

KB금융그룹



2025년 상반기 K-디지털 트레이닝

사용자 인증

[KB] IT's Your Life

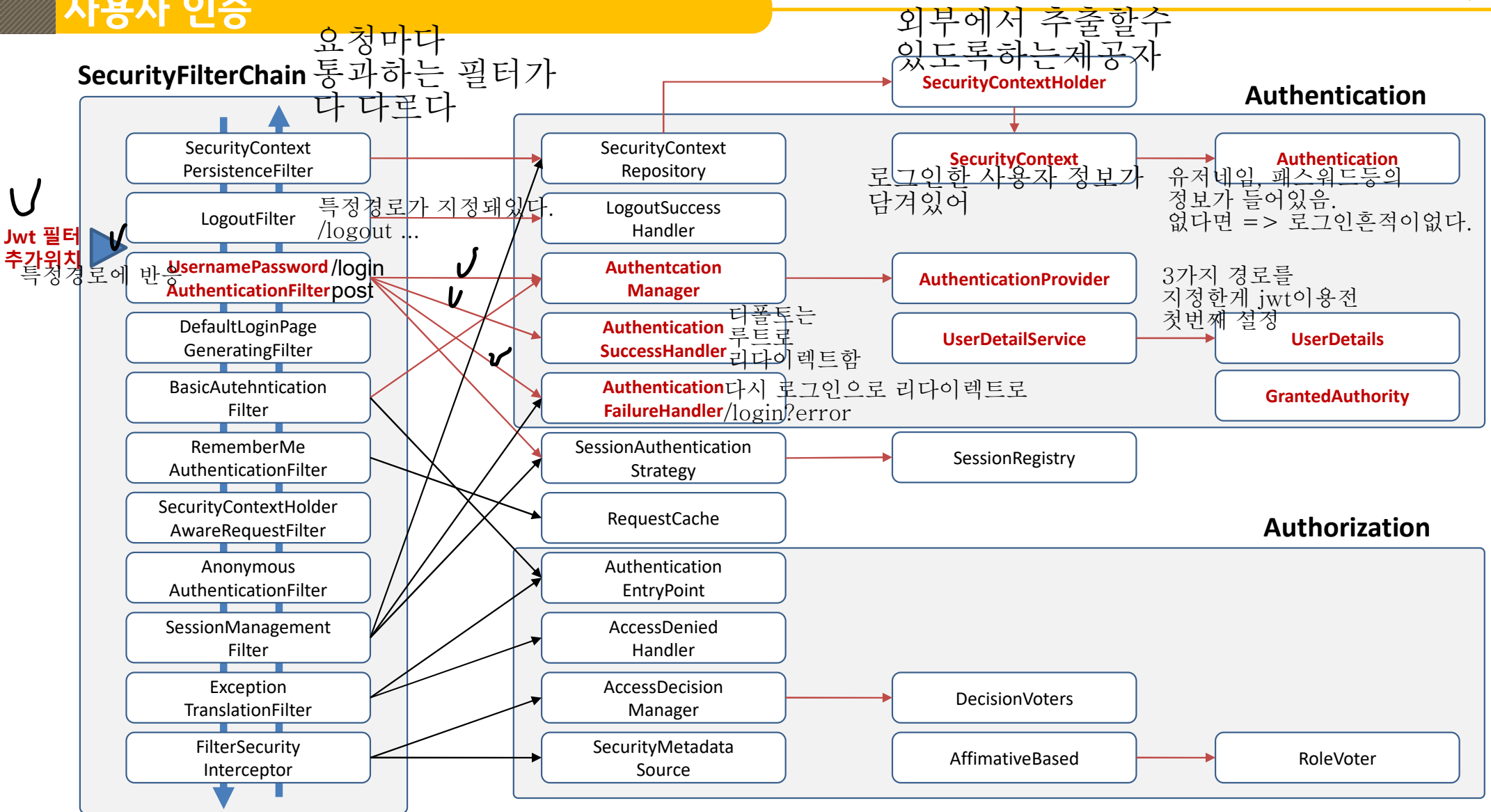
우리의 jwt메카니즘을 넣어서 커스터마이징 하자
이 방법도 여러가지다.

최대한 spring security filter chain에
맞게 커스터마이징 하자

jwt사용시 세션을 이용한 상용자 정보 저장 사용은 X

jwt있으면 로그인 했던 사용자 없으면 로그인 안한 사용자

사용자 인증



✓ JwtAuthenticationFilter

○ 모든 요청에 대해서 헤더에 토큰이 있는지 검사

▪ Authorization 헤더 항목 조사

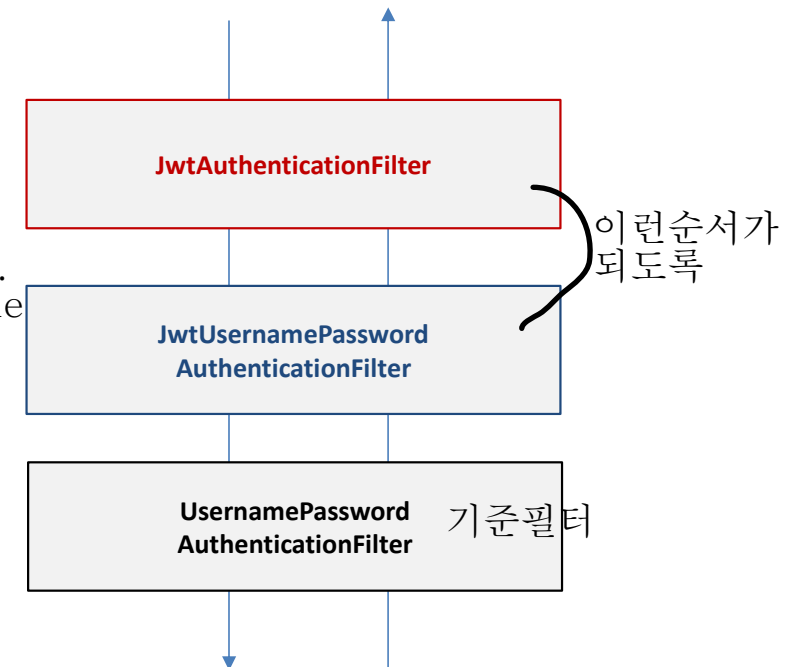
- Authorization: Bearer <jwt 토큰 문자열> 관례다. JWT에서는 해당 문자열이 있으면 로그인한거 아니면 안한거.
- 토큰 추출 클라에서도 맞춰줘야함 토큰 문자열에는 username 이 들어있다. 추출하고 디비에서 정보 꺼내기

○ 토큰의 유효성 검사

- 토큰이 유효하면 SecurityContextHolder에 사용자 로그인 정보 설정
SCH를 통해서 authentication에서 정보 꺼내기

○ 다음 필터로 이동

나중에 authorization 인가를 귀한을 확인할때 해당 정보가 활용되겠지



→ SecurityConfig에서 필터 추가

✓ JwtAuthenticationFilter

○ OncePerRequestFilter 상속 ✓

- 컨트롤러가 forward 했을 때 필터를 다시 통과하게 됨
→ 이미 보안 필터는 통과했는데, 또 통과하게 됨

- 요청당 한 번만 필터가 동작하도록 해줌

) 리퀘스트 하나에 왜 같은 필터가 여러번 통과되냐
포워딩때문에.

servlet -> servlet -> ...

->jsp -> jsp -> ...

포워딩이 여러개있는 경우 계속

할때마다 필터가 동작하여 통과한다.

굳이 이럴필요 없으니까

요청당 한번만 필터가 동작하도록 하는

onceperrequestfilter을 상속받아 필터를 만들자

security.filter.JwtAuthenticationFilter.java V

```
package org.scoula.security.filter;
```

```
...
```

```
@Component V
```

```
@Log4j2
```

```
@RequiredArgsConstructor V
```

생성자 주입 활용

```
public class JwtAuthenticationFilter extends OncePerRequestFilter { V
```

```
    public static final String AUTHORIZATION_HEADER = "Authorization";
```

```
    public static final String BEARER_PREFIX = "Bearer "; // 끝에 공백 있음
```

```
    private final JwtProcessor jwtProcessor;
```

```
    private final UserDetailsService userDetailsService;
```

생성자 DI

디비에서 꺼내기 위한 서비스

```
    private Authentication getAuthentication(String token) { JWT
```

```
        String username = jwtProcessor.getUsername(token);
```

```
        UserDetails principi = userDetailsService.loadUserByUsername(username);
```

```
        return new UsernamePasswordAuthenticationToken(principi, null, principi.getAuthorities());
```

```
    }
```

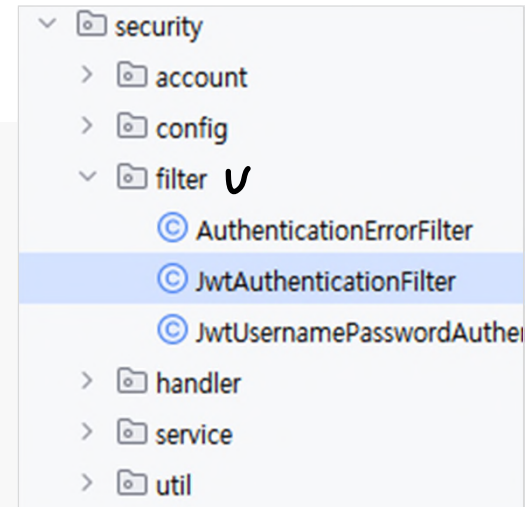
jwt 파싱할때 예외발생 가능. 특히 만기 예외

usernotfount 런타임 예외가 발생할 수 있죠?

귀한목록

o 모든 요청에 대해서 헤더에 토큰이 있는지 검사

- Authorization 헤더 항목 조사
 - Authorization: Bearer <jwt 토큰 문자열>
 - 토큰 추출 클라에서도 맞춰줘야함



토큰을 통해 로그인 사용자로면은 security context에 authentication을 등록하는것이 목적.

사용자 인증

토큰은 request에 담겨있쥬

security.filter.JwtAuthenticationFilter.java

해당필터는 모든 요청에 대해서 다 반응함
JWT있냐 없냐 에 대해서 체크

@Override ~~onceperrequestfilter~~메소드 오버라이드

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)

throws ServletException, IOException {

String bearerToken = request.getHeader(AUTHORIZATION_HEADER);

null이면 로그인을 안한 사용자 == anonymous
계정이 없는 것이 아닌 anonymous계정을 부여하여
계정이 있는지 없는지 확인할 필요가 없어진다. 모든 사용자가
계정을 가지니까. 일단 지금 실습에서는 anonymous계정부여는
하지 않는다.

if (bearerToken != null && bearerToken.startsWith(BEARER_PREFIX)) {

원하는 접두어 있는지

String token = bearerToken.substring(BEARER_PREFIX.length()); 접두어 분리

// 토큰에서 사용자 정보 추출 및 Authentication 객체 구성 후 SecurityContext에 저장

Authentication authentication = ~~getAuthentication(token)~~; 토큰을 통해서 아까말한 authentication을 얻고

SecurityContextHolder.getContext().setAuthentication(authentication); security context에다가 얻은 authentication을 등록!

super.doFilter(request, response, filterChain);

} 그 다음에 토큰 있든 없든 다음 필터로

}

JWT
처리

예외발생 가능

앞에 예외처리 필터를
넣고 처리하게끔하자.

반대로 getAuthentication으로
정보를 얻어낼 수 있다.

security.config.SecurityConfig.java 필터를 등록시키자

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final UserDetailsService userDetailsService;
```

생성자 주입되도록

```
private final JwtAuthenticationFilter jwtAuthenticationFilter;
```

@Autowired 해당 필터는 final 붙이면 에러가 난기 때문에 멤버주입 이용

```
private JwtUsernamePasswordAuthenticationFilter jwtUsernamePasswordAuthenticationFilter;
```

```
...
```

```
@Override
```

```
public void configure(HttpSecurity http) throws Exception {
```

```
    // 한글 인코딩 필터 설정
```

```
    http.addFilterBefore(encodingFilter(), CsrfFilter.class)
```

```
    // Jwt 인증 필터
```

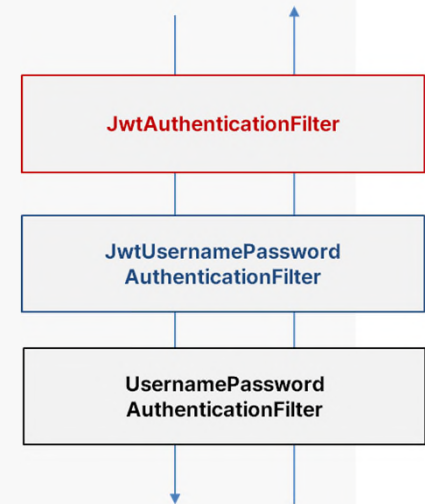
```
    .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class)
```

```
    // 로그인 인증 필터
```

```
    .addFilterBefore(jwtUsernamePasswordAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);
```

```
...
```

```
}
```



필터 동작 순서 지정 추가

해당과정에서는 refresh token을 운용하지 않는다.
너무 복잡

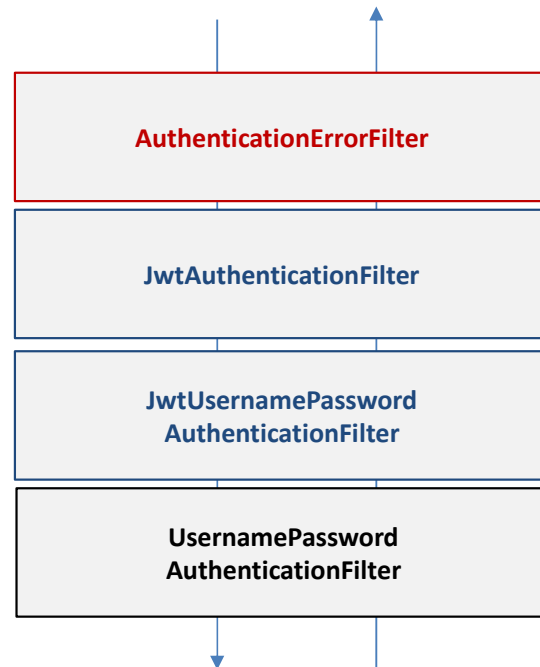
사용자 인증

✓ 인증 예외 처리

○ 토큰 처리시 예외 발생 가능 ✓

- 여러 예외 중 토큰 만기 예외인 경우, 401 인증 에러로 리턴 ✓
- 그 외의 예외는 일반 예외 메시지로 처리

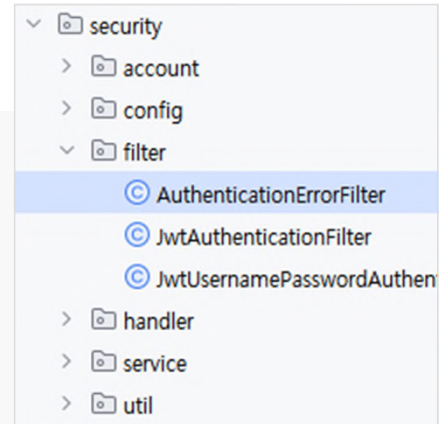
client는 재발급 과정을 거친다. refresh token 이용한



앞서 말한 에러처리 필터

security.filter.AuthenticationErrorFilter.java

```
package org.scoula.security.filter;
...
import io.jsonwebtoken.security.SignatureException;
...
@Component
public class AuthenticationErrorFilter extends OncePerRequestFilter {
    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
        throws ServletException, IOException {
        try {
            super.doFilter(request, response, filterChain); ✓
        } catch (ExpiredJwtException e) {
            401
            JsonResponse.sendError(response, HttpStatus.UNAUTHORIZED, "토큰의 유효시간이 지났습니다.");
        } catch (UnsupportedJwtException | MalformedJwtException | SignatureException e) {
            JsonResponse.sendError(response, HttpStatus.UNAUTHORIZED, e.getMessage());
        } catch (ServletException e) { 나머지
            JsonResponse.sendError(response, HttpStatus.INTERNAL_SERVER_ERROR, e.getMessage());
        }
    }
}
```

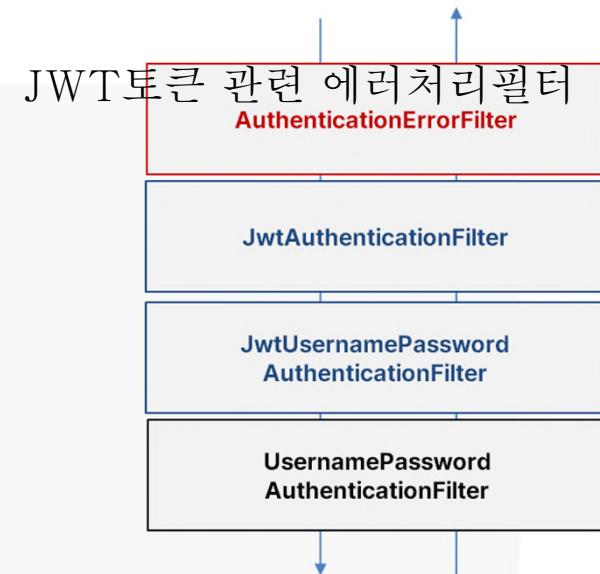


→ 갱신절차 진행해야겠죠

security.config.SecurityConfig.java

필터 등록!

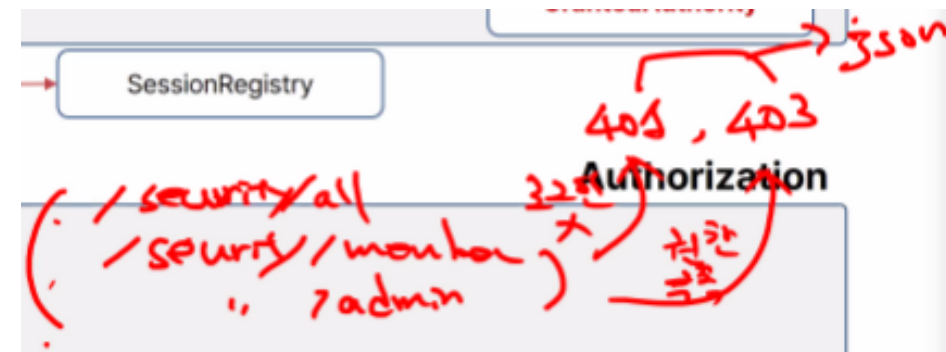
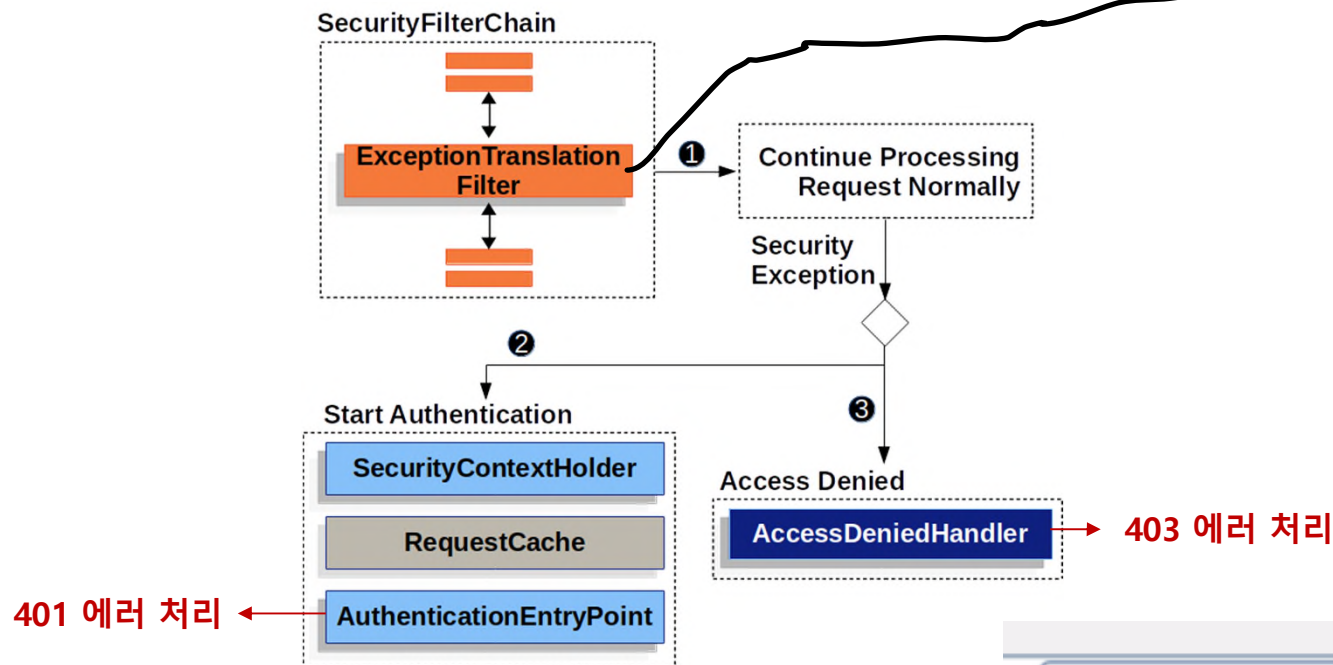
```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final UserDetailsService userDetailsService;
    private final JwtAuthenticationFilter jwtAuthenticationFilter;
    private final AuthenticationErrorFilter authenticationErrorFilter;
    @Autowired
    private JwtUsernamePasswordAuthenticationFilter jwtUsernamePasswordAuthenticationFilter;
    ...
    @Override
    public void configure(HttpSecurity http) throws Exception {
        // 한글 인코딩 필터 설정
        http.addFilterBefore(encodingFilter(), CsrfFilter.class)
        // 인증 에러 필터
        .addFilterBefore(authenticationErrorFilter, UsernamePasswordAuthenticationFilter.class)
        // Jwt 인증 필터
        .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class)
        // 로그인 인증 필터
        .addFilterBefore(jwtUsernamePasswordAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);
        ...
    }
}
```



✓ 인증/인가 에러 처리

- 디폴트 401, 403 처리는 에러 페이지로 응답
- JSON 응답 처리로 변경해야 함
- AuthenticationEntryPoint, AccessDeniedHandler 커스텀마이징

권한, 로그인 관련 에러 처리 필터

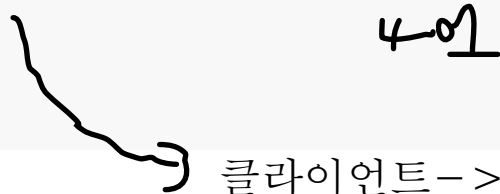
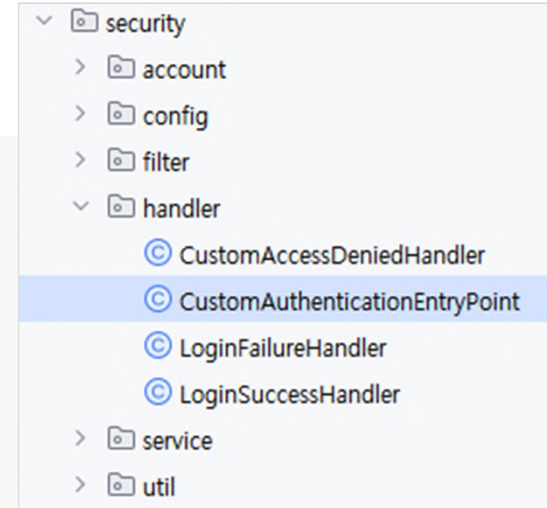


security.handler.CustomAuthenticationEntryPoint.java

```
package org.scoula.security.handler;
...

@Log4j2
@Component
public class CustomAuthenticationEntryPoint implements AuthenticationEntryPoint {

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response, AuthenticationException authException)
        throws IOException, ServletException {
        log.error("===== 인증 에러 =====");
        JsonResponse.sendError(response, HttpStatus.UNAUTHORIZED, authException.getMessage());
    }
}
```



클라이언트 -> login 페이지로 이동 -> 로그인 성공 -> 가고자한 페이지로 이동

/security/member가려다가 로그인 실패
그럼 로그인 페이지로 이동하고
성공하면
위래 가려했던 /sercurity/member로 이동시켜야함

이건 프론트. vue에서 진행해야함

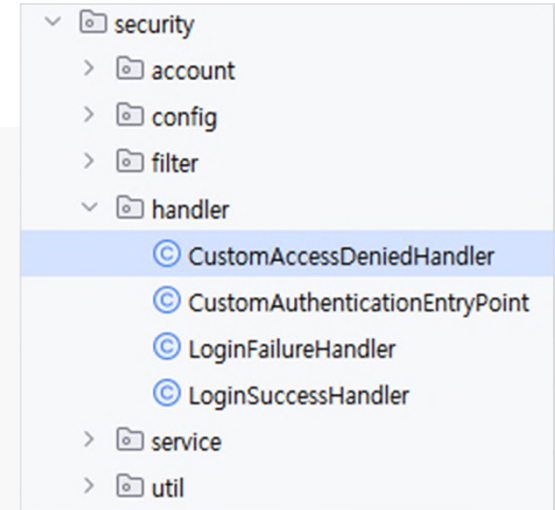
CustomAccessDeniedHandler.java

```
package org.scoula.security.handler;
...

@Component
@Log4j2
public class CustomAccessDeniedHandler implements AccessDeniedHandler {

    @Override
    public void handle(HttpServletRequest request, HttpServletResponse response,
        AccessDeniedException accessDeniedException) throws IOException, ServletException {
        log.error("===== 인가 에러 =====");
        JsonResponse.sendError(response, HttpStatus.FORBIDDEN, "권한이 부족합니다.");
    }
}
```

↳ 클라리언트 측 알리면 돼



security.config.SecurityConfig.java

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    ...
    private final JwtUsernamePasswordAuthenticationFilter jwtUsernamePasswordAuthenticationFilter;

    {
        private final CustomAccessDeniedHandler accessDeniedHandler;
        private final CustomAuthenticationEntryPoint authenticationEntryPoint;
    }
    ...
    @Override
    public void configure(HttpSecurity http) throws Exception {
        // 한글 인코딩 필터 설정
        http.addFilterBefore(encodingFilter(), CsrfFilter.class)
        ...;
        // 예외 처리 설정
        {
            http
                .exceptionHandling()
                .authenticationEntryPoint(authenticationEntryPoint)
                .accessDeniedHandler(accessDeniedHandler);
        }
        ...
    }
}
```

✓ 테스트

- `/api/security/all`
 - 인증 없이 접근 가능
- `/api/security/member`
 - `ROLE_MEMBER`가 있어야 접근 가능, `ROLE_ADMIN`도 포함
- `/api/security/admin`
 - `ROLE_MANAGER`가 있어야 접근 가능
admin
- 토큰 유효 시간을 10분으로 설정
 - `JwtProcessor`
 - `TOKEN_VALID_MILLISECOND = 1000L * 60 * 10;`

controller.SecurityController.java

변경

```
package org.scoula.controller;
...
```

@Log4j2

@RequestMapping("/api/security") ✓

@RestController ✓ JSON을 주고 받으니까. rest기반 api 하니까!

```
public class SecurityController {
```

```
    @GetMapping("/all")
```

```
    public ResponseEntity<String> doAll() {
```

```
        log.info("do all can access everybody");
```

```
        return ResponseEntity.ok("All can access everybody");
```

```
    }
```

```
    @GetMapping("/member")
```

```
    public ResponseEntity<String> doMember(Authentication authentication) {
```

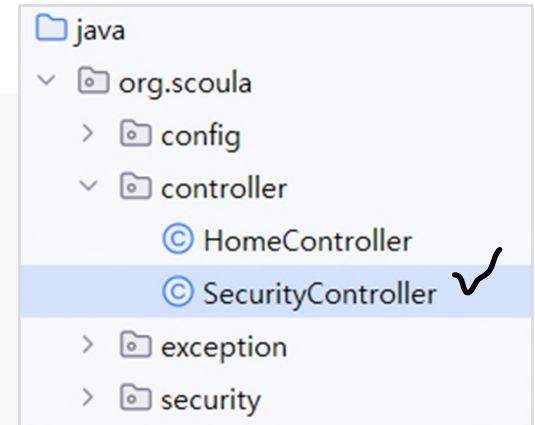
```
        UserDetails userDetails = (UserDetails)authentication.getPrincipal();
```

```
        log.info("username = " + userDetails.getUsername());
```

```
        return ResponseEntity.ok(userDetails.getUsername());
```

```
    }
```

security context에서 가져온 걸 인자로 넣어주겠쥬



controller.SecurityController.java

```
@GetMapping("/admin")
public ResponseEntity<MemberVO> doAdmin(@AuthenticationPrincipal CustomUser customUser) {
    MemberVO member = customUser.getMember();
    log.info("username = " + member);
    return ResponseEntity.ok(member);
}
}
```

401 403 에러가 일어날 수 있음

security.config.SecurityConfig.java

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

```
...
```

```
@Override
```

```
public void configure(HttpSecurity http) throws Exception {
```

```
...
```

```
http
```

```
.authorizeRequests() // 경로별 접근 권한 설정
```

```
.antMatchers(HttpMethod.OPTIONS).permitAll()
```

```
.antMatchers("/api/security/all").permitAll()
```

```
// 모두 허용
```

```
.antMatchers("/api/security/member").access("hasRole('ROLE_MEMBER')") // ROLE_MEMBER 이상 접근 허용
```

```
.antMatchers("/api/security/admin").access("hasRole('ROLE_ADMIN')") // ROLE_ADMIN 이상 접근 허용
```

```
.anyRequest().authenticated(); // 나머지는 로그인 된 경우 모두 허용
```

```
}
```

```
...
```

```
} 주의사항  
마지막에 .anyRequest().permitAll() 하면 제대로 동작안함
```

✓ 로그인 안한 상태

- <http://localhost:8080/api/security/all>

The screenshot shows a web browser's developer tools interface. At the top, the 'METHOD' dropdown is set to 'GET' (marked with a checkmark), and the 'URL' field contains 'http://localhost:8080/api/security/all'. A 'Send' button is visible next to the URL. Below the URL, the 'QUERY PARAMETERS' section is empty. The 'HEADERS' section is on the left, and the 'BODY' section is on the right. The 'BODY' section shows a message: 'XHR does not allow payloads for GET request.' Below this, the 'Response' section is displayed, showing a status of '200' (highlighted in green) and a message 'Cache Detected - Elapsed Time: 5ms'. The 'Response' section is divided into 'HEADERS' and 'BODY' tabs. The 'HEADERS' tab is selected, showing the following headers: 'X-Content-Type: nosniff', 'X-XSS-Protection: 1; mode=block', 'Cache-Control: no-cache, no-store, max-age=0, must-revalidate', 'Pragma: no-cache', and 'Expires: 0'. The 'BODY' tab is also visible, showing the text 'All can access everybody' and a 'copy' button. The 'length: 24 bytes' is indicated at the bottom right of the body section.

✓ 로그인 안한 상태

- <http://localhost:8080/api/security/member>

The screenshot shows a web client interface with the following details:

- Request:**
 - METHOD:** GET
 - URL:** `http://localhost:8080/api/security/member` (length: 41 byte(s))
 - QUERY PARAMETERS:** (empty)
 - HEADERS:** (empty)
 - BODY:** XHR does not allow payloads for GET request.
- Response:**
 - Status:** 401
 - Cache:** Cache Detected - Elapsed Time: 4ms
 - HEADERS:**
 - X-Content-Ty... nosniff
 - X-XSS-Protec... 1; mode=block
 - Cache-Contro... no-cache, no-store, max-age=0, must-revalidate
 - Pragma: no-cache
 - Expires: 0
 - X-Frame-Opti... DENY
 - BODY:**
 - Unexpected character (F) at position 0
 - Full authentication is required to access this resource
 - Footer:** lines nums, copy, length: 55 bytes

✓ 로그인

- <http://localhost:8080/api/auth/login>

The screenshot shows a REST client interface with the following details:

- METHOD:** POST (checked with a handwritten checkmark)
- URL:** `http://localhost:8080/api/auth/login` (checked with a handwritten checkmark)
- Content-Type:** `application/json` (checked with a checkbox)
- Body:**

```
{
  "username": "user0",
  "password": "1234"
}
```

 (checked with a handwritten checkmark)
- Response:** 200 (status bar)
- Response Headers:** `X-Content-Type: nosniff`, `X-XSS-Protection: 1; mode=block`, `Cache-Control: no-cache, no-store, max-age=0, must-revalidate`, `Pragma: no-cache`, `Expires: 0`, `X-Frame-Options: DENY`, `Content-Type: application/json; charset=UTF-8`, `Transfer-Encoding: chunked`, `Date: Thu, 25 Jul 2024 07:45:31 GMT`, `Keep-Alive: timeout=20`, `Connection: keep-alive`
- Response Body:**

```
{
  token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ5IiwiaWF0Ij0iMTY3MjUwMDAwMCJ9,
  userInfo: {
    name: "user0",
    email: "user0@galapagos.org",
    roles: [
      "ROLE_MANAGER",
      "ROLE_MEMBER"
    ]
  }
}
```

A blue callout box with the text "토큰값 복사" (Copy token value) points to the `token` field in the response body, which is highlighted with a red box. A handwritten checkmark is next to the response body.

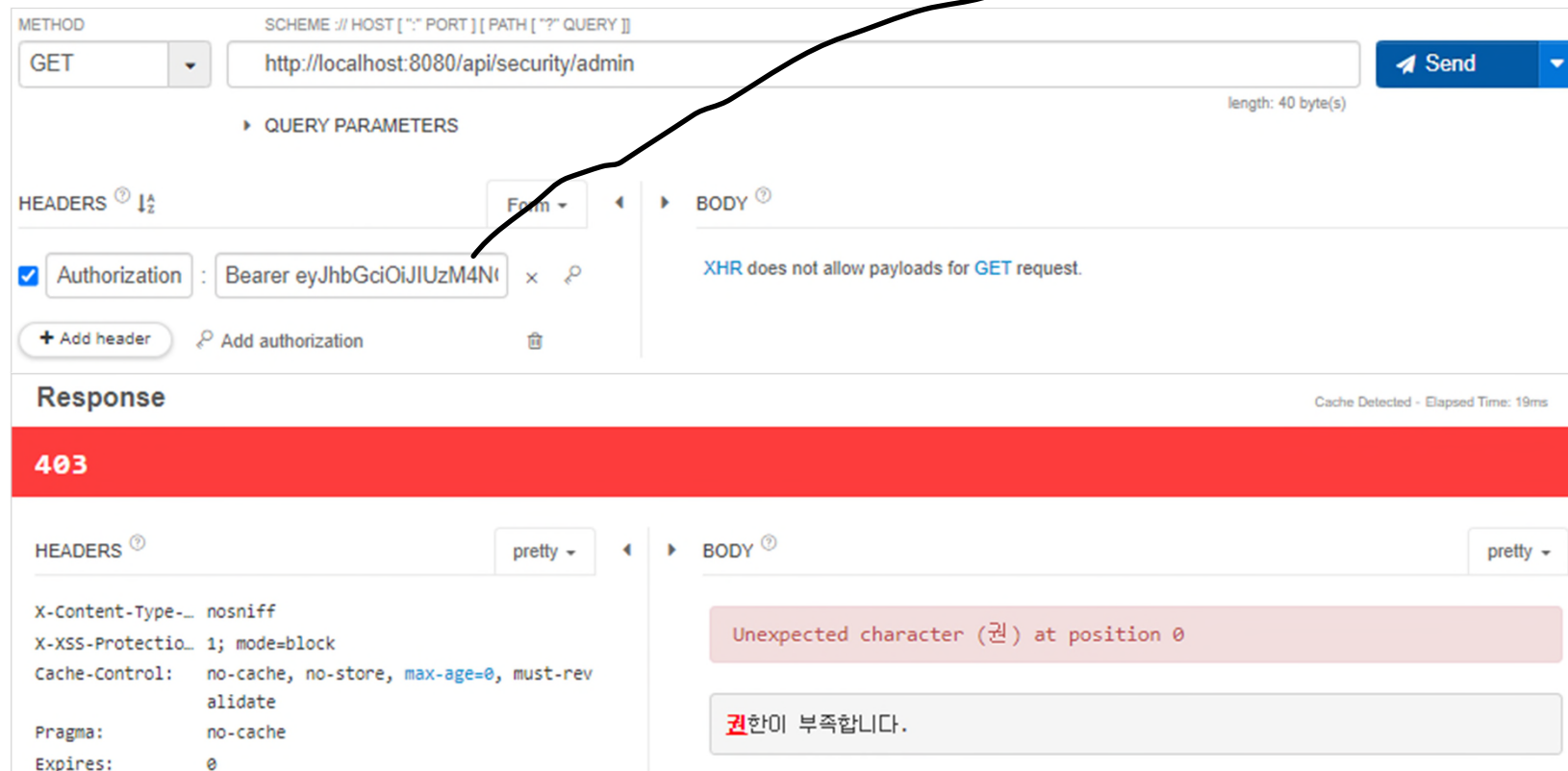
- <http://localhost:8080/api/security/member>
- 요청 헤더에 토큰 직접 설정 Authorization: Bearer 토큰_문자열

22

✓ 로그인 상태(user0, ROLE_MANAGER)

jwt authentic filter가 분석

- <http://localhost:8080/api/security/admin>
- 요청 헤더에 토큰 직접 설정 Authorization: Bearer 토큰_문자열



METHOD: GET
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]
http://localhost:8080/api/security/admin
length: 40 byte(s)
Send

QUERY PARAMETERS

HEADERS
Authorization: Bearer eyJhbGciOiJIUzI4NiIsInR5cCI6IkpXLT...

BODY
XHR does not allow payloads for GET request.

Response
Cache Detected - Elapsed Time: 19ms

403

HEADERS
X-Content-Type-...: nosniff
X-XSS-Protection...: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0

BODY
Unexpected character (권) at position 0
권한이 부족합니다.

✓ 로그인 상태(admin, ROLE_ADMIN)

- <http://localhost:8080/api/security/admin>
- 요청 헤더에 토큰 직접 설정 Authorization: **Bearer** 토큰_문자열

METHOD: GET | SCHEME // HOST [":" PORT] [PATH ["?" QUERY]] | length: 40 byte(s)

QUERY PARAMETERS

HEADERS: Authorization: Bearer eyJhbGciOiJIUzI4NCJ9.e

BODY: XHR does not allow payloads for GET request.

Response: 200 | Cache Detected - Elapsed Time: 40ms

HEADERS: X-Content-Type-Opt_ nosniff, X-XSS-Protection: 1; mode=block, Cache-Control: no-cache, no-store, max-age=0, must-revalidate, Pragma: no-cache, Expires: 0

BODY: { username: "admin", password: "\$2a\$10\$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/1ddCyn0nic5dFo3VfJYrXYC", email: "admin@galapagos.org", regDate: 1721790112000 }

✅ 로그인 상태(admin, ROLE_ADMIN)

- <http://localhost:8080/api/security/admin>
- 요청 헤더에 잘못된 토큰 직접 설정 Authorization: Bearer 토큰_문자열x

METHOD

GET

Scheme // Host [Port] Path [Query]

http://localhost:8080/api/security/admin

length: 40 byte(s)

Send

QUERY PARAMETERS

HEADERS

Authorization : Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTQwMjYyLWVudC9e

+ Add header

Add authorization

BODY

XHR does not allow payloads for GET request.

Response

Cache Detected - Elapsed Time: 19ms

401

HEADERS

pretty

X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0

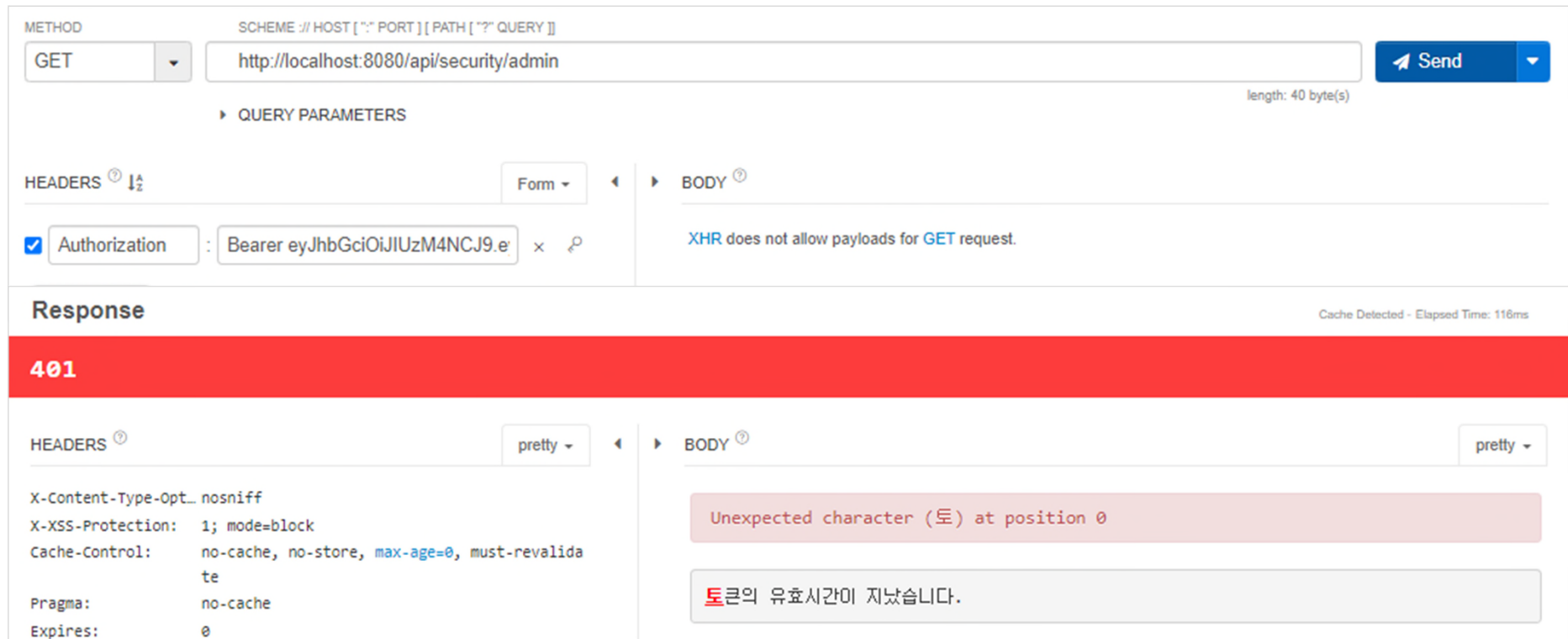
BODY

pretty

Unexpected character (M) at position 0

Malformed JWT JSON: {"alg":"HS31

- ✓ 10분 경과, 토큰 유효 시간 이후 접근
 - <http://localhost:8080/api/security/admin>



METHOD: GET
SCHEME // HOST [":" PORT] [PATH ["?" QUERY]]
`http://localhost:8080/api/security/admin`
length: 40 byte(s)
Send

QUERY PARAMETERS

HEADERS 1/2
Authorization : Bearer eyJhbGciOiJIUzI4NCJ9.e

BODY
XHR does not allow payloads for GET request.

Response
Cache Detected - Elapsed Time: 116ms

401

HEADERS pretty
X-Content-Type-Opt... nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalida
te
Pragma: no-cache
Expires: 0

BODY pretty
Unexpected character (토) at position 0
토큰의 유효시간이 지났습니다.

보안은 잘 만든 다음에 필요한 부분만 커스터마이징하는 방식으로 프로젝트에 적용하면 된다!!!!