# spring 한스푼(6주차)

http? / 프론트-백엔드 흐름도 /
code review / 유저 구현 / jwt?

# HTTP (HyperText Transfer Protocol)

**설명**

인터넷에서 데이터 통신을 위한 규칙 및 규약 중 하나로, 웹 브라우징 및 웹 서비스와 관련된 데이터 교환에 사용

**동작**

요청 (request) : client -> server

응답 (response) : server -> client

HTML 뿐만 아니라 XML, JSON 형태로도 주고 받을 수 있다.

# Request method

GET – 서버로부터 데이터 **요청**하여 가져온다. Read

POST – 서버에 데이터를 **생성**, 작성 Create

PUT – 서버의 데이터를 **수정**, 작성 Update

DELETE – 서버의 데이터를 **삭제** Delete

ex)

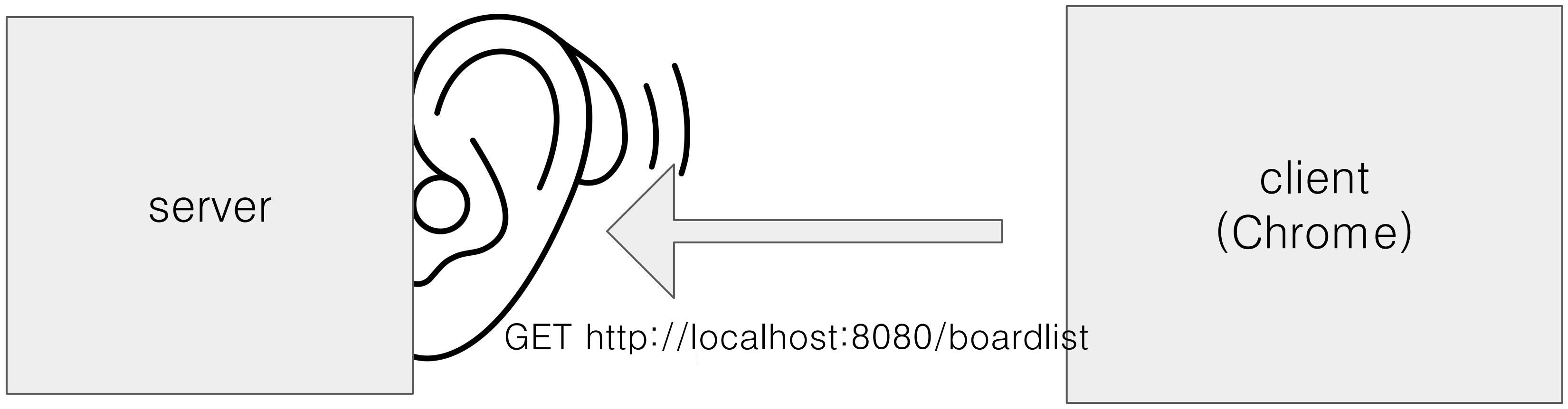
GET http://localhost:8080/boardlist

User-Agent : Mozilla/5.0 (Windows NT 10. win64; x64) …

Upgrade-Insecure-Requests : 1

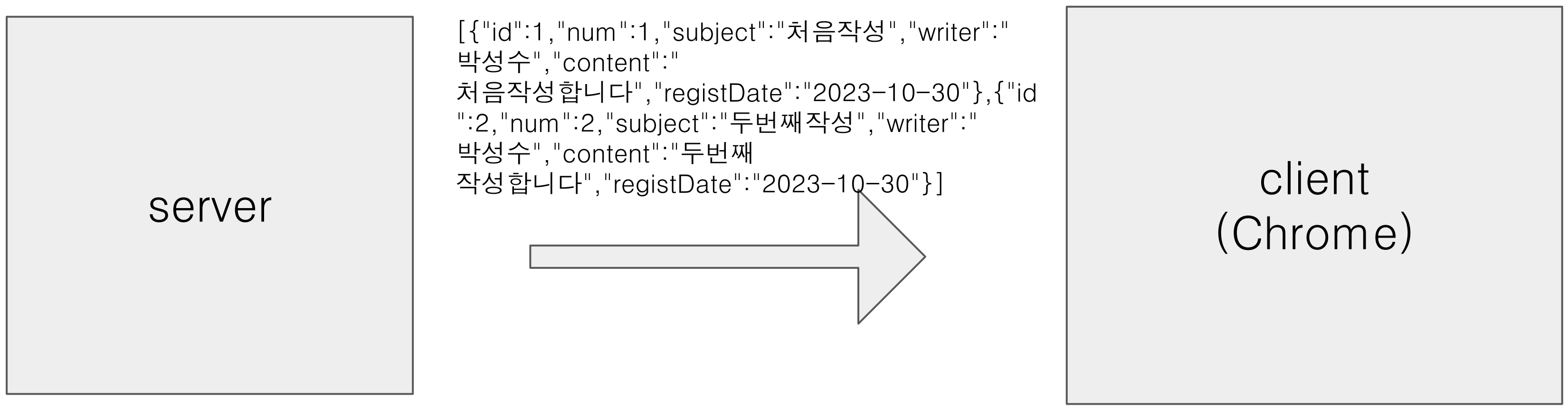
{} : body 부분

요청

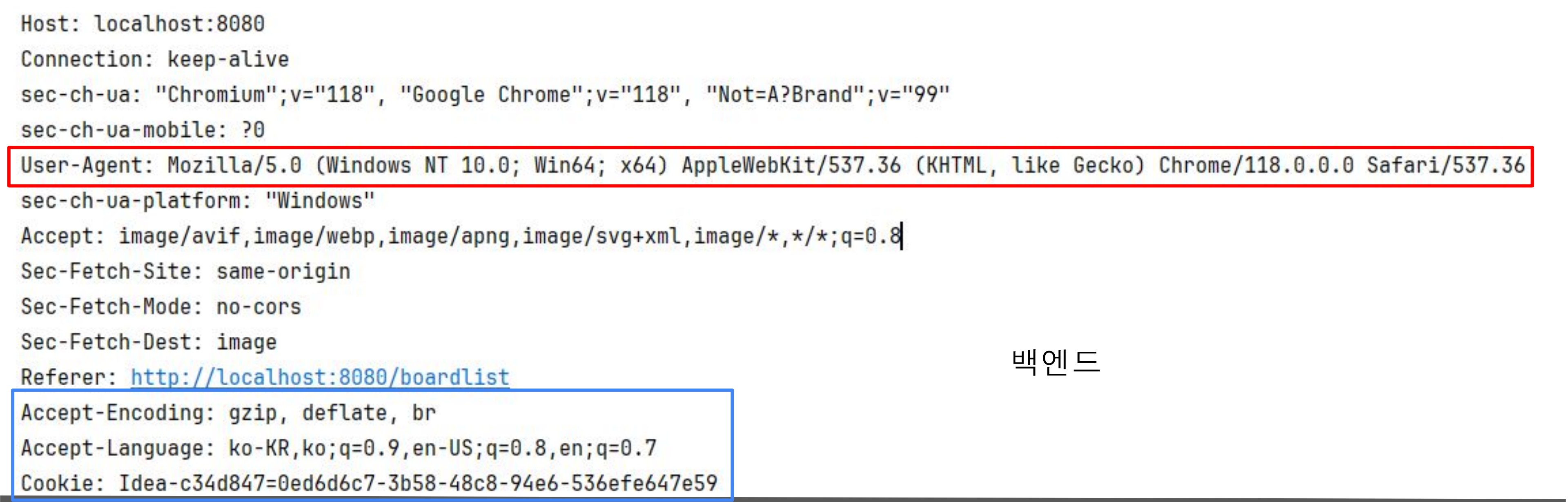


```
GET http://localhost:8080/boardlist
```

응답

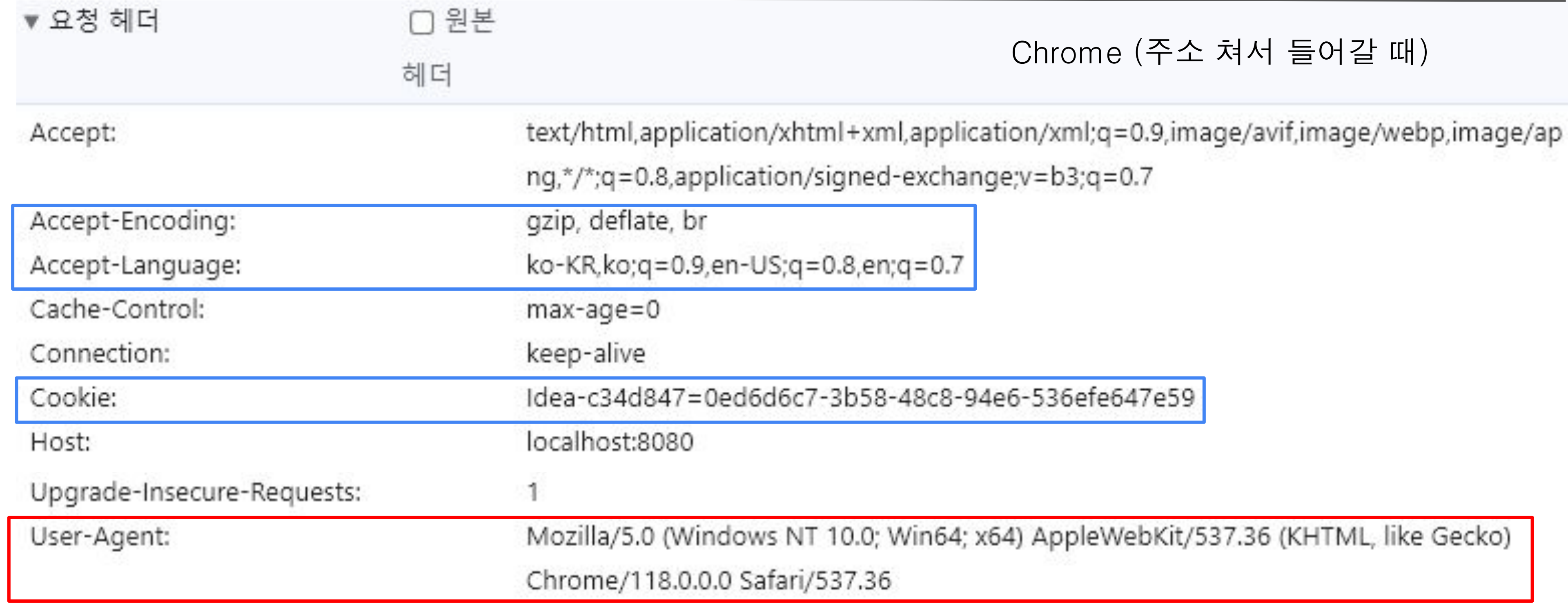


```
[{"id":1,"num":1,"subject":"처음작성","writer":"
박성수","content":"
처음작성합니다","registDate":"2023-10-30"},{"id
":2,"num":2,"subject":"두번째작성","writer":"
박성수","content":"두번째
작성합니다","registDate":"2023-10-30"}]
```



```
Host: localhost:8080
Connection: keep-alive
sec-ch-ua: "Chromium";v="118", "Google Chrome";v="118", "Not=A?Brand";v="99"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36
sec-ch-ua-platform: "Windows"
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:8080/boardlist
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: Idea-c34d847=0ed6d6c7-3b58-48c8-94e6-536efe647e59
```

백엔드

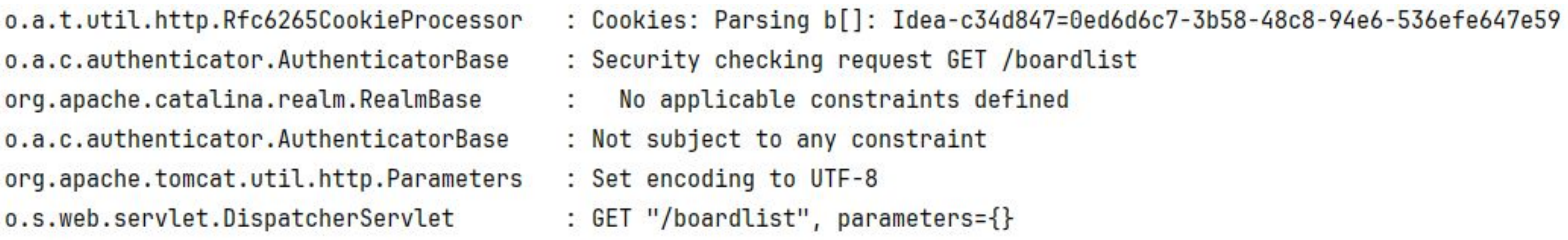


Chrome (주소 쳐서 들어갈 때)

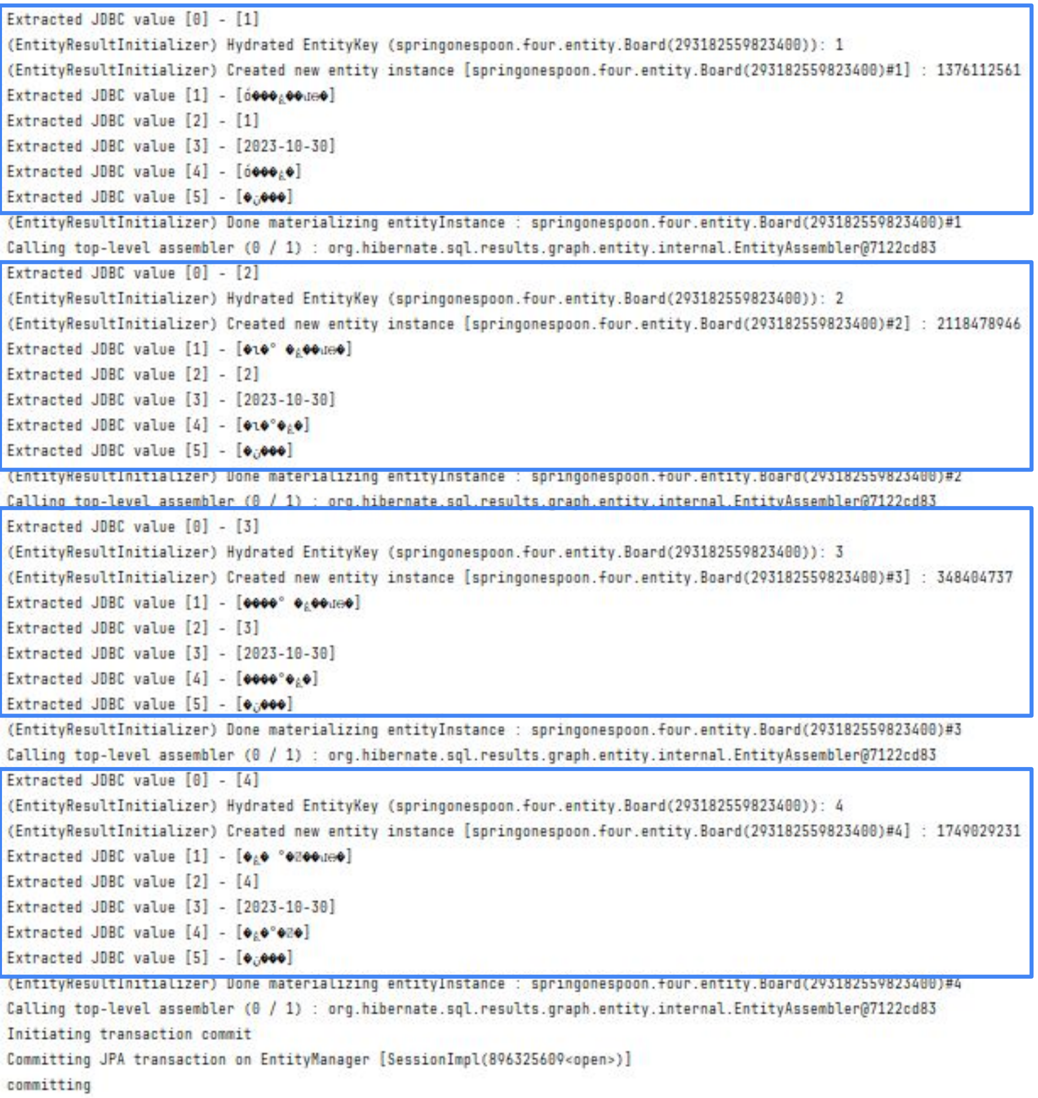
▼ 요청 헤더 ☐ 원본

헤더

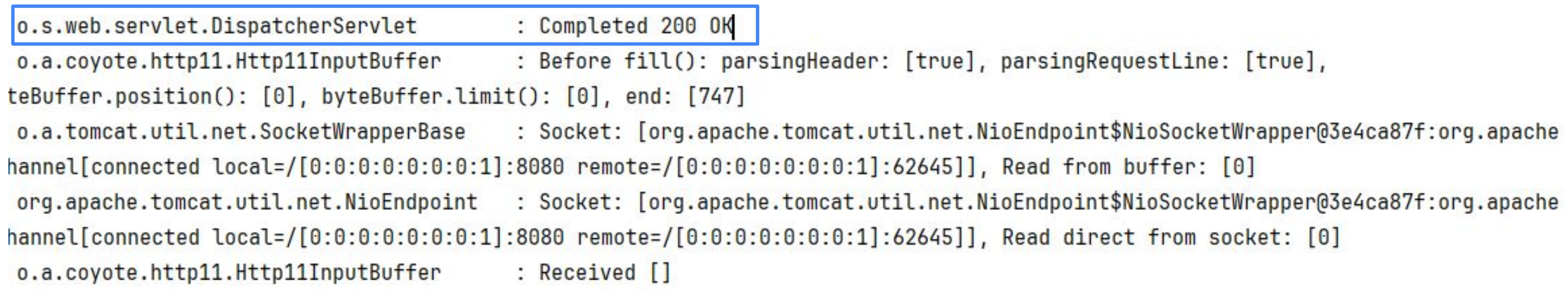
| | |
|---|---|
| Accept: | text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 |
| Accept-Encoding: | gzip, deflate, br |
| Accept-Language: | ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7 |
| Cache-Control: | max-age=0 |
| Connection: | keep-alive |
| Cookie: | Idea-c34d847=0ed6d6c7-3b58-48c8-94e6-536efe647e59 |
| Host: | localhost:8080 |
| Upgrade-Insecure-Requests: | 1 |
| User-Agent: | Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 |



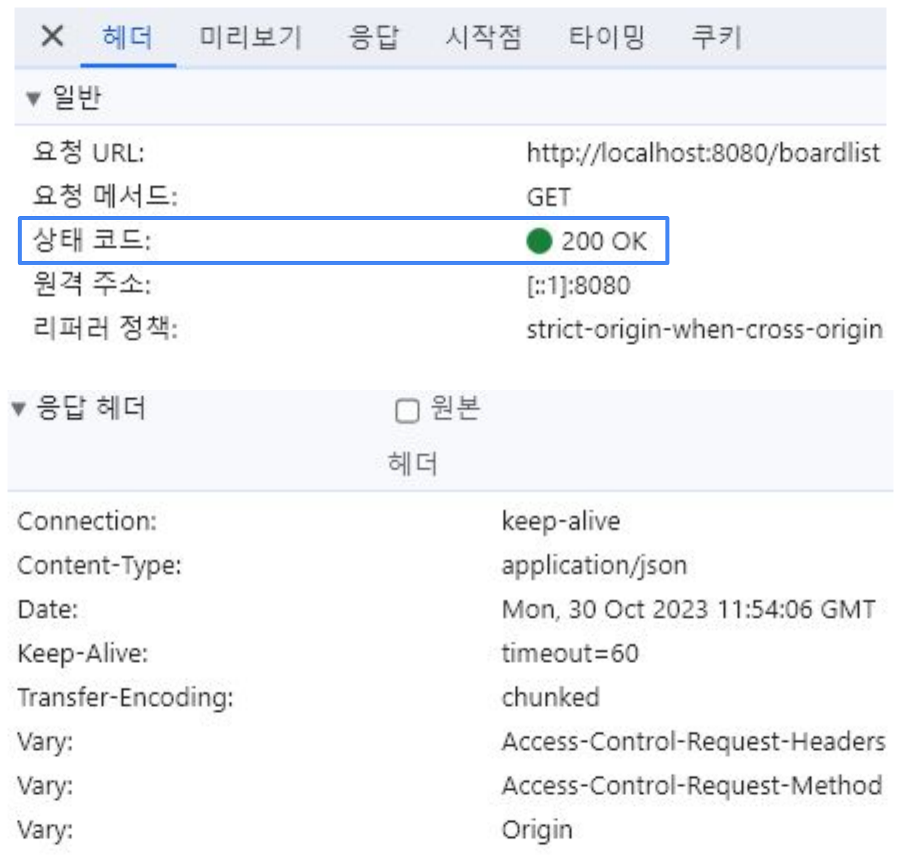
```
o.a.t.util.http.Rfc6265CookieProcessor   : Cookies: Parsing b[]: Idea-c34d847=0ed6d6c7-3b58-48c8-94e6-536efe647e59
o.a.c.authenticator.AuthenticatorBase    : Security checking request GET /boardlist
org.apache.catalina.realm.RealmBase      :   No applicable constraints defined
o.a.c.authenticator.AuthenticatorBase    : Not subject to any constraint
org.apache.tomcat.util.http.Parameters   : Set encoding to UTF-8
o.s.web.servlet.DispatcherServlet        : GET "/boardlist", parameters={}
```



```
Extracted JDBC value [0] - [1]
(EntityResultInitializer) Hydrated EntityKey (springonespoon.four.entity.Board(293182559823400)): 1
(EntityResultInitializer) Created new entity instance [springonespoon.four.entity.Board(293182559823400)#1] : 1376112561
Extracted JDBC value [1] - [ó���ˌ��ᴊe�]
Extracted JDBC value [2] - [1]
Extracted JDBC value [3] - [2023-10-30]
Extracted JDBC value [4] - [ó���ˌ�]
Extracted JDBC value [5] - [�ˋ���]
(EntityResultInitializer) Done materializing entityInstance : springonespoon.four.entity.Board(293182559823400)#1
Calling top-level assembler (0 / 1) : org.hibernate.sql.results.graph.entity.internal.EntityAssembler@7122cd83
Extracted JDBC value [0] - [2]
(EntityResultInitializer) Hydrated EntityKey (springonespoon.four.entity.Board(293182559823400)): 2
(EntityResultInitializer) Created new entity instance [springonespoon.four.entity.Board(293182559823400)#2] : 2118478946
Extracted JDBC value [1] - [�ɩ�° �ˌ��ᴊe�]
Extracted JDBC value [2] - [2]
Extracted JDBC value [3] - [2023-10-30]
Extracted JDBC value [4] - [�ɩ�°�ˌ�]
Extracted JDBC value [5] - [�ˋ���]
(EntityResultInitializer) Done materializing entityInstance : springonespoon.four.entity.Board(293182559823400)#2
Calling top-level assembler (0 / 1) : org.hibernate.sql.results.graph.entity.internal.EntityAssembler@7122cd83
Extracted JDBC value [0] - [3]
(EntityResultInitializer) Hydrated EntityKey (springonespoon.four.entity.Board(293182559823400)): 3
(EntityResultInitializer) Created new entity instance [springonespoon.four.entity.Board(293182559823400)#3] : 348404737
Extracted JDBC value [1] - [����° �ˌ��ᴊe�]
Extracted JDBC value [2] - [3]
Extracted JDBC value [3] - [2023-10-30]
Extracted JDBC value [4] - [����°�ˌ�]
Extracted JDBC value [5] - [�ˋ���]
(EntityResultInitializer) Done materializing entityInstance : springonespoon.four.entity.Board(293182559823400)#3
Calling top-level assembler (0 / 1) : org.hibernate.sql.results.graph.entity.internal.EntityAssembler@7122cd83
Extracted JDBC value [0] - [4]
(EntityResultInitializer) Hydrated EntityKey (springonespoon.four.entity.Board(293182559823400)): 4
(EntityResultInitializer) Created new entity instance [springonespoon.four.entity.Board(293182559823400)#4] : 1749029231
Extracted JDBC value [1] - [�ˌ� °�Z��ᴊe�]
Extracted JDBC value [2] - [4]
Extracted JDBC value [3] - [2023-10-30]
Extracted JDBC value [4] - [�ˌ�°�Z�]
Extracted JDBC value [5] - [�ˋ���]
(EntityResultInitializer) Done materializing entityInstance : springonespoon.four.entity.Board(293182559823400)#4
Calling top-level assembler (0 / 1) : org.hibernate.sql.results.graph.entity.internal.EntityAssembler@7122cd83
Initiating transaction commit
Committing JPA transaction on EntityManager [SessionImpl(896325609<open>)]
committing
```



```
o.s.web.servlet.DispatcherServlet        : Completed 200 OK
 o.a.coyote.http11.Http11InputBuffer      : Before fill(): parsingHeader: [true], parsingRequestLine: [true],
teBuffer.position(): [0], byteBuffer.limit(): [0], end: [747]
 o.a.tomcat.util.net.SocketWrapperBase    : Socket: [org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@3e4ca87f:org.apache
hannel[connected local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:62645]], Read from buffer: [0]
 org.apache.tomcat.util.net.NioEndpoint   : Socket: [org.apache.tomcat.util.net.NioEndpoint$NioSocketWrapper@3e4ca87f:org.apache
hannel[connected local=/[0:0:0:0:0:0:0:1]:8080 remote=/[0:0:0:0:0:0:0:1]:62645]], Read direct from socket: [0]
 o.a.coyote.http11.Http11InputBuffer      : Received []
```

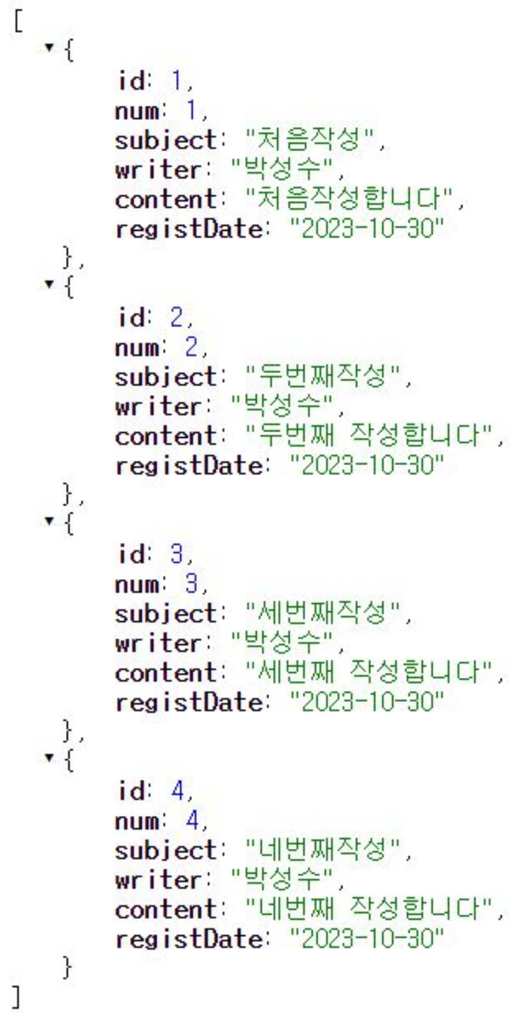


▼ 일반

| | |
|---|---|
| 요청 URL: | http://localhost:8080/boardlist |
| 요청 메서드: | GET |
| 상태 코드: | ● 200 OK |
| 원격 주소: | [::1]:8080 |
| 리퍼러 정책: | strict-origin-when-cross-origin |

▼ 응답 헤더 ☐ 원본 헤더

| | |
|---|---|
| Connection: | keep-alive |
| Content-Type: | application/json |
| Date: | Mon, 30 Oct 2023 11:54:06 GMT |
| Keep-Alive: | timeout=60 |
| Transfer-Encoding: | chunked |
| Vary: | Access-Control-Request-Headers |
| Vary: | Access-Control-Request-Method |
| Vary: | Origin |



```
[
  ▾{
      id: 1,
      num: 1,
      subject: "처음작성",
      writer: "박성수",
      content: "처음작성합니다",
      registDate: "2023-10-30"
   },
  ▾{
      id: 2,
      num: 2,
      subject: "두번째작성",
      writer: "박성수",
      content: "두번째 작성합니다",
      registDate: "2023-10-30"
   },
  ▾{
      id: 3,
      num: 3,
      subject: "세번째작성",
      writer: "박성수",
      content: "세번째 작성합니다",
      registDate: "2023-10-30"
   },
  ▾{
      id: 4,
      num: 4,
      subject: "네번째작성",
      writer: "박성수",
      content: "네번째 작성합니다",
      registDate: "2023-10-30"
   }
]
```

# http state (HTTP 응답코드)

서버가 클라이언트에게 응답할 때 쓰는 코드로써

1xx, 2xx, 3xx, 4xx, 5xx 가 있다.

주로 쓰이는 응답 코드

200 OK - 서버가 정상적으로 처리되었을 때

201 Create - 서버가 정상적으로 데이터를 저장 했을 때

400 Bad request - 클라이언트가 잘못 요청 할 때

403 Forbidden - 서버에서 거부

404 Not found - 찾을 수 없는 리소스.

500 Internal Server error - 내부 서버 오류

## HTTPS와의 차이점

HTTP는 보안 설정이 안되있어 해커들이 한 서버 주소 HTTP를 잠적하고 있으면 정보가 새어 나갈 수 있다.

HTTPS는 전송 내용이 암호화 되어 전달 되기 때문에 정보가 새어나가도 복호화 하기도 어렵기에 안전하다.

code review

## User Entity

```
@Builder
@Getter
@Entity
@NoArgsConstructor
@AllArgsConstructor
class User{
     @Id
     @GeneratedValue (strategy = GenerationType.AUTO)
     private Long id;
     @Column(nullable = false, unique = true)
     private String username;
     private String password;
     private String email;
     private String phone;
}
```

## UserRepository

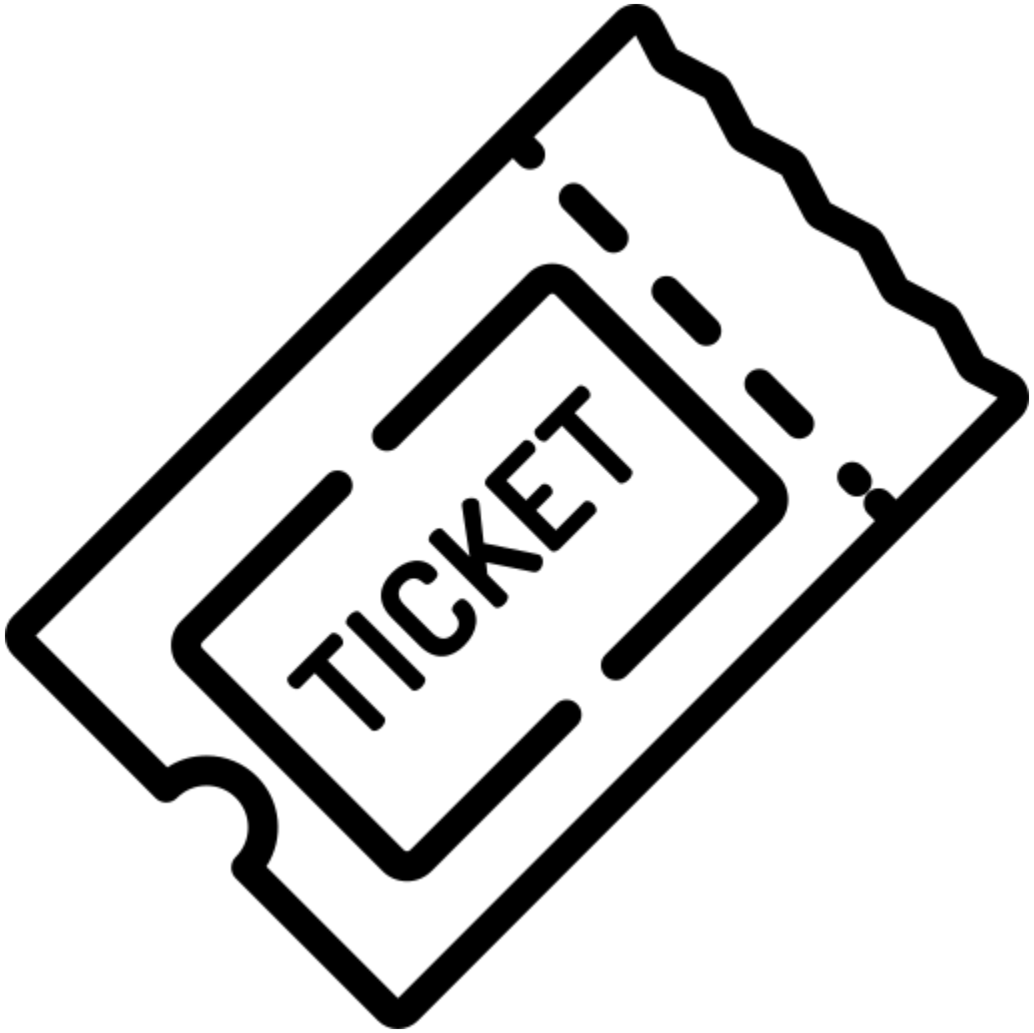
```
interface UserRepository extends JpaRepository<User, Long>{

    User findOneByUsername(String username);

    boolean existsByUsername(String username);

}
```

## JWT(Json Web Token)



## session을 알고 가자!

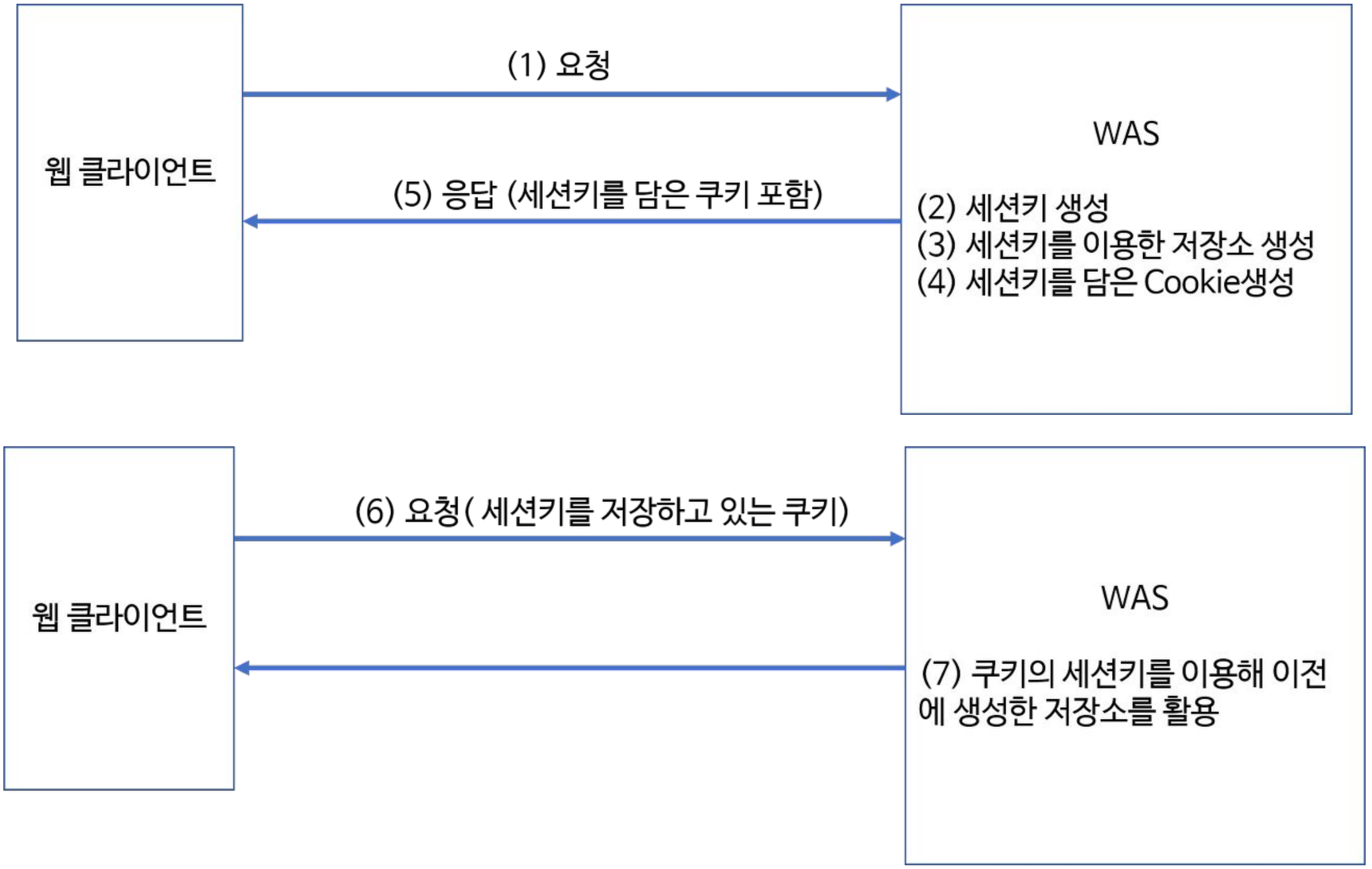
로그인 인증 방식 중 하나.

1)클라이언트가 로그인

2)서버가 응답으로 세션키를 담은 쿠키를 줌.

3)클라이언트가 로그인 인증 하기 위해 세션키를 포함한 요청보냄.

4)서버가 세션키를 판별후 클라이언트에게 응답(true or false)함.


웹 클라이언트
(1) 요청
WAS
(5) 응답 (세션키를 담은 쿠키 포함)
(2) 세션키 생성
(3) 세션키를 이용한 저장소 생성
(4) 세션키를 담은 Cookie생성
웹 클라이언트
(6) 요청( 세션키를 저장하고 있는 쿠키)
WAS
(7) 쿠키의 세션키를 이용해 이전에 생성한 저장소를 활용

## 영상

JWT 생활코딩

https://youtu.be/XXseiON9CV0?si=AGFGw2qA1N-T5hO5

## JWT 뭐가 들어 있을까?

header

typ : “JWT” -  토큰의 타입을 지정.

alg : “HS256” - 해싱 알고리즘을 지정

payload(정보)  선택적.

iss : 토큰 발급자 / sub : 토큰 제목 / aud : 토큰 대상자

exp : 토큰의 만료 시간 / nbf : 토큰의 활성 시간 / iat : 토큰의 발급 시간

jti : JWT의 고유 식별자

signature(서명)

```
   HMACSHA256(
 base64UrlEncode(header) + "." +
 base64UrlEncode(payload),
 secret)
```