MSA 7~8주차

스터디원	소현
Status	Done
URL	https://github.com/SPRING-STUDY-2023/sohyeon-spring-cloud-msa/pull/4
비고	MSA Section6
제출 마감일	@November 1, 2023 10:00 PM

Users Microservice(2)

- Users Microservice - Login
- JWT (Json Web Token)
- API Gateway service - AuthorizationHeaderFilter

인증과 권한 기능 개요

Features

- ✓ ~~회원 등록~~
- ☐ 회원 로그인
- ✓ ~~상세 정보 확인~~
- ✓ ~~상품 주문~~
- ✓ ~~주문 내역 확인~~

기능	URI	HTTP Method
로그인	/user-service/login	POST

- **RequestLogin**
→ 사용자 로그인 정보를 저장하기 위한 RequestLogin Model 클래스
- **AuthenticationFilter (인증)**
→ Spring Security를 이용한 로그인 요청 발생 시 작업을 처리해주는 Custom Filter 클래스
→ UsernamePasswordAuthenticationFilter 상속
- **WebSecurity (권한)**
→ 사용자 요청에 대해 AuthenticationFilter를 거치도록 수정

AuthenticationFilter 추가

RequestLogin

```
@Data
public class RequestLogin {
    @NotNull(message = "Email cannot be null")
    @Size(min = 2, message = "Email not be less than two characters")
    @Email
    private String email;

    @NotNull(message = "Password cannot be null")
    @Size(min = 8, message = "Password must be equals or greater than 8 characters")
    private String password;
}
```

AuthenticationFiletr

```
public class AuthenticationFilter extends UsernamePasswordAuthenticationFilter {
    @Override
    public Authentication attemptAuthentication(
        HttpServletRequest request,
        HttpServletResponse response
    ) throws AuthenticationException {
        try {
            RequestLogin creds = new ObjectMapper().readValue(request.getInputStream(), RequestLo

            return getAuthenticationManager().authenticate(
                new UsernamePasswordAuthenticationToken(
                    creds.getEmail(),
                    creds.getPassword(),
                    new ArrayList<>()
                )
            );
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

WebSecurity

```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class WebSecurity {
    private final UserService userService;
    private final BCryptPasswordEncoder bCryptPasswordEncoder;
    private final ObjectPostProcessor<Object> objectPostProcessor;

    private static final String[] WHITE_LIST = {
        "/users/**",
        "/",
        "/**"
    };

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        return http
            .csrf(CsrfConfigurer::disable)
            .authorizeHttpRequests(authorizeRequests ->
                authorizeRequests
                    // .requestMatchers(WHITE_LIST).permitAll()
                    .requestMatchers(PathRequest.toH2Console()).permitAll()
                    .requestMatchers(new IpAddressMatcher("127.0.0.1")).permitAll()
            )
            .addFilter(getAuthenticationFilter())
            .headers(header -> header.frameOptions(HeadersConfigurer.FrameOptionsConfig::disable)
            .build();
    }

    public AuthenticationManager authenticationManager(AuthenticationManagerBuilder auth) throws
        auth.userDetailsService(userService).passwordEncoder(bCryptPasswordEncoder);
        return auth.build();
    }

    private AuthenticationFilter getAuthenticationFilter() throws Exception {
        AuthenticationFilter authenticationFilter = new AuthenticationFilter();
        AuthenticationManagerBuilder builder = new AuthenticationManagerBuilder(objectPostProcess
        authenticationFilter.setAuthenticationManager(authenticationManager(builder));
        return authenticationFilter;
    }
}
```

```
public interface UserService extends UserDetailsService {
    ...
}
```

```
public class UserServiceImpl implements UserService {
    ...

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        return null;
    }
}
```

Trouble Shooting

Spring Security 최신버전(Spring Boot 3.X.X 대)의 WebSecurity 설정 공유드립니다. - 인프런 | 질문 & 답변

최신버전으로 진행하다보니 막혔었는데요. 구글링, ChatGPT 등을 통해서 동작하는 코드 공유드립니다.정확한 구현은 아닐 수 있겠지만, 강의를 진행하는 데는 문제 없는 것 같습니다. 참고만 부탁드려요~package com.example.userservice.security...

https://www.inflearn.com/questions/812490/spring-security-최신버전-spring-boot-3-x-x-대-의-websecurity-설정-공유드립니다

loadUserByUsername() 구현

UserService

```
public class UserServiceImpl implements UserService {
    ...

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        UserEntity userEntity = userRepository.findByEmail(username)
            .orElseThrow(() -> new UsernameNotFoundException(username));

        return new User(
            userEntity.getEmail(),
            userEntity.getEncryptedPwd(),
            true,
            true,
            true,
            true,
            new ArrayList<>()
        );
    }
    ...
}
```

UserRepository

```
public interface UserRepository extends JpaRepository<UserEntity, Long> {
    ..
    Optional<UserEntity> findByEmail(String email);
}
```

Routes 정보 변경

Application YML 파일

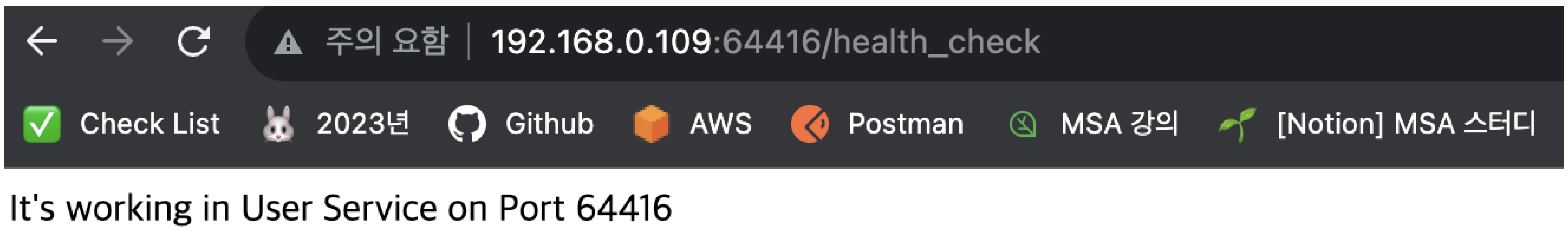
```
spring:
    cloud:
        gateway:
            routes:
                - id: user-service
          uri: lb://USER-SERVICE
          predicates:
            - Path=/user-service/login # 로그인
            - Method=POST
          filters:
            - RemoveRequestHeader=Cookie
            - RewritePath=/user-service/(?<segment>.*), /$\{segment}
        - id: user-service
          uri: lb://USER-SERVICE
          predicates:
            - Path=/user-service/users # 회원가입
            - Method=POST
          filters:
            - RemoveRequestHeader=Cookie
            - RewritePath=/user-service/(?<segment>.*), /$\{segment}
        - id: user-service
          uri: lb://USER-SERVICE
          predicates:
            - Path=/user-service/** # 그 외 모든 user-service API
            - Method=GET
          filters:
            - RemoveRequestHeader=Cookie
            - RewritePath=/user-service/(?<segment>.*), /$\{segment}
```

💡 **RewritePath=/user-service/(?<segment>.*), /$\{segment}**

`localhost:8761/user-service/**` 요청 형식을 `localhost:8761/**` 패턴으로 동일화

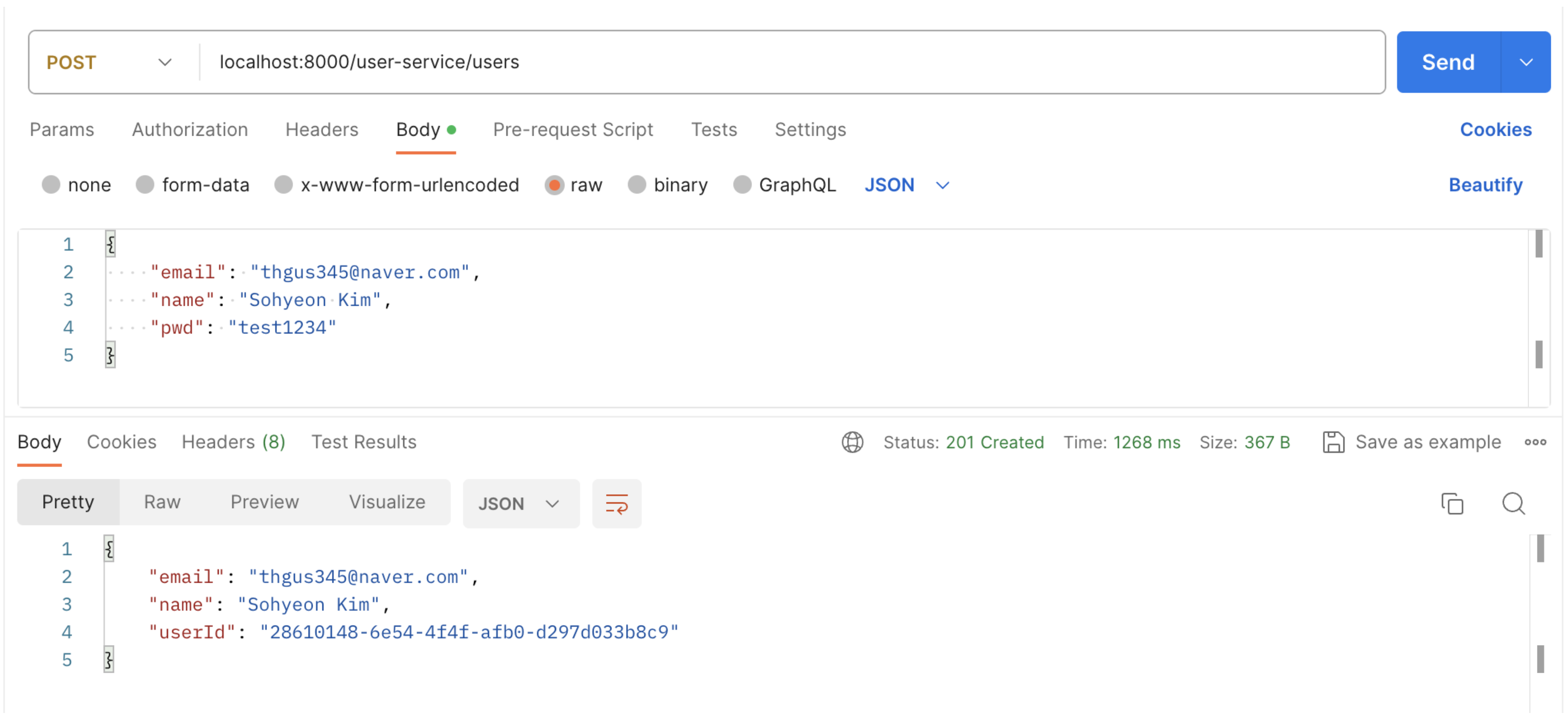
Routes 테스트

Health Check



It's working in User Service on Port 64416

회원가입



localhost:8000/user-service/users

none · form-data · x-www-form-urlencoded · [✓ raw] · binary · GraphQL · JSON

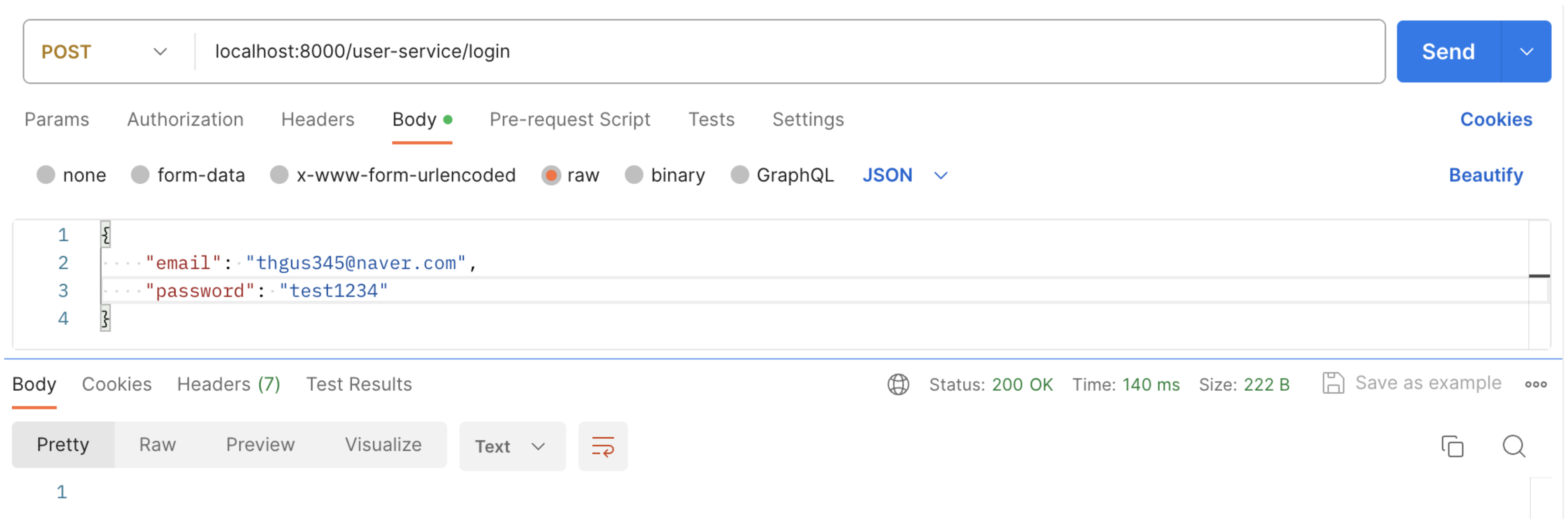
```
{
    "email": "thgus345@naver.com",
    "name": "Sohyeon Kim",
    "pwd": "test1234"
}
```

Body · Cookies · Headers (8) · Test Results — Status: 201 Created Time: 1268 ms Size: 367 B

```
{
    "email": "thgus345@naver.com",
    "name": "Sohyeon Kim",
    "userId": "28610148-6e54-4f4f-afb0-d297d033b8c9"
}
```

회원가입 성공

로그인 (Spring Security에서 제공)



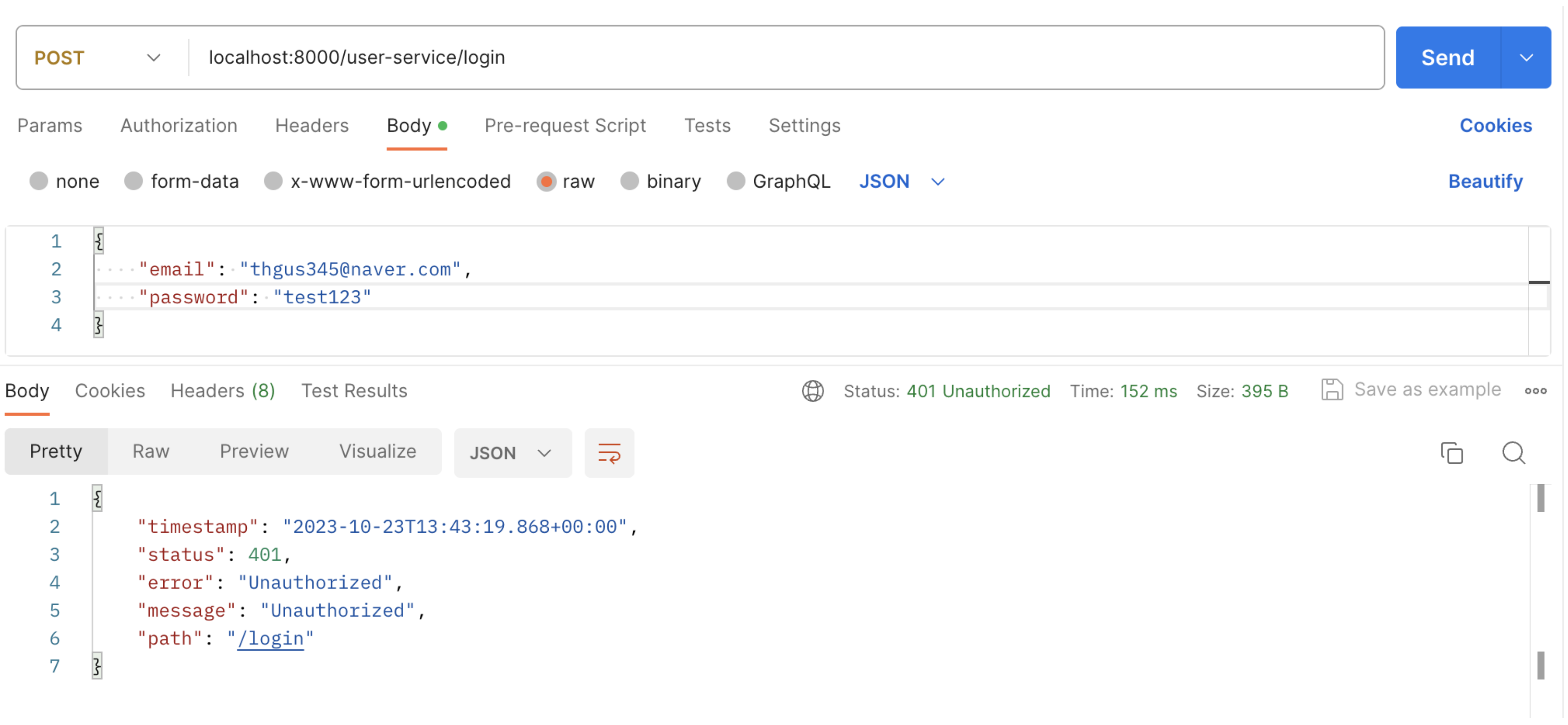
localhost:8000/user-service/login

none · form-data · x-www-form-urlencoded · [✓ raw] · binary · GraphQL · JSON

```
{
    "email": "thgus345@naver.com",
    "password": "test1234"
}
```

Body · Cookies · Headers (7) · Test Results — Status: 200 OK Time: 140 ms Size: 222 B

로그인 성공



localhost:8000/user-service/login

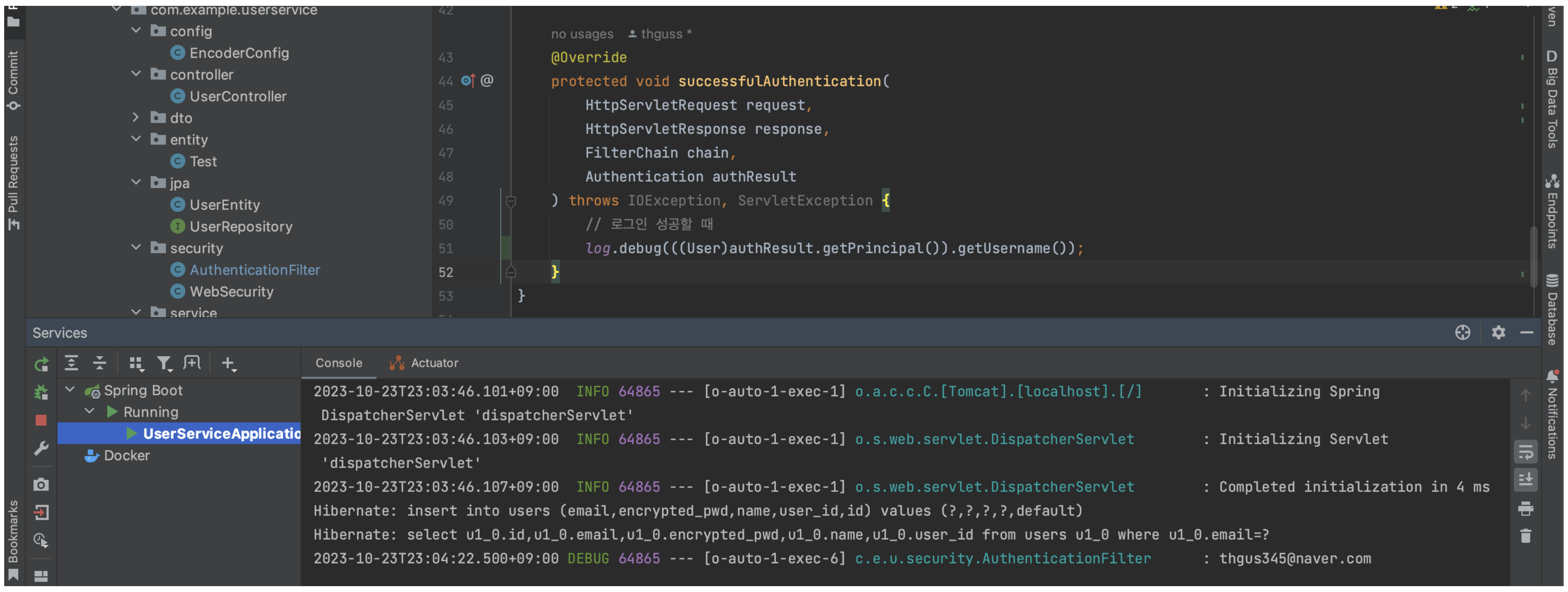
```
{
    "email": "thgus345@naver.com",
    "password": "test123"
}
```

Status: 401 Unauthorized Time: 152 ms Size: 395 B

```
{
    "timestamp": "2023-10-23T13:43:19.868+00:00",
    "status": 401,
    "error": "Unauthorized",
    "message": "Unauthorized",
    "path": "/login"
}
```

로그인 실패

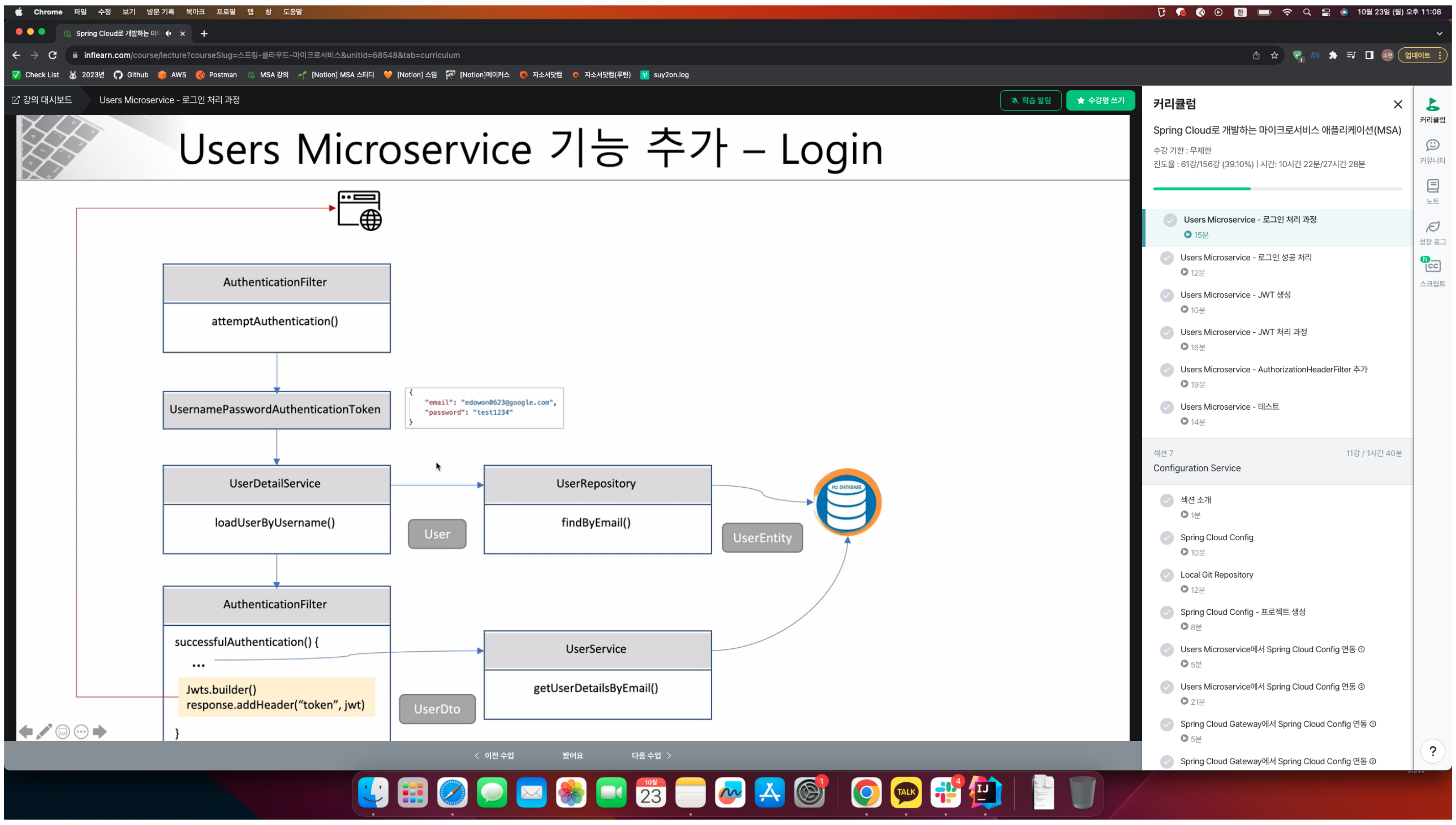
로그인 처리 과정



```
no usages  thguss *
@Override
protected void successfulAuthentication(
    HttpServletRequest request,
    HttpServletResponse response,
    FilterChain chain,
    Authentication authResult
) throws IOException, ServletException {
    // 로그인 성공할 때
    log.debug(((User)authResult.getPrincipal()).getUsername());
}
}
```

```
2023-10-23T23:03:46.101+09:00  INFO 64865 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring
 DispatcherServlet 'dispatcherServlet'
2023-10-23T23:03:46.103+09:00  INFO 64865 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet
 'dispatcherServlet'
2023-10-23T23:03:46.107+09:00  INFO 64865 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 4 ms
Hibernate: insert into users (email,encrypted_pwd,name,user_id,id) values (?,?,?,?,default)
Hibernate: select u1_0.id,u1_0.email,u1_0.encrypted_pwd,u1_0.name,u1_0.user_id from users u1_0 where u1_0.email=?
2023-10-23T23:04:22.500+09:00 DEBUG 64865 --- [o-auto-1-exec-6] c.e.u.security.AuthenticationFilter      : thgus345@naver.com
```

로그인 성공 후 들어온 username 입력 값 logging


Users Microservice 기능 추가 – Login
AuthenticationFilter
attemptAuthentication()
UsernamePasswordAuthenticationToken
{ "email": "edowon0623@google.com", "password": "test1234" }
UserDetailService
loadUserByUsername()
User
UserRepository
findByEmail()
UserEntity
AuthenticationFilter
successfulAuthentication() {
...
Jwts.builder()
response.addHeader("token", jwt)
}
UserDto
UserService
getUserDetailsByEmail()
커리큘럼
Spring Cloud로 개발하는 마이크로서비스 애플리케이션(MSA)
Users Microservice - 로그인 처리 과정
Users Microservice - 로그인 성공 처리
Users Microservice - JWT 생성
Users Microservice - JWT 처리 과정
Users Microservice - AuthorizationHeaderFilter 추가
Users Microservice - 테스트
Configuration Service
섹션 소개
Spring Cloud Config
Local Git Repository
Spring Cloud Config - 프로젝트 생성
Users Microservice에서 Spring Cloud Config 연동
Spring Cloud Gateway에서 Spring Cloud Config 연동

- AuthenticationFilter의 successfulAuthentication() 수정
  → 인증 성공 시 사용자에게 Token 발생
- WebSecurity의 configure 수정
  → User Service 사용할 수 있도록 메소드 수정
- UserService, UserServiceImpl, UserRepository
  → 사용자 인증을 위한 검색 메소드 추가

로그인 성공 처리

AuthenticationFilter 수정

- 생성자 추가

```
public class AuthenticationFilter extends UsernamePasswordAuthenticationFilter {
    private final UserService userService;
    private final Environment env;

    public AuthenticationFilter(AuthenticationManager authenticationManager,
        UserService userService, Environment env) {
        super.setAuthenticationManager(authenticationManager);
        this.userService = userService;
        this.env = env;
    }
    ...
}
```

- 로그인 성공 후 처리

```
public class AuthenticationFilter extends UsernamePasswordAuthenticationFilter {
    ...

    @Override
    protected void successfulAuthentication(
        HttpServletRequest request,
        HttpServletResponse response,
        FilterChain chain,
        Authentication authResult
    ) throws IOException, ServletException {
        String userName = ((User)authResult.getPrincipal()).getUsername();
        UserDto userDto = userService.getUserDetailByEmail(userName);
    }
}
```

UserService 수정

- 메소드 추가

```
public class UserServiceImpl implements UserService {
    ...

    @Override
    public UserDto getUserDetailByEmail(String email) {
        UserEntity userEntity = userRepository.findByEmail(email)
            .orElseThrow(() -> new UsernameNotFoundException(email));

        return new ModelMapper().map(userEntity, UserDto.class);
    }
}
```

WebSecurity 수정

```
public class WebSecurity {
    ...

    private AuthenticationFilter getAuthenticationFilter() throws Exception {
        return new AuthenticationFilter(
            authenticationManager(new AuthenticationManagerBuilder(objectPostProcessor)),
            userService,
            env
        );
    }
}
```

JWT 생성

Dependency 추가

```
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
        <version>0.9.1</version>
</dependency>
```

```
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.0</version>
</dependency>
```

Application YML 추가

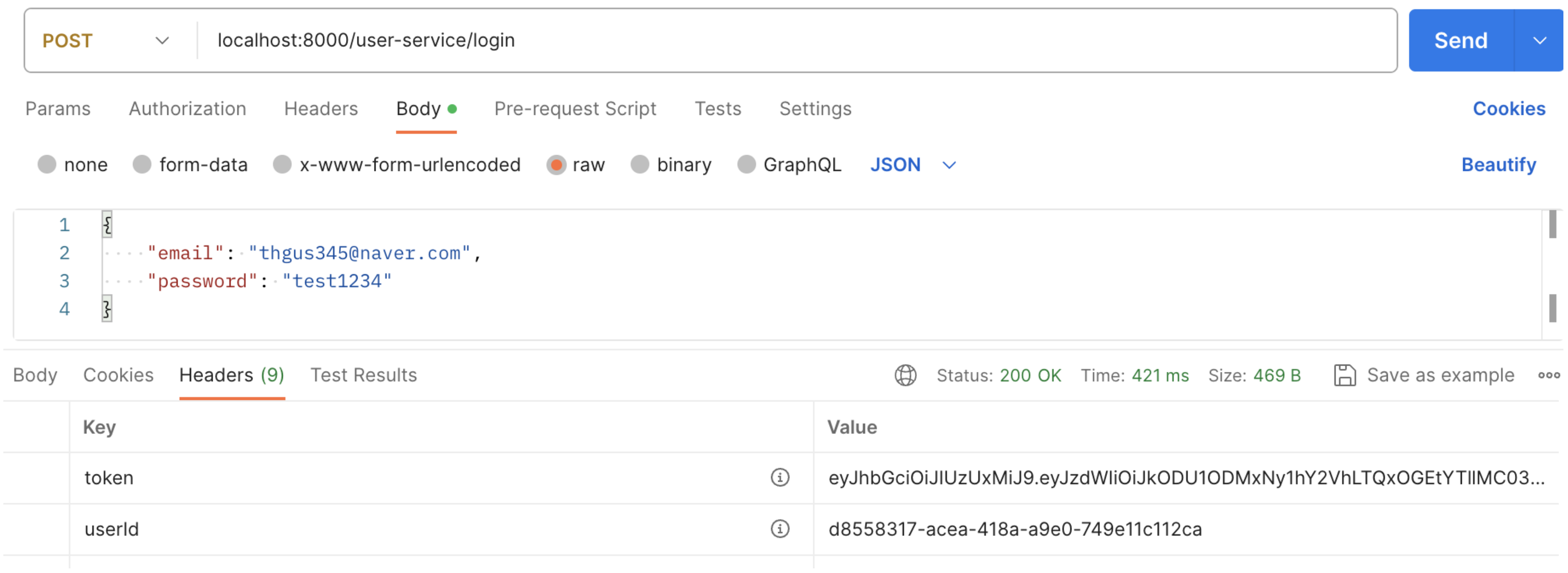
```
token:
  expiration_time: 86400000 # 60 * 60 * 24 * 1000 = 1day
  secret: feelMyRhythm
```

AuthenticationFilter 수정

- successfulAuthentication 재정의

```
@Override
protected void successfulAuthentication(
    HttpServletRequest request,
    HttpServletResponse response,
    FilterChain chain,
    Authentication authResult
) throws IOException, ServletException {
    String userName = ((User)authResult.getPrincipal()).getUsername();
    UserDto userDetails = userService.getUserDetailByEmail(userName);

    String token = Jwts.builder()
        .setSubject(userDetails.getUserId())
        .setExpiration(new Date(System.currentTimeMillis() +
            Long.parseLong(requireNonNull(env.getProperty("token.expiration_time")))))
        .signWith(SignatureAlgorithm.HS512, env.getProperty("token.secret"))
        .compact();

    response.addHeader("token", token);
    response.addHeader("userId", userDetails.getUserId());
}
```

테스트



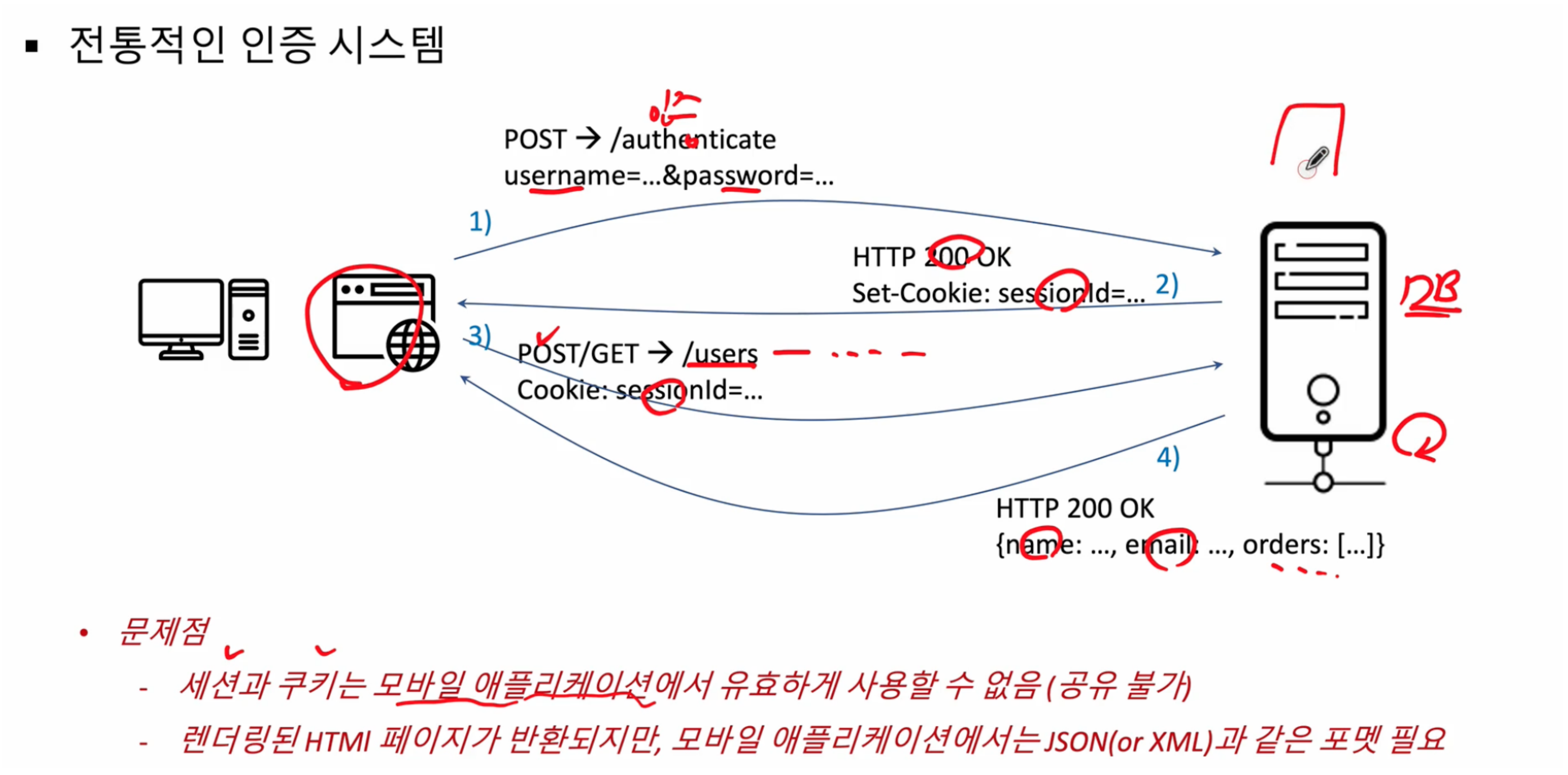
localhost:8000/user-service/login

```
{
    "email": "thgus345@naver.com",
    "password": "test1234"
}
```

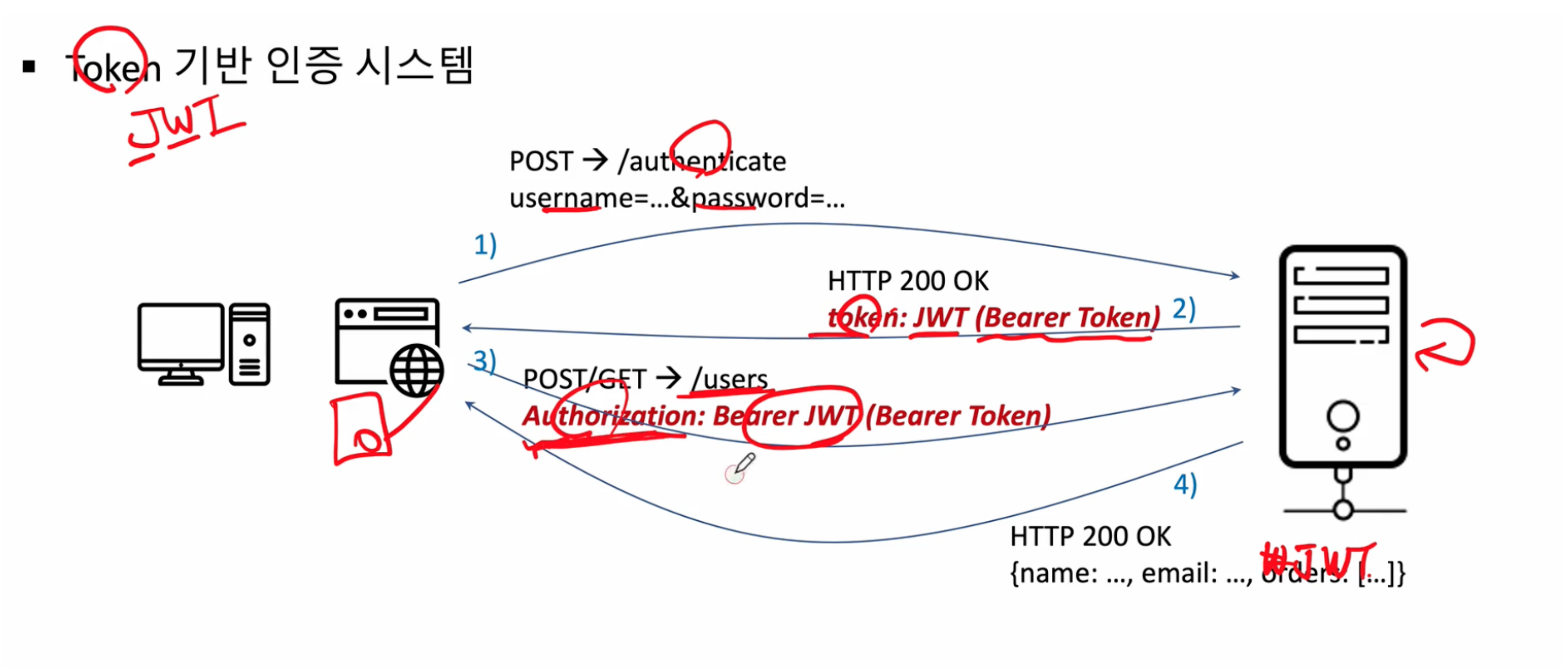
Status: 200 OK Time: 421 ms Size: 469 B

Key	Value
token	eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJkODU1ODMxNy1hY2VhLTQxOGEtYTllMC03...
userId	d8558317-acea-418a-a9e0-749e11c112ca

JWT 처리 과정



• 문제점
- 세션과 쿠키는 모바일 애플리케이션에서 유효하게 사용할 수 없음(공유 불가)
- 렌더링된 HTML 페이지가 반환되지만, 모바일 애플리케이션에서는 JSON(or XML)과 같은 포멧 필요


▪ Token 기반 인증 시스템
JWT
POST → /authenticate
username=...&password=...
1)
HTTP 200 OK
token: JWT (Bearer Token) 2)
3) POST/GET → /users
Authorization: Bearer JWT (Bearer Token)
4)
HTTP 200 OK
{name: ..., email: ..., orders: [...]}
JWT

- 인증 헤더 내에서 사용되는 토큰 포맷
- 2개의 시스템 간에 안전한 방법으로 통신 가능 → 클라이언트 독립적인 서비스(stateless)
- CDN(Content Delivery Network) → 지리적으로 분산된 서버들을 연결한 네트워크
- No Cookie-Session (No CSRF, 사이트 간 요청 위조)
- 지속적인 토큰 저장

- API Gate Service : Spring Security, JWT Token 사용 추가
- AuthorizationHeaderFilter 추가

JWT 처리 과정 (API Gateway Service 수정)

Dependency

```
implementation group: 'io.jsonwebtoken', name: 'jjwt-api', version: '0.11.5'
implementation group: 'io.jsonwebtoken', name: 'jjwt-impl', version: '0.11.5'
implementation group: 'io.jsonwebtoken', name: 'jjwt-jackson', version: '0.11.5'
```

application YML 수정

```
token:
  secret: feelMyRhythm
```

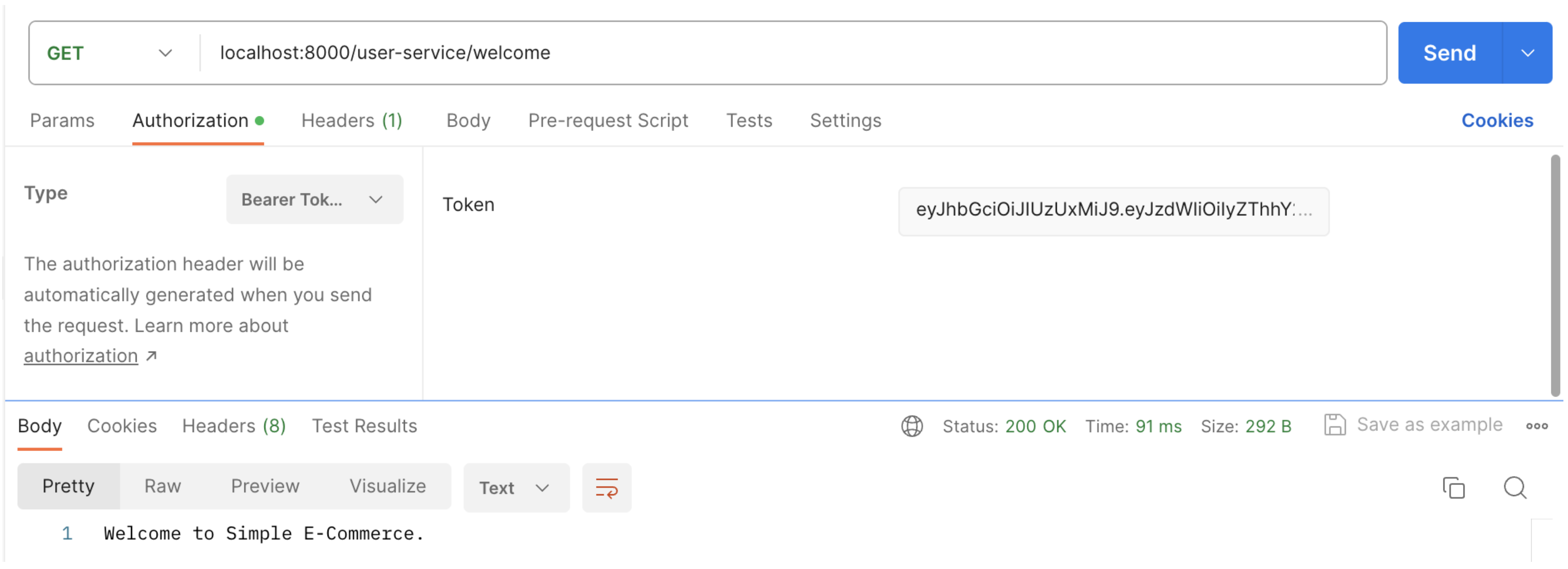
```
- id: user-service
  uri: lb://USER-SERVICE
  predicates:
    - Path=/user-service/**
    - Method=GET
  filters:
    - RemoveRequestHeader=Cookie
    - RewritePath=/user-service/(?<segment>.*), /$\{segment}
    - AuthorizationHeaderFilter
```

AuthorizationHeaderFilter 추가

```
@Slf4j
@Component
public class AuthorizationHeaderFilter extends AbstractGatewayFilterFactory<AuthorizationHeaderFi
    Environment env;

    public static class Config {
    }

    public AuthorizationHeaderFilter(Environment env) {
        super(Config.class);
        this.env = env;
    }

    @Override
    public GatewayFilter apply(Config config) {
        return (exchange, chain) -> {
            ServerHttpRequest request = exchange.getRequest();

            if (!request.getHeaders().containsKey(HttpHeaders.AUTHORIZATION)) {
                return onError(exchange, "No Authorization Header.", HttpStatus.UNAUTHORIZED);
            }

            String authorizationHeader = request.getHeaders().get(HttpHeaders.AUTHORIZATION).get(
            String jwt = authorizationHeader.replace("Bearer", "");

            if (!isJwtValid(jwt)) {
                return onError(exchange, "JWT token is not valid.", HttpStatus.UNAUTHORIZED);
            }

            return chain.filter(exchange);
        };
    }

    private boolean isJwtValid(String jwt) {
        boolean returnValue = true;

        String subject = null;

        try {
            subject = Jwts.parserBuilder()
```

```
                .setSigningKey(env.getProperty("token.secret"))
                .build()
                .parseClaimsJwt(jwt).getBody()
                .getSubject();
        } catch (Exception ex) {
            returnValue = false;
        }

        if (subject == null || subject.isEmpty()) {
            returnValue = false;
        }

        return returnValue;
    }

    // Mono, Flux -> Spring WEbFlux
    private Mono<Void> onError(ServerWebExchange exchange, String err, HttpStatus httpStatus) {
        ServerHttpResponse response = exchange.getResponse();
        response.setStatusCode(httpStatus);

        log.error(err);

        return response.setComplete();
    }
}
```

테스트



GET localhost:8000/user-service/welcome

Type: Bearer Tok...

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Token: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIyZThhY:...

Status: 200 OK Time: 91 ms Size: 292 B

```
Welcome to Simple E-Commerce.
```

Github (PR)

https://github.com/SPRING-STUDY-2023/sohyeon-spring-cloud-msa/pull/4