



쿠키, 세션, (토큰), 캐시, (CDN), 이벤트 위임

웹 환경에서는 이처럼 반복적으로 사용되는 데이터나 정보를 종류와 특성에 맞게 저장하고 재활용하기 위해 여러 방식을 사용합니다.

HTTP 프로토콜의 특성이자 약점을 보완하기 위해서 쿠키 또는 세션을 사용합니다. 기본적으로 HTTP 프로토콜 환경은 "connectionless, stateless"한 특성을 가지기 때문에 서버는 클라이언트가 누구인지 매번 확인해야 합니다. 이 특성을 보완하기 위해서 쿠키와 세션을 사용하게 됩니다.

connectionless

클라이언트가 요청을 한 후 응답을 받으면 그 연결을 끊어 버리는 특징

- HTTP는 먼저 클라이언트가 request를 서버에 보내면, 서버는 클라이언트에게 요청에 맞는 response를 보내고 접속을 끊는 특성이 있다.
- 헤더에 keep-alive라는 값을 줘서 커넥션을 재활용하는데 HTTP1.1에서는 이것이 디폴트다.
- HTTP가 tcp위에서 구현되었기 때문에 (tcp는 연결지향,udp는 비연결지향) 네트워크 관점에서 keep-alive는 옵션으로 connectionless의 연결비용을 줄이는 것을 장점으로 비연결지향이라 한다.

stateless

- 통신이 끝나면 상태를 유지하지 않는 특징

쿠키와 세션은 위의 두 가지 특징을 해결하기 위해 사용합니다.

쿠키와 세션을 사용했을 경우, 한 번 로그인을 하면 **어떠한 방식에** 의해서 그 사용자에게 대한 인증을 유지하게 됩니다.

브라우저에 저장되는 정보 : 쿠키

쿠키 때문에 쇼핑 사이트에 로그인하지 않아도 장바구니에 물건을 담아두거나 검색 기록에서 이전에 입력했던 검색어들을 찾아볼 수 있습니다. 나의 웹 서핑 내역이 마케팅과 광고에 활용되는 것도 쿠키를 통해 이뤄지는 일 이죠.

쿠키는 크롬이나 사파리 같은 브라우저에 저장되는 작은 텍스트 조각입니다. 브라우저는 사용자의 컴퓨터에 설치된 소프트웨어이므로 쿠키는 사용자가 갖고 있는 정보라고 할 수 있죠.

사용자는 브라우저의 설정 화면이나 개발자 도구에서 쿠키를 **확인**하고 **수정**, **삭제**할 수 있습니다. 다만 쿠키는 당사자뿐만 아니라 제 3자가 조회하는 것도 가능하기 때문에 개인 정보를 담은 내용이나 보안상 민감한 정보를 저장하는 데에는 적합하지 않습니다.

따라서 혹여 남에게 탈취되거나 사용자에게 의해 조작되어도 크게 문제되지 않을 정보를 브라우저에 저장함으로써 웹사이트 이용을 편리하게 해 주는 것이 쿠키입니다. 예를 들면 자주 보는 웹툰 목록이나 웹 페이지의 다크 모드 설정 여부 등과 같은 간단한 정보 말이죠.

- 쿠키는 클라이언트(브라우저) 로컬에 저장되는 키와 값이 들어있는 작은 데이터 파일입니다.
- 사용자 인증이 유효한 시간을 명시할 수 있으며, 유효 시간이 정해지면 브라우저가 종료되어도 인증이 유지된다는 특징이 있습니다.
- 쿠키는 클라이언트의 상태 정보를 로컬에 저장했다가 참조합니다.
- 클라이언트에 300개까지 쿠키저장 가능, 하나의 도메인당 20개의 값만 가질 수 있음, 하나의 쿠키값은 4KB까지 저장합니다.
- Response Header에 Set-Cookie 속성을 사용하면 클라이언트에 쿠키를 만들 수 있습니다.

- 쿠키는 사용자가 따로 요청하지 않아도 브라우저가 Request시에 Request Header를 넣어서 자동으로 서버에 전송합니다.

쿠키의 구성 요소

- 이름 : 각각의 쿠키를 구별하는 데 사용되는 이름
- 값 : 쿠키의 이름과 관련된 값
- 유효시간 : 쿠키의 유지시간
- 도메인 : 쿠키를 전송할 도메인
- 경로 : 쿠키를 전송할 요청 경로

쿠키의 동작 방식

1. 클라이언트가 페이지를 요청
2. 서버에서 쿠키를 생성
3. HTTP 헤더에 쿠키를 포함 시켜 응답
4. 브라우저가 종료되어도 쿠키 만료 기간이 있다면 클라이언트에서 보관하고 있음
5. 같은 요청을 할 경우 HTTP 헤더에 쿠키를 함께 보냄
6. 서버에서 쿠키를 읽어 이전 상태 정보를 변경 할 필요가 있을 때 쿠키를 업데이트 하여 변경된 쿠키를 HTTP 헤더에 포함시켜 응답

쿠키의 사용 예

- 방문 사이트에서 로그인 시, "아이디와 비밀번호를 저장하시겠습니까?"
- 쇼핑몰의 장바구니 기능
- 자동로그인, 팝업에서 "오늘 더 이상 이 창을 보지 않음" 체크, 쇼핑몰의 장바구니

쿠키와 document.cookie



<https://ko.javascript.info/cookie>



JAVASCRIPT.INFO
The Modern JavaScript Tutorial

서버가 나를 알아보는 방법: 세션

웹사이트에 아이디와 비밀번호를 입력해서 로그인하면 해당 사이트의 회원에게만 허용된 기능들을 사용할 수 있습니다. 마이페이지를 클릭해서 내 정보를 볼 수도 있고, 회원 전용 게시판의 글쓰기 버튼을 클릭해서 질문을 남기거나 리뷰를 쓸 수도 있죠.

문제는 이와 같은 클릭 하나하나는 매번 서버에게 새로 보내는 익명 편지와도 같아서 사이트에 로그인을 하는 등의 이전 행동들과 연결되어 있지 않다는 것입니다. 다시 말해 **서버는 아이디와 비밀번호를 입력해 로그인에 성공한 사용자와 로그인한 다음 마이페이지 버튼을 누른 사용자가 동일 인물임을 알지 못한다**는 것입니다. (stateless)

그렇기 때문에 사용자가 사이트에 로그인한 상태라는 점을 서버에 인증하지 못하면 클릭을 할 때마다 반복해서 아이디와 비밀번호를 서버에 제공해야 합니다. (connectless)

이런 번거로움을 해결하기 위해 사용하는 것이 바로 **세션**입니다.

사용자가 사이트에 한 번 로그인하면 유효기간이 끝날 때까지 더 이상 아이디와 비밀번호를 입력하지 않아도 되도록 사용자가 이미 서버로부터 인증받았음을 증명해 주는 세션이라는 증거가 필요합니다.

사용자가 서버에 올바른 아이디와 비밀번호로 로그인에 성공하면 서버는 **세션 아이디**라는 데이터를 만듭니다. 보통은 '2sd98dbawix4'와 같은 식으로 **알파벳과 숫자가 혼합된 형식**을 갖고 있습니다. 서버는 **영화관에서 티켓을 보관용 부분만 찢어 건네주듯 세션 아이디를 사용자에게 전달**하고, **메모리에 아이디 사본을 어떤 사용자의 것인지 적어서 보관**합니다.

사용자는 서버로부터 받은 **세션 아이디**를 **쿠키로 저장**한 다음 앞으로의 **모든 요청에 함께 전달**합니다. 친구 목록을 볼 때, 댓글을 작성하거나 삭제할 때, 구매한 상품 내역을 볼 때도 서버에게 세션 아이디를 적은 편지를 보냅니다.

서버는 사용자에게서 친구 목록을 보겠다는 요청을 받으면 그 편지에 세션 아이디가 적혀있는지를 확인합니다. 아이디가 있다면 서버가 보관하고 있는 세션 아이디 중에 동일한 정보가 있는지 찾아보고 그것이 누구의 계정인지도 알아내죠. 그렇게 편지를 보낸 사람이 누구인지 파악한 다음 해당 사용자의 친구 목록을 보내주는 것입니다.

- 세션은 쿠키를 기반하고 있지만, 사용자 정보 파일을 브라우저에 저장하는 쿠키와 달리 세션은 서버 측에서 관리합니다.
- 서버에서는 클라이언트를 구분하기 위해 세션 ID를 부여하며 웹 브라우저가 서버에 접속해서 브라우저를 종료할 때까지 인증상태를 유지합니다.
- 물론 접속 시간에 제한을 두어 일정 시간 응답이 없다면 정보가 유지되지 않게 설정이 가능 합니다.
- 사용자에게 대한 정보를 서버에 두기 때문에 쿠키보다 보안에 좋지만, 사용자가 많아질수록 서버 메모리를 많이 차지하게 됩니다.
- 즉 동접자 수가 많은 웹 사이트인 경우 서버에 과부하를 주게 되므로 성능 저하의 요인이 됩니다.
- 클라이언트가 Request를 보내면, 해당 서버의 엔진이 클라이언트에게 유일한 ID를 부여하는 데 이것이 세션 ID입니다.

세션의 동작 방식

1. 클라이언트가 서버에 접속 시 세션 ID를 발급 받음
2. 클라이언트는 세션 ID에 대해 쿠키를 사용해서 저장하고 가지고 있음
3. 클라이언트는 서버에 요청할 때, 이 쿠키의 세션 ID를 같이 서버에 전달해서 요청
4. 서버는 세션 ID를 전달 받아서 별다른 작업없이 세션 ID로 세션에 있는 클라이언트 정보를 가져와서 사용
5. 클라이언트 정보를 가지고 서버 요청을 처리하여 클라이언트에게 응답

세션의 특징

- 각 클라이언트에게 고유 ID를 부여
- 세션 ID로 클라이언트를 구분해서 클라이언트의 요구에 맞는 서비스를 제공
- 보안 면에서 쿠키보다 우수
- 사용자가 많아질수록 서버 메모리를 많이 차지하게 됨

세션의 사용 예

- 로그인 같이 보안상 중요한 작업을 수행할 때 사용

세션을 사용하면 좋은데 왜 쿠키를 사용할까?

세션은 서버의 자원을 사용하기 때문에 무분별하게 만들다보면 서버의 메모리가 감당할 수 없어질 수가 있고 속도가 느려질 수 있기 때문에 쿠키가 유리한 경우가 있습니다.

쿠키/세션은 캐시와 엄연히 다르다!

- 캐시는 이미지나 css, js파일 등을 브라우저나 서버 앞 단에 저장해놓고 사용하는 것입니다.
- 한번 캐시에 저장되면 브라우저를 참고하기 때문에 서버에서 변경이 되어도 사용자는 변경되지 않게 보일 수 있는데 이런 부분을 캐시를 지워주거나 서버에서 클라이언트로 응답을 보낼 때 header에 캐시 만료시간을 명시하는 방법등을 이용할 수 있습니다.
- 보통 쿠키와 세션의 차이를 물어볼 때 저장위치와 보안에 대해서는 잘 말하는데 사실 중요한 것은 라이프사이클을 얘기하는 것입니다.

[JavaScript] session 사용하기 (feat. sessionStorage)

- 자바스크립트에서 세션 사용하기 - js나 스크립팅이 아닌 영역에서 session을 사용하는 방법은 sessionStorage 라는 녀석을 사용하면 된다. sessionStorage에서 지원하는 메서드는 다음과 같다. >sessionStorage 지원 메서드 목록 메서드 설명 setItem(key,

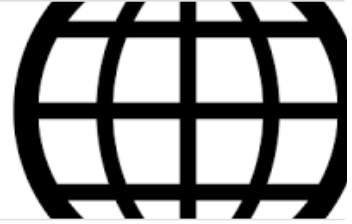
🔗 <https://mine-it-record.tistory.com/265>

JS

[Session Management (3/3)] Javascript로 구현하고 이해하는 Session Management

Session Management는 크게 3가지 페이지로 구성된다. 1) 쿠키란 무엇인가 2) 세션이란 무엇인가 3) Javascript로 구현하고 이해하는 Session Management 본 글은 마지막 장인 Javascript로 구현하고 이해하는 Session Management 에 대해 다룬다. 학습방법 : 이번 페이지만 읽어도 간단한 예

🔗 <https://etloveguitar.tistory.com/81>



java script 에서 세션등록하기

```
sessionStorage.setItem('세션 명', '세션 값'); // 세션 등록 sessionStorage.getItem('세션 명') // 세션 가져오기
```

🔗 <https://eternalteach.tistory.com/49>



세션과는 또 다른 로그인 유지 방식: 토큰

세션 방식은 안전하고 효과적이지만 **단점**도 있습니다. 서버는 요청마다 함께 딸려 오는 세션 아이디를 바로바로 확인할 수 있도록 로그인한 사용자의 아이디를 메모리라는 '책상'에 올려둡니다. 메모리에 올려둔 데이터를 빠르게 확인할 수 있다는 장점이 있는 대신 **공간이 한정**되어 있죠. 서버에 동시 접속하는 사용자가 많아지면 **메모리 공간이 부족**해져서 **서버에 부하**가 걸리고 **화면**이 움직이지 않는 등의 문제가 발생할 수 있습니다.

메모리 공간을 많이 차지하는 세션 방식의 대안은 로그인한 사용자에게 세션 아이디 대신 **토큰**을 발급해 주는 것입니다. 이러한 토큰에는 **특수한 수학적 원리**가 적용되어 있어서 마치 위조 방지 장치가 있는 지폐처럼 **서버만이 유효한 토큰을 발행**할 수 있습니다.

그렇기 때문에 토큰을 받아간 사용자가 이를 쿠키로 저장해 두고 필요할 때마다 제시하면 서버는 따로 책상에 올려놓은 것을 확인할 필요 없이 자기가 발급한 토큰임을 알아보고 사용자의 요청을 허가해 주는 것입니다. 더 이상 이미 로그인한 사용자의 티켓을 책상(메모리)에 올려 두고 있을 필요가 없으니 서버 부하를 줄일 수 있는 것이죠.

토큰 방식은 해당 서버만이 만들 수 있는 토큰을 발급함으로써 상태를 저장하지 않고도 사용자의 로그인 여부를 파악할 수 있도록 합니다.

물론 토큰 방식에도 **한계**는 있습니다. 여러 기기에서의 로그인을 제한하기 위해 필요한 때 에 로그인되어 있는 사용자를 **서버가 강제로 로그아웃**을 시킬 수 있어야 하는데, **토큰 방식**에서는 이것이 **불가능**합니다.

한 번 발행한 토큰은 **유효기간**이 끝나기 전까지 따로 통제할 수 없기 때문에 세션에 비해 토큰 정보를 탈취 당할 가능성이 높습니다. 그러나 토큰은 쿠키처럼 **만료 기간**을 정할 수 있어서 만료 시간을 짧게 지정해 피해를 줄일 수 있습니다. 토큰방식은 쿠키와 세션을 적절히 섞은 것과 비슷합니다.

	세션 방식	토큰 방식
장점	사용자의 상태를 원하는대로 통제 가능	상태를 따로 기억해 둘 필요가 없음
단점	메모리에 로그인되어 있는 사용자의 상태를 보관해야 함	한 번 로그인한 사용자의 상태의 토큰

전송량은 줄이고 속도는 높이고: 캐시

우리는 매일같이 웹사이트나 유튜브, 온라인 게임 등을 통해 이미지, 동영상, 웹 페이지 코드와 같은 대량의 데이터를 서버로부터 전송받습니다. 이러한 **데이터 전송**에는 시간이 소요될 뿐만 아니라 통신비도 지출됩니다. 고화질 동영상처럼 크기가 큰 데이터일수록 비용은 더욱 커집니다.

그러나 한 번 전송받은 데이터는 저장해 놔다가 다시 사용할 때 꺼내 쓴다면 반복적으로 서버에 데이터 전송을 요청할 필요가 없습니다. 이때 사용되는 기술이 **캐시**(cash가 아니고 cache)입니다. 캐시 덕분에 우리는 반복적으로 사용하는 콘텐츠를 빠르게 이용할 수 있고 데이터 사용량도 줄일 수 있습니다.

캐시는 인터넷 환경뿐만 아니라 다양한 곳에서 사용되는 개념입니다. 컴퓨터의 하드웨어 안에서도 메모리 안에 들어있는 정보를 더 빨리 가져올 수 있도록 하는 CPU 캐시 등이 있습니다.

하드디스크나 SSD는 책상 여러 개를 보관해 놓을 수 있는 '창고'와 같아서 메모리보다 저장 공간이 훨씬 넓지만, 저장된 정보를 꺼내는 데 시간이 오래 걸립니다. 그렇기 때문에 빠르고 많이 들어오는 요청에 맞게 세션 아이디를 찾아 확인하는 작업에는 용량은 적지만 더 빨리 이용할 수 있는 메모리가 적합합니다.



일반적으로 사용자 입장에서 가장 가까이 접하는 캐시는 브라우저 캐시입니다. **사용자가 컴퓨터나 스마트폰에서 인터넷 서핑할 때 받아온 데이터는 브라우저에 캐시 형태로 저장됩니다.** 쿠키와 같이 캐시도 각 브라우저의 설정 화면에서 조작해 비울 수 있습니다. 캐시 덕분에 사용자는 같은 사이트를 다시 방문하거나 동영상을 다시 시청할 때 추가로 통신비를 지출하지 않고 로딩 없이 콘텐츠를 이용할 수 있습니다.

서버 부담은 줄이고 사용자와는 가깝게: CDN

캐시를 사용하지 않고 매번 정보가 전송되는 것은 사용자로서도 불편하지만, **이용자가 많은 서비스일수록 서버에도 큰 부담이 됩니다.** 전 세계의 수많은 사용자로부터 끊임없이 들어오는 요청을 처리하다 보면 아무리 **고사양의 컴퓨터도 감당하기 어려운 부하**가 걸리거든요. 이로 인한 **응답 속도 저하나 서버 오류는 결국 사용자의 불편**으로 이어집니다.

이런 문제를 해결하기 위해 사용하는 것이 바로 **CDN(콘텐츠 전송 네트워크)**입니다. **CDN은 지리적으로 분산된 여러 개의 서버를 이용해 웹 콘텐츠를 사용자와 가까운 서버에서 전송함으로써 전송 속도를 높입니다.**

치킨 체인점이 전국의 주문을 본사에서 전부 처리하지 않고 각지의 체인점에서 받는 것처럼 일정 규모 이상의 사용자를 가진 서비스들은 CDN을 사용하여 사용자의 요청을 가까운 서버에서 분산 처리합니다.

서버가 데이터를 전 세계 각지에 세워진 캐시 저장 및 전달용 컴퓨터(CDN 업체 소유)들에 보내면 사용자는 본 서버가 아닌 본인에게서 가장 가까운 캐시 서버로 요청을 보내고 데이터를 받아오는 것입니다.

CDN은 여러 지역에 설치된 캐시 서버들을 사용하여 본 서버로 들어오는 요청들을 분산 처리하는 서비스입니다.

CDN을 사용하면 본 서버는 캐시 서버에 데이터를 한 번씩만 전송하면 됩니다. CDN이 마치 세계적으로 유명한 체인점 같아서 본사의 부담을 최소화하고 사용자 역시 데이터를 보다 빠르고 안정적으로 받아들 수 있는 것입니다. 대량의 데이터를 전송하는 서비스, 특히 유튜브나 넷플릭스와 같은 동영상 서비스에 CDN은 필수입니다.

AWS의 CloudFront나 CloudFlare, Akamai 등이 전 세계 캐시 서버를 운영하는 대표적인 CDN 업체입니다. GS 네오텍, SK 브로드밴드, KT 등 한국 서비스에 최적화된 국내 CDN 업체들도 많이 있죠. 온라인 서비스를 운영할 때는 필요에 따라 호스팅 업체와는 별개의 CDN 서비스를 선택하여 이용하기도 하고, AWS의 CloudFront 등 호스팅 및 클라우드 업체에서 함께 제공하는 CDN을 사용하기도 합니다. 서비스의 특성과 사용자 규모, 대상 지역에 따라 적합한 CDN 업체와 상품을 선택함으로써 안정적으로 서비스를 제공할 수 있습니다.

완벽 정리! 쿠키, 세션, 토큰, 캐시 그리고 CDN

웹 서핑을 하면서 어떤 사이트에 들어가면 쿠키를 설정하라는 문구를 본 적이 있을 거예요. 이 쿠키 때문에 쇼핑 사이트에 로그인하지 않아도 장바구니에 물건을 담아두거나 검색 기록에서 이전에 입력했던 검색어들을 찾아볼 수 있습니다.

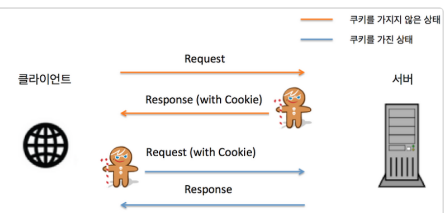
<https://hongong.hanbit.co.kr/완벽-정리-쿠키-세션-토큰-캐시-그리고-cdn/>



쿠키와 세션 개념

노션 페이지(아래 내용과 동일) 개요 쿠키와 세션은 개발자 말고도 인터넷 사용자라면 누구나 많이 들어본 단어입니다. 하지만 개념에 대해서는 많은 사람들이 헷갈려 하기에 쉽고 간단하게 정리해보려고 합니다. 일단 쿠키를 발급받고 사용하는 과정에 대해서 그림으로 보

러면 <https://interconnection.tistory.com/74>



이벤트 위임

캡처링과 버블링을 활용하면 강력한 이벤트 핸들링 패턴인 *이벤트 위임(event delegation)* 을 구현할 수 있습니다.

이벤트 위임은 비슷한 방식으로 여러 요소를 다뤄야 할 때 사용됩니다.



이벤트 위임을 사용하면 요소마다 핸들러를 할당하지 않고, 요소의 공통 조상에 이벤트 핸들러를 단 하나만 할당해도 여러 요소를 한꺼번에 다룰 수 있습니다.

공통 조상에 할당한 핸들러에서 `event.target` 을 이용하면 실제 어디서 이벤트가 발생했는지 알 수 있습니다. 이를 이용해 이벤트를 핸들링합니다. 이벤트 위임은 유사한 요소에 동일한 핸들러를 적용할 때 주로 사용하지만 꼭 이런 경우에만 사용할 수 있는 것은 아닙니다.

이벤트 위임은 다음과 같은 알고리즘으로 동작합니다.

1. 컨테이너에 하나의 핸들러를 할당합니다.
2. 핸들러의 `event.target` 을 사용해 이벤트가 발생한 요소가 어디인지 알아냅니다.
3. 원하는 요소에서 이벤트가 발생했다고 확인되면 이벤트를 핸들링합니다.

이벤트 위임의 **장점**은 다음과 같습니다.

- 많은 핸들러를 할당하지 않아도 되기 때문에 초기화가 단순해지고 메모리가 절약됩니다.
- 요소를 추가하거나 제거할 때 해당 요소에 할당된 핸들러를 추가하거나 제거할 필요가 없기 때문에 코드가 짧아집니다.
- `innerHTML` 이나 유사한 기능을 하는 스크립트로 요소 덩어리를 더하거나 뺄 수 있기 때문에 DOM 수정이 쉬워집니다.

이벤트 위임에도 물론 **단점**이 있습니다.

- 이벤트 위임을 사용하려면 이벤트가 반드시 버블링 되어야 합니다. 하지만 몇몇 이벤트는 버블링 되지 않습니다. 그리고 낮은 레벨에 할당된 핸들러엔 `event.stopPropagation()` 를 쓸 수 없습니다.
- 컨테이너 수준에 할당된 핸들러가 응답할 필요가 있는 이벤트이든 아니든 상관없이 모든 하위 컨테이너에서 발생하는 이벤트에 응답해야 하므로 CPU 작업 부하가 늘어날 수 있습니다. 그런데 이런 부하는 무시할 만한 수준이므로 실제로는 잘 고려하지 않습니다.

실제로 실무에서 이벤트 위임을 사용해본 경험이 있을까요?
있다면, 어떻게 사용했는지 짧게 공유해보아요!


이벤트 위임

 <https://ko.javascript.info/event-delegation>



왜 이벤트 위임(delegation)을 해야 하는가?

과거 웹오피스의 워드를 개발했던 때에, 각주 기능을 개발했던 적이 있다. 문서의 특정 부분에 커서를 두고 "각주"를 추가하면 해당 부분에 대한 코멘트를 문서의 우측 영역에 작성할 수 있는 기능이었다. 며칠 만에 기능을 완성하고 개별적으로 성능을 측정했다. 각주의

 https://ui.toast.com/posts/ko_20160826

· 사용하면,

본문에 대한 참조 문헌이
의 낱말, 문장 등의 뜻을..

접기