

KB금융그룹



2025년 상반기 K-디지털 트레이닝

# 사용자 인증

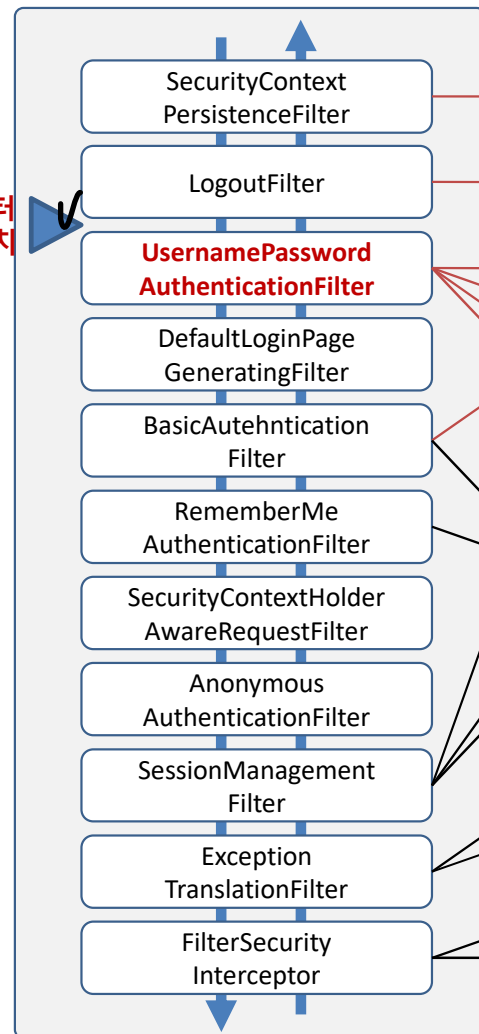
[KB] IT's Your Life

우리의 jwt메카니즘을 넣어서 커스터마이징 하자  
이 방법도 여러가지다.

최대한 spring security filter chain에  
맞게 커스터마이징 하자

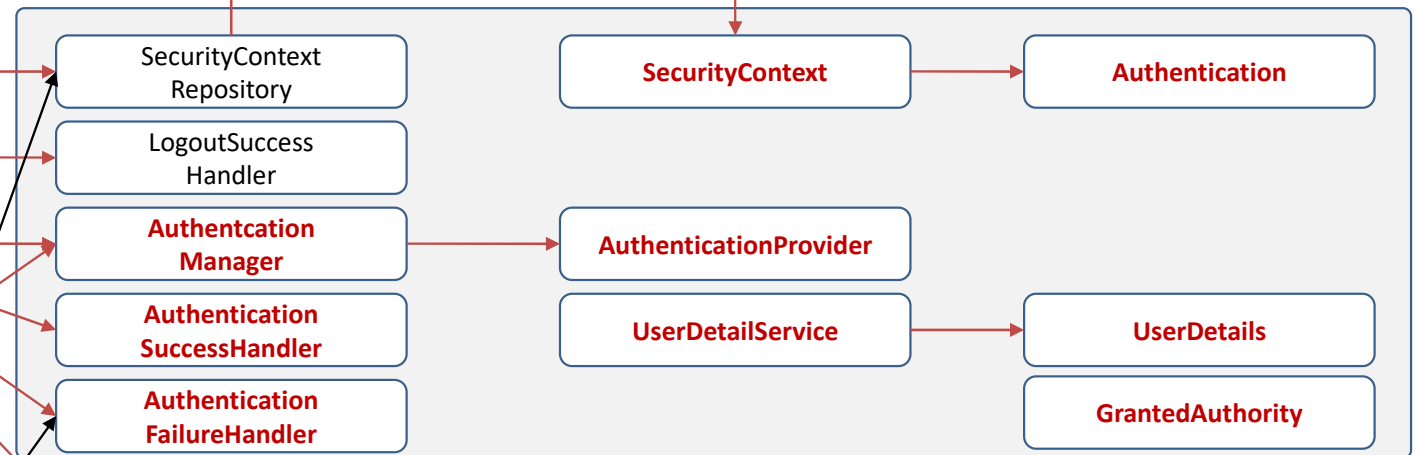
U  
Jwt 필터  
추가위치

## SecurityFilterChain

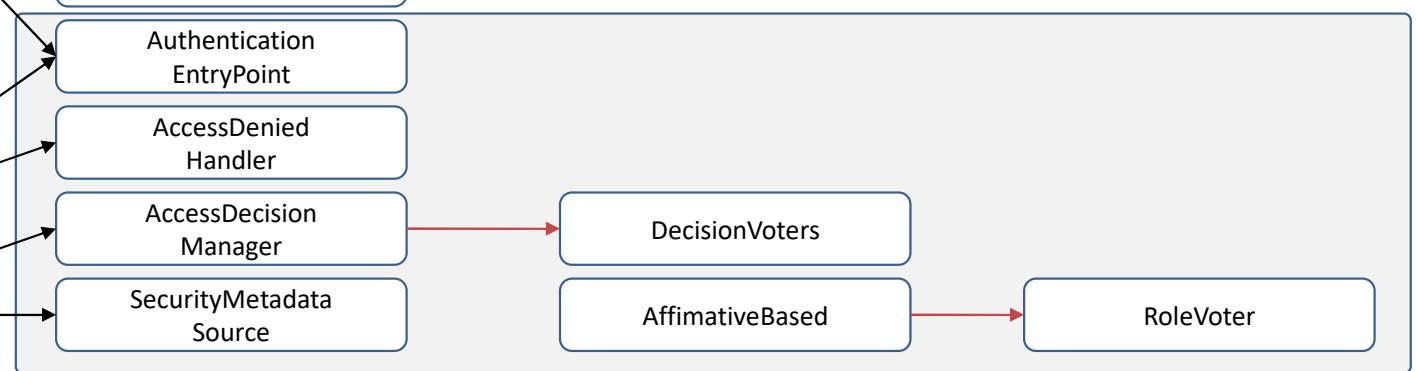


## SecurityContextHolder

## Authentication



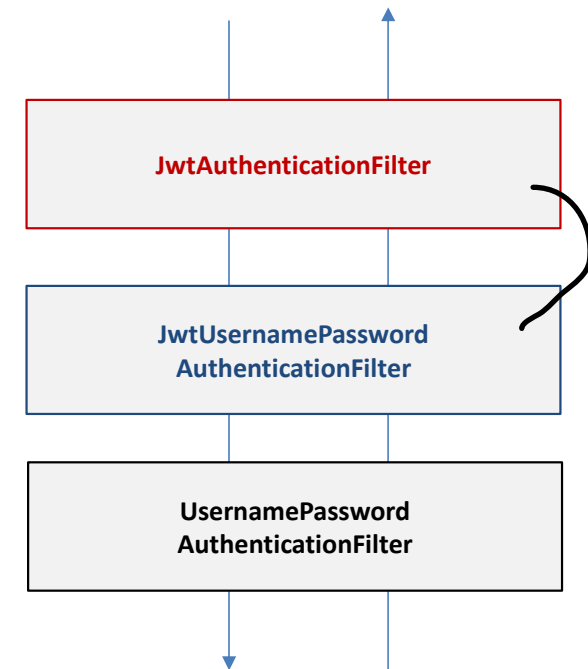
## Authorization



### ✓ JwtAuthenticationFilter

- 모든 요청에 대해서 헤더에 토큰이 있는지 검사
  - Authorization 헤더 항목 조사
    - Authorization: Bearer <jwt 토큰 문자열>
    - 토큰 추출
- 토큰의 유효성 검사
  - 토큰이 유효하면 SecurityContextHolder에 사용자 로그인 정보 설정
- 다음 필터로 이동

→ 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

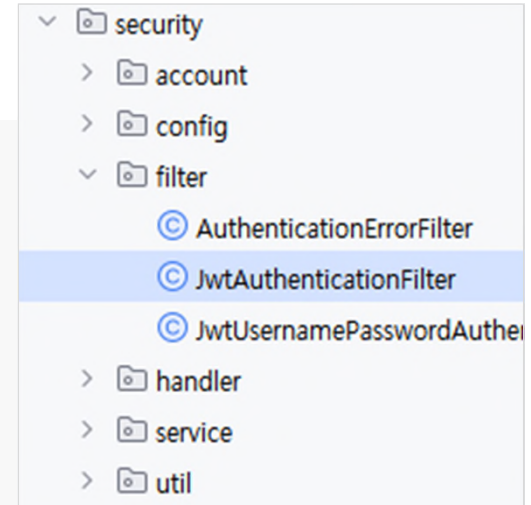
## security.filter.JwtAuthenticationFilter.java

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

@Component
@Log4j2
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    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;

    private Authentication getAuthentication(String token) {
        String username = jwtProcessor.getUsername(token);
        UserDetails principl = userDetailsService.loadUserByUsername(username);
        return new UsernamePasswordAuthenticationToken(princiapl, null, principl.getAuthorities());
    }
}
```



### security.filter.JwtAuthenticationFilter.java

```
@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
    throws ServletException, IOException {

    String bearerToken = request.getHeader(AUTHORIZATION_HEADER);

    if (bearerToken != null && bearerToken.startsWith(BEARER_PREFIX)) {
        String token = bearerToken.substring(BEARER_PREFIX.length());

        // 토큰에서 사용자 정보 추출 및 Authentication 객체 구성 후 SecurityContext에 저장
        Authentication authentication = getAuthentication(token);
        SecurityContextHolder.getContext().setAuthentication(authentication);
    }

    super.doFilter(request, response, filterChain);
}
```

## security.config.SecurityConfig.java

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

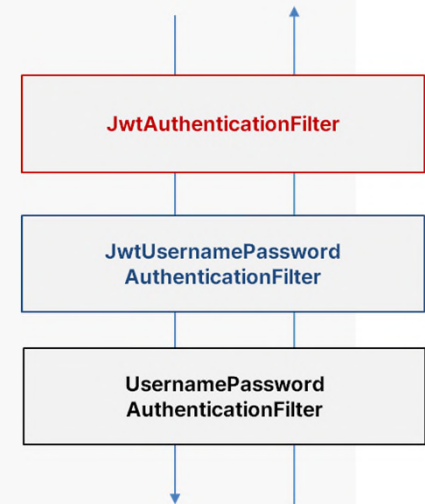
    private final JwtAuthenticationFilter jwtAuthenticationFilter;

    @Autowired
    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);

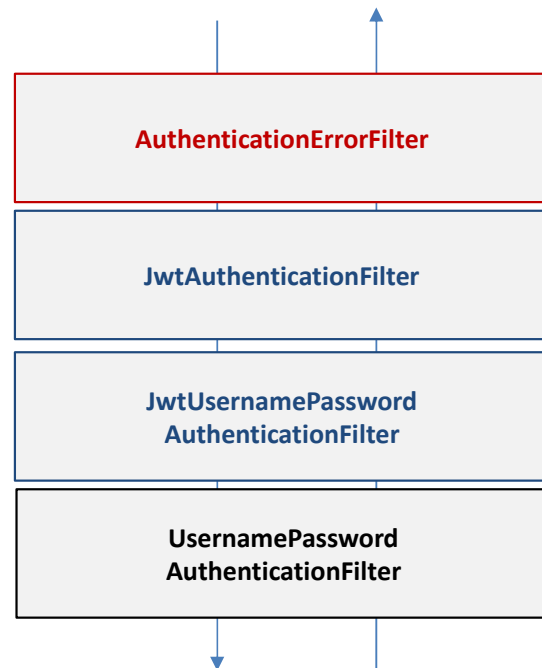
        ...
    }
}
```



### ✓ 인증 예외 처리

#### ○ 토큰 처리시 예외 발생 가능

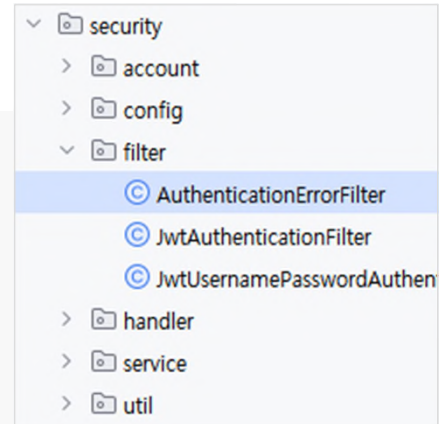
- 여러 예외 중 토큰 만기 예외인 경우, 401 인증 에러로 리턴
- 그 외의 예외는 일반 예외 메시지로 처리





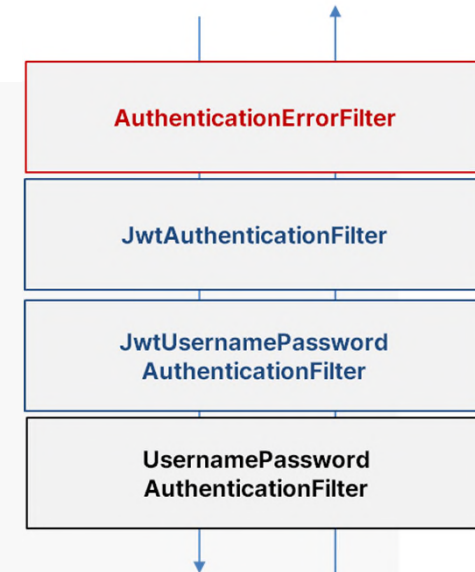
## security.filter.AuthenticationErrorFilter.java

```
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) {
            JsonResponse.sendError(response, HttpStatus.UNAUTHORIZED, "토큰의 유효시간이 지났습니다.");
        } catch (UnsupportedJwtException | MalformedJwtException | SignatureException e) {
            JsonResponse.sendError(response, HttpStatus.UNAUTHORIZED, e.getMessage());
        } catch (ServletException e) {
            JsonResponse.sendError(response, HttpStatus.INTERNAL_SERVER_ERROR, e.getMessage());
        }
    }
}
```



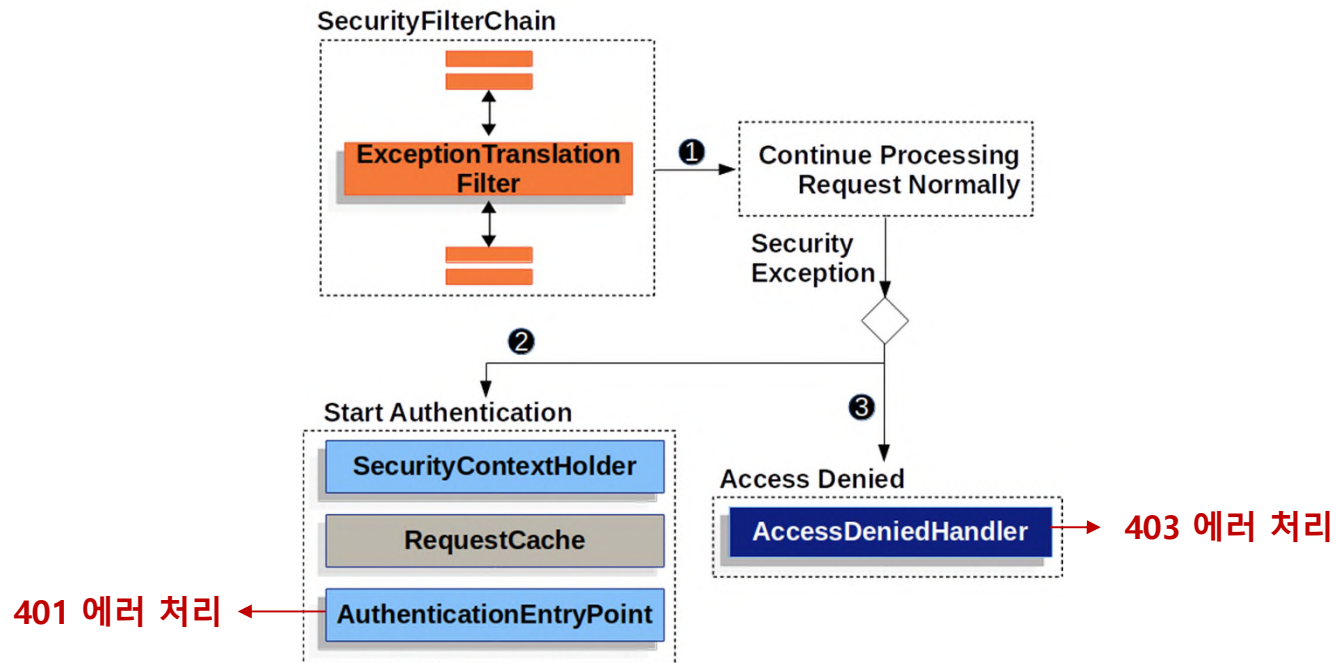
## security.config.SecurityConfig.java

```
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



## 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)
```

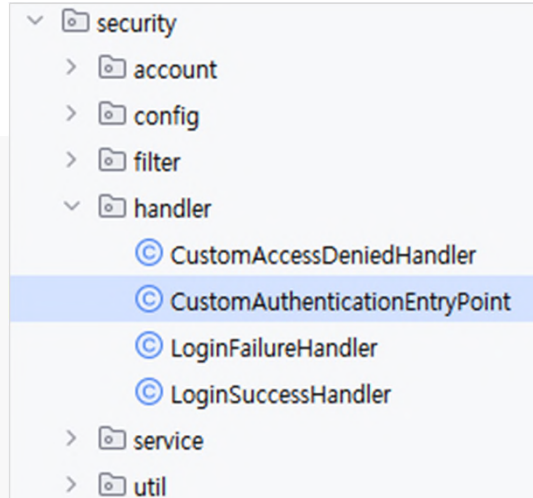
```
        throws IOException, ServletException {
```

```
        log.error("===== 인증 에러 =====");
```

```
        JsonResponse.sendError(response, HttpStatus.UNAUTHORIZED, authException.getMessage());
```

```
    }
```

```
}
```

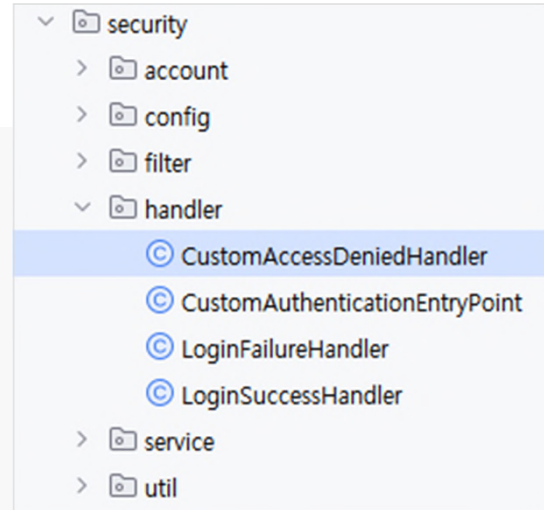


## CustomAccessDeniedHandler.java

```
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.");
    }
}
```



### 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
- `/api/security/admin`
  - `ROLE_MANAGER`가 있어야 접근 가능
- 토큰 유효 시간을 10분으로 설정
  - `JwtProcessor`
    - `TOKEN_VALID_MILLISECOND = 1000L * 60 * 10;`

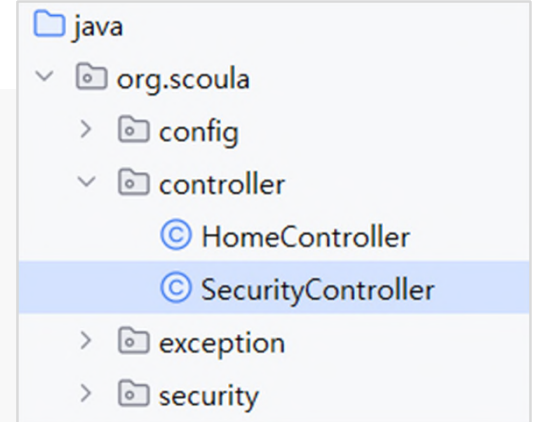
### controller.SecurityController.java

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

@Log4j2
@RequestMapping("/api/security")
@RestController
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());
    }
}
```





### controller.SecurityController.java

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

### 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  
    }  
    ...  
}
```

### ✓ 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' and the URL bar shows 'http://localhost:8080/api/security/all'. A 'Send' button is visible next to the URL. Below the URL bar, the 'QUERY PARAMETERS' section is expanded. The 'HEADERS' section is also expanded, showing a 'Form' dropdown and buttons for '+ Add header' and 'Add authorization'. The 'BODY' section is expanded, showing a message: 'XHR does not allow payloads for GET request.' Below the request section, the 'Response' section is expanded, showing a status code of '200' and a message: 'All can access everybody'. The 'HEADERS' section of the response is expanded, showing 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' section of the response is expanded, showing the text 'All can access everybody' and a 'copy' button. The 'length: 24 bytes' is also displayed.

METHOD: GET | SCHEME :// HOST [ ":" PORT ] [ PATH [ "?" QUERY ] ] | length: 38 byte(s)

Send request (Alt + Enter)

QUERY PARAMETERS

HEADERS ? | Form | BODY ?

+ Add header | Add authorization

XHR does not allow payloads for GET request.

Response | Cache Detected - Elapsed Time: 5ms

200

HEADERS ? | pretty | BODY ? | raw

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

All can access everybody

copy | length: 24 bytes

### ✓ 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
  - length:** 55 bytes

## ✓ 로그인

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

The screenshot displays a REST client interface for a POST request to `http://localhost:8080/api/auth/login`. The request body is a JSON object: `{ "username": "user0", "password": "1234" }`. The response is a 200 OK status with a JSON body containing a token and user information. A blue callout box labeled "토큰값 복사" (Copy token value) points to the token field in the response body.

**Request:**

- METHOD: POST
- URL: `http://localhost:8080/api/auth/login`
- Content-Type: `application/json`
- Body: 

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

**Response:**

- Status: 200
- 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`
- Body: 

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vybm91dCI6Imlhbmh",
  "userInfo": {
    "name": "user0",
    "email": "user0@galapagos.org",
    "roles": [
      "ROLE_MANAGER",
      "ROLE_MEMBER"
    ]
  }
}
```

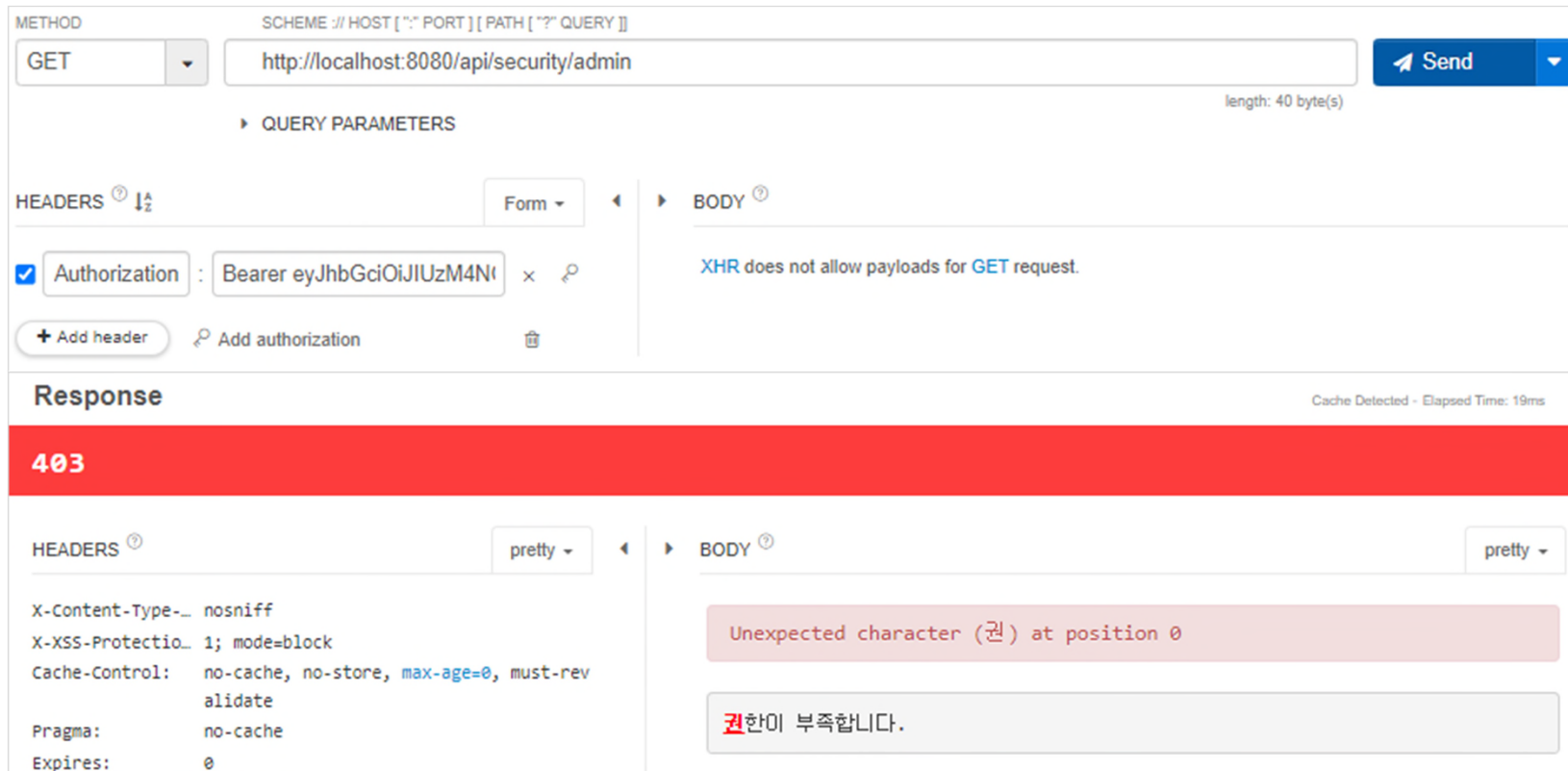
## ✅ 로그인 상태(user0, ROLE\_MANAGER)

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

The screenshot displays the Swagger UI interface for a REST API. The top section shows the request configuration for a GET method to the endpoint `http://localhost:8080/api/security/member`. The request is configured with the Authorization header set to `Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTUxMjM0NTY3ODk0IiwiaWF0IjoxNjU0MjM0NTY3ODk0fQ`. The response is a 200 status code with headers including `X-Content-Type-Options: nosniff`, `X-XSS-Protection: 1; mode=block`, and `Cache-Control: no-cache, no-store, max-age=0, must-revalidate`. The body of the response is `user0`.

### ✓ 로그인 상태(user0, ROLE\_MANAGER)

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



METHOD: GET, URL: http://localhost:8080/api/security/admin, length: 40 byte(s)

HEADERS: Authorization: Bearer eyJhbGciOiJIUzI4NiIsInR5cCI6IkpXLT...

Response: 403, Cache Detected - Elapsed Time: 19ms

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)

Send

QUERY PARAMETERS

HEADERS: Authorization: Bearer eyJhbGciOiJIUzI4NCJ9.e

Form

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

pretty

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, URL: http://localhost:8080/api/security/admin, length: 40 byte(s)

HEADERS: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cGU6IjY9e. (Token value is highlighted with a blue box and text: 토큰값 임의로 수정)

BODY: XHR does not allow payloads for GET request.

Response: 401, Cache Detected - Elapsed Time: 19ms

HEADERS: X-Content-Type-Opt\_ nosniff, X-XSS-Protection: 1; mode=block, Cache-Control: no-cache, no-store, max-age=0, must-revalidat e, Pragma: no-cache, Expires: 0

BODY: Unexpected character (M) at position 0, Malformed JWT JSON: {"alg":"HS31"}

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

26