { }
```

각 사용자가 새로운 결제 수단을 추가하라는 뜻인가?

라이브러리/프레임워크에서 지원하는 결제 수단이 한정되어 있다는 뜻인가?

```
public sealed abstract class PaymentMethod permits Card, BankTransfer { }
```

라이브러리/프레임워크에서 지원하는 결제 수단이 한정되어 있네.

안전한 타입 분기

```
public static String route(PaymentMethod method) {
    return switch (method.type()) {
        case CARD -> "CARD_GATEWAY";
        case BANK_TRANSFER -> "BANK_API";
        default -> "UNKNOWN";
    };
}
```

잘못 동작할 수 있음

sealed 클래스를 사용해야 하는 이유

상속을 할 수 있는 타입을 제한해 설계 의도를 명확화

```
public abstract class PaymentMethod { }
```

각 사용자가 새로운 결제 수단을 추가하라는 뜻인가?

라이브러리/프레임워크에서 지원하는 결제 수단이 한정되어 있다는 뜻인가?

```
public sealed abstract class PaymentMethod permits Card, BankTransfer { }
```

라이브러리/프레임워크에서 지원하는 결제 수단이 한정되어 있네.

안전한 타입 분기

```
public static String route(PaymentMethod method) {
    return switch (method.type()) {
        case CARD -> "CARD_GATEWAY";
        case BANK_TRANSFER -> "BANK_API";

    };
}
```

모든 경우에 대해 처리하는지 컴파일러가 확인할 수 있음

(여전히 default는 사용할 수 있으나 컴파일러 확인을 포기하는 것임)

의도 살펴보기

```
public sealed  abstract class ResponseBody<T> permits SuccessResponseBody, FailedResponseBody {
    @JsonInclude(JsonInclude.Include.NON_NULL)
    private String code;
}
```

들어 주셔서 감사합니다

박연종

E-mail	park@duck.com
LinkedIn	yeonjong-park
Instagram	yeonjong.park
GitHub	patulus

2026.01.28.