쿠키/세션, JWT, 로그인 동작

- 쿠키/세션 방식으로 로그인을 구현하고

쿠키/세션 사용 이유

HTTP(hypertext transfer protocol) 에는 비연결성과 비상태성이라는 특징이 있음.

이로 인해 요청 이후 사용자와의 연결 상태를 유지하거나 상태정보를 저장하지 않기 때문에 사용자 식별이 불가하여 같은 사용자가 요청을 하더라도 매번 새로운 사용자로 인식하는 단점이 있음.

이런 HTTP 비상태, 비연결의 특성을 보완한 기술이 쿠키와 세션.

쿠키(Cookie)

- 서버를 대신하여 정보를 웹 브라우저를 이용하고 있는 컴퓨터에 저장하고 사용자가 요청할 때 그 정보를 함께 보내서 서버가 사용자를 식별할 수 있게 해준다. 이로써 로그인 기능도 이용 가능하게 되고 사용자에 맞게 사용자 별로 다른 정보를 제공하게 되는데, 이 때 저장되는 데이터를 쿠키라고 한다.
- 쿠키 주요 사용 목적
  1. 세션 관리 : 로그인, 사용자 닉네임, 장바구니 등 서버가 알아야 할 정보 저장
  2. 개인화 : 사용자 별로 적절한 페이지를 보여줄 수 있음
  3. 트래킹 : 사용자의 행동과 패턴을 분석하고 기록함
- 쿠키 사용 예
  1. ID 저장, 로그인 상태 유지 (페이지 이동 시 재 로그인 필요 없음)
  2. 7일 간 다시보지 않기.(쿠키에 체크한 날짜를 기록, 다시 방문 했을 대의 시간을 비교)
  3. 최근 검색한 상품 광고 추천
  4. 쇼핑몰 장바구니 기능 등
- Key, Value로 구성. String 형태로 이루어져 있음. 4KB 이상 저장 불가
- 브라우저 마다 저장되는 쿠키가 다름(크롬에서 보다가 익스플로러로 가면 내 정보를 기억 못하는 이유)

Session 쿠키 & Permanent 쿠키

- Session 쿠키 : 웹 브라우저 종료 시 제거되는 쿠키
- Permanent 쿠키 : 웹 브라우저 종료 시에도 유지되는 쿠키

세션(Session)

서버가 사용자를 식별할 때 사용하는 것.

쿠키에 직접 저장하여 서버 전달 시 비밀번호와 같은 인증 정보의 유출 가능성이 큼.

이를 방지하기 위해 세션은 비밀번호와 같은 인증 정보를 식별자로 형태를 바꾸어 저장하여 서버에 전달됨.

웹 브라우저가 서버에 요청 시 서버는 세션 해당 클라이언트에 세션 아이디를 할당해서 응답할 때 함께 전달, 웹 브라우저는 이 세션 아이디를 쿠키에 저장해두고 매 요청마다 세션 아이디를 함께 전달한다.

- 세션 동작 순서
    1. 클라이언트가 서버에 처음으로 Request를 보냄 (첫 요청이기 때문에 session id가 존재하지 않음)
    2. 서버에서는 session id 쿠키 값이 없는 것을 확인하고 새로 발급해서 응답
    3. 이후 클라이언트는 전달받은 session id 값을 매 요청마다 헤더 쿠키에 넣어서 요청
    4. 서버는 session id를 확인하여 사용자를 식별
    5. 클라이언트가 로그인을 요청하면 서버는 session을 로그인한 사용자 정보로 갱신하고 새로운 session id를 발급하여 응답
    6. 이후 클라이언트는 로그인 사용자의 session id 쿠키를 요청과 함께 전달하고 서버에서도 로그인된 사용자로 식별 가능
    7. 클라이언트 종료 (브라우저 종료) 시 session id 제거, 서버에서도 세션 제거

JWT(Json Web Token) 인증 방식

토큰 기반 인증 방식으로, 클라이언트의 세션 상태를 저장하는 것이 아닌 필요한 정보를 토큰 형태로 저장해 클라이언트가 가지고 있고 그것을 증명서처럼 사용하는 방식

- 토큰의 구성(Header, Payload, Verify Signature)
    - Header : 토큰 구성 정보를 암호화할 방식(알고리즘), 타입 등이 들어간다.

    - Payload : 서버에서 보낼 데이터가 들어간다. 일반적으로 사용자의 고유 ID값, 유효기간이 들어감
    - Verify Signature : 인코딩한 Header와 Payload 그리고 Secret key를 더한 후 서명된다.
- Verify Signature 만 Secret key를 통해 암호화 된다.
- JWT 동작 순서
    1. 사용자가 로그인을 한다.
    2. 서버에서는 계정정보를 읽어 사용자를 확인 후, 사용자의 고유한 ID값을 부여한 후, 기타 정보와 함께 Payload에 넣습니다.
    3. JWT 토큰의 유효기간을 설정합니다.
    4. 암호화할 SECRET KEY를 이용해 ACCESS TOKEN을 발급합니다.
    5. 사용자는 Access Token을 받아 저장한 후, 인증이 필요한 요청마다 토큰을 헤더에 실어 보냅니다.
    6. 서버에서는 해당 토큰의 Verify Signature를 SECRET KEY로 복호화한 후, 조작 여부, 유효기간을 확인합니다.
    7. 검증이 완료된다면, Payload를 디코딩하여 사용자의 ID에 맞는 데이터를 가져옵니다.

JWT 인증 방식 장단점

- 쿠키/세션 방식과의 차이(JWT 인증 방식 등장 배경)

  쿠키/세션 방식은 세션 저장소(서버)에 정보를 넣는 반면, JWT는 토큰 안에 유저의 정보를 넣어 사용자에 보낸다는 차이를 갖는다. 별도의 저장소를 이용하느냐, 인증을 위한 암호화를 하느냐의 차이로 서버의 확장성에 영향을 준다.

- 장점
    1. 세션/쿠키는 별도 저장소의 관리가 필요한데 비해 JWT는 발급 후 검증만 하면되므로, 추가 저장소가 필요하지 않다. 이는 서버를 확장하거나 유지, 보수하는데 유리하다.
    2. 확장성이 뛰어나다. 토큰 기반 다른 인증 시스템에 접근이 가능.(Facebook 로그인, Google 로그인 등)
- 단점
    1. 이미 발급된 JWT에 대해서는 돌이킬 수 없다. (유효기간을 짧게 하여 해결 가능)

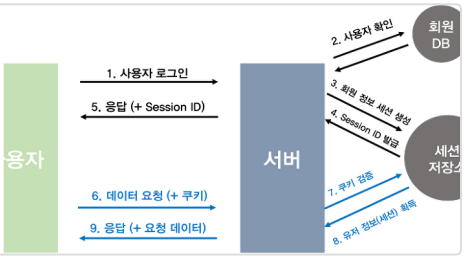
2. Payload 정보가 제한적이다. 따로 암호화하지 않기 때문에 중요 정보 넣을 수 없음
3. 길이가 길다. 필요 요청이 많아질 수록 서버의 자원 낭비 발생

참고

쉽게 알아보는 서버 인증 1편(세션/쿠키 , JWT)

앱 개발을 처음 배우게 됐을 때, 각종 화면을 디자인해보면서 프론트엔드 개발에 큰 흥미가 생겼습니다. 한때 프론트엔드 개발자를 꿈꾸기도 했었죠 (현실은 ...) 그러나 서버와 통신을 처음 배웠을 때 마냥 쉬운 일

https://tansfil.tistory.com/58



쿠키(Cookie)와 세션(Session) & 로그인 동작 방법

웹에서 쿠키와 세션를 이용한 회원 인증이 어떻게 이루어지는 지와 세션과 쿠키에 대한 개념, 용도 등에 대해서 알아본다. HTTP 프로토콜에는 비연결성(Connectionless)과 비상태성(Stateless)라는 특징

https://cjh5414.github.io/cookie-and-session/