KB금융그룹 | 국민의 평생 금융파트너

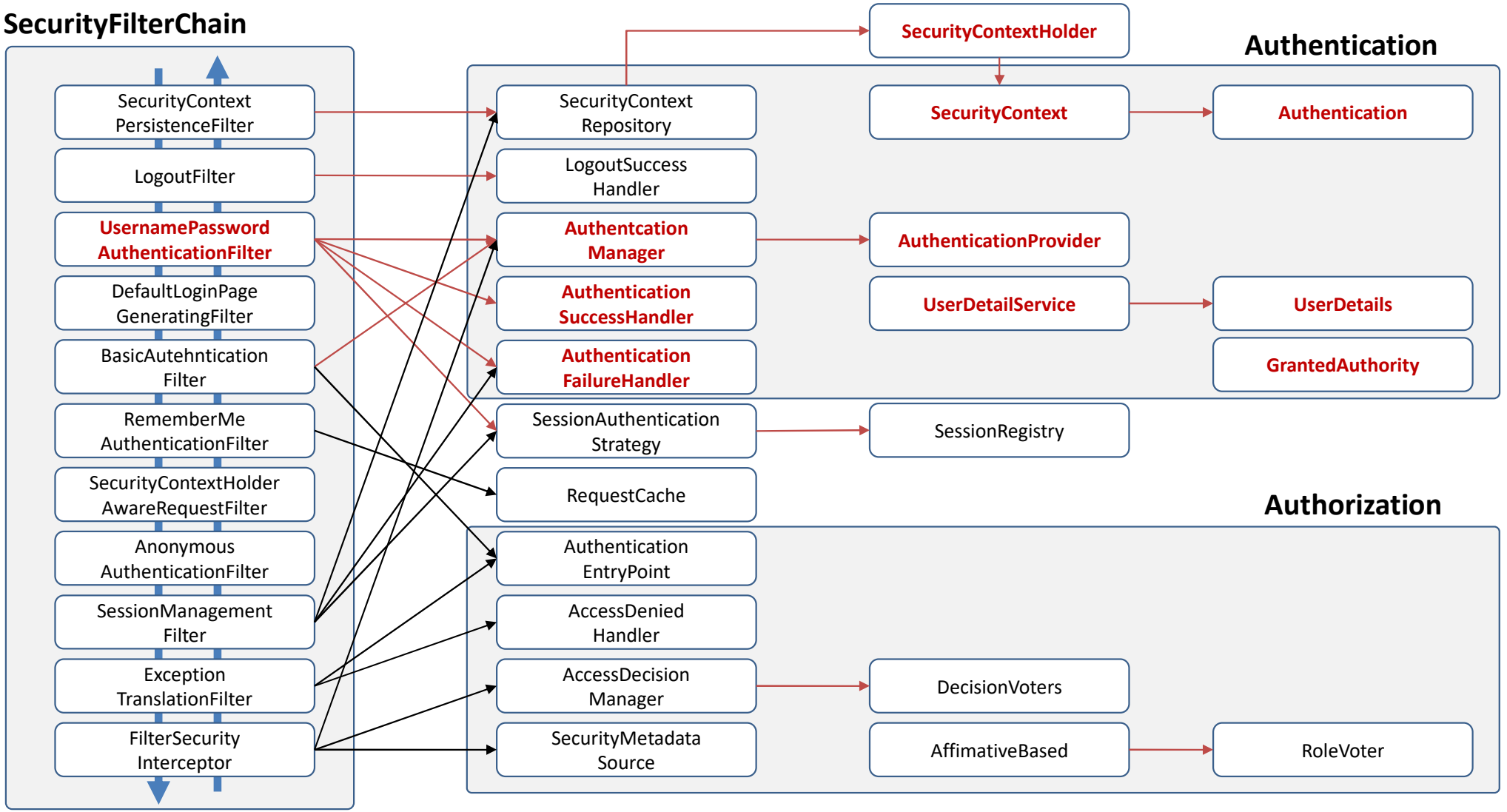
2025년 상반기 K-디지털 트레이닝

API 로그인

[KB] IT's Your Life

KB 국민은행

API 로그인


SecurityFilterChain
SecurityContext PersistenceFilter
LogoutFilter
UsernamePassword AuthenticationFilter
DefaultLoginPage GeneratingFilter
BasicAutehntication Filter
RememberMe AuthenticationFilter
SecurityContextHolder AwareRequestFilter
Anonymous AuthenticationFilter
SessionManagement Filter
Exception TranslationFilter
FilterSecurity Interceptor
SecurityContextHolder
Authentication
SecurityContext Repository
LogoutSuccess Handler
Authentcation Manager
Authentication SuccessHandler
Authentication FailureHandler
SessionAuthentication Strategy
RequestCache
SecurityContext
Authentication
AuthenticationProvider
UserDetailService
UserDetails
GrantedAuthority
SessionRegistry
Authorization
Authentication EntryPoint
AccessDenied Handler
AccessDecision Manager
SecurityMetadata Source
DecisionVoters
AffimativeBased
RoleVoter

API 로그인

LoginDTO

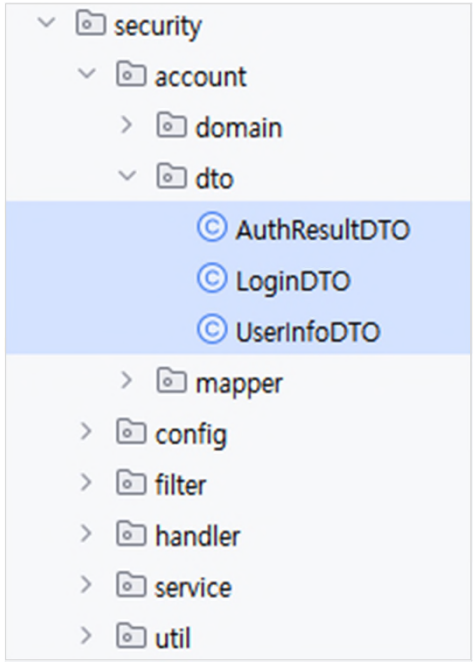
- Spring Security 규약에 따라 username, password 프로퍼티를 가짐
- Requuest body의 json 문자열에서 LoginDTO 객체로 역직렬화 직접 수행
  - Jackson 사용

UserInfoDTO

- 로그인 성공 시 응답에 포함시킬 사용자 정보
- token, user{ username, email, roles }포함

AuthResultDTO

- 로그인 성공 결과를 나타내는 응답
- 인증 token과 UserInfoDTO로 구성



security.account.dto.LoginDTO.java

```
package org.scoula.security.account.dto;
...

@NoArgsConstructor
@AllArgsConstructor
@Data
public class LoginDTO {
    private String username;
    private String password;

    public static LoginDTO of(HttpServletRequest request) {
        ObjectMapper om = new ObjectMapper();
        try {
            return om.readValue(request.getInputStream(), LoginDTO.class);
        }catch (Exception e) {
            e.printStackTrace();
            throw new BadCredentialsException("username 또는 password가 없습니다.");
        }
    }
}
```

security.account.dto.UserInfoDTO.java

```
package org.scoula.security.account.dto;
…
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserInfoDTO {
   String username;
   String email;
   List<String> roles;

   public static UserInfoDTO of(MemberVO member) {
      return new UserInfoDTO(
            member.getUsername(),
            member.getEmail(),
            member.getAuthList().stream()
                  .map(a-> a.getAuth())
                  .toList()
      );
   }
}
```

security.account.dto.AuthResultDTO.java

```
package org.scoula.security.account.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class AuthResultDTO {
    String token;
    UserInfoDTO user;
}
```

API 로그인

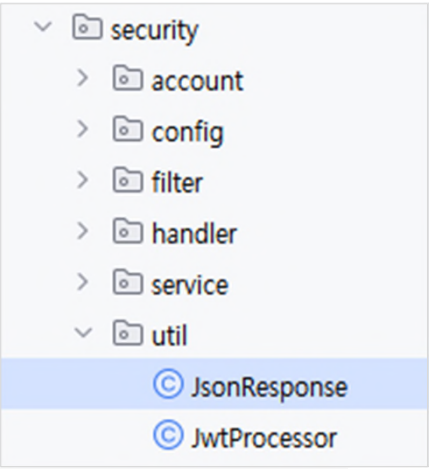
JsonResponse

- 로그인 결과를 필터에서 직접 Json 응답하기 위한 유틸리티 클래스

- static <T> void send(HttpServletResponse response, T result) throws IOException
  - Jackson으로 T를 직렬화 한 후 response로 직접 전송

- static void sendError(HttpServletResponse response, HttpStatus status, String message) throws IOException
  - 응답 코드와 에러 메시지를 출력

security.util.JsonResponse.java




```
package org.scoula.security.util;

import com.fasterxml.jackson.databind.ObjectMapper;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.Writer;

public class JsonResponse {
    public static <T> void send(HttpServletResponse response, T result) throws IOException {
        ObjectMapper om = new ObjectMapper();

        response.setContentType("application/json;charset=UTF-8");
        Writer out = response.getWriter();
        out.write(om.writeValueAsString(result));
        out.flush();
    }
```

security.util.JsonResponse.java

```
    public static void sendError(HttpServletResponse response, HttpStatus status, String message) throws IOException {
        response.setStatus(status.value());
        response.setContentType("application/json;charset=UTF-8");
        Writer out = response.getWriter();
        out.write(message);
        out.flush();
    }
}
```

API 로그인

API를 통한 로그인 절차

JwtUsernamePasswordAuthenticationFilter

- 로그인 url과 로그인 성공/실패 처리기를 등록

1. body에서 LoginDTO 추출
2. UsernamePasswordAuthenticationToken 준비
3. AuthenticationManager에게 인증 위임

AuthenticationManager의 인증 절차는 formLogin 때와 동일

인증 성공/실패 여부에 따라

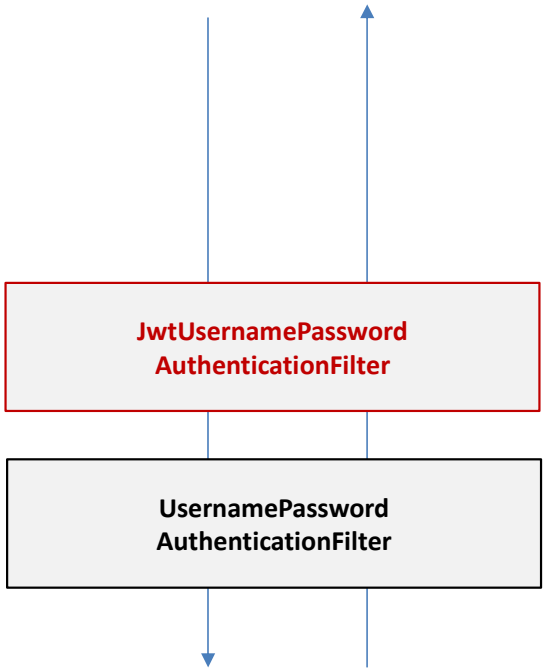
- 성공 시: LoginSuccessHandler가 처리
  - 접근 토큰과 사용자 정보를 json으로 응답

- 실패 시: LoginFailureHandler가 처리
  - 디폴트 예외처리기(ApiExceptionAdvice)로 처리 위임

API 로그인

JwtUsernamePasswordAuthenticationFilter

- 약속된 login url 요청인 경우 로그인 절차 수행
- UsernamePasswordAuthenticationFilter 상속
- 생성자에서 의존성 주입
  - AuthenticationManager
  - LoginSuccessHandler
  - LoginFailureHandler



  - 초기화
    - 인증 url 등록
    - 인증 성공시 핸들러, 인증 실패시 핸들러 등록

  - 주의
    - AuthenticationManager는 SecurityConfig가 생성된 이후에 Bean으로 등록
    - SecurityConfig에서 @Autowired로 연결해야 함

- SecurityConfig
  - 기존 UsernamePasswordAuthenticationFilter 앞에 추가

API 로그인

JwtUsernamePasswordAuthenticationFilter

- attemptAuthentication() 메서드
  - 등록한 인증 url로 요청이 오면 호출
  - 요청에서 username, password 부분을 추출
    → UsernamePasswordAuthenticationToken 구성
  - AuthenticationManager에게 인증 요청
  - 인증 경과 Authentication을 리턴

API 로그인

인증 성공 핸들러

- AuthenticationSuccessHandler 인터페이스의 구현체

```
public interface AuthenticationSuccessHandler {

    default void onAuthenticationSuccess(
          HttpServletRequest request, HttpServletResponse response,
          FilterChain chain, Authentication authentication) throws IOException, ServletException {
        this.onAuthenticationSuccess(request, response, authentication);
        chain.doFilter(request, response);
    }

    void onAuthenticationSuccess(
                HttpServletRequest request,
                HttpServletResponse response,
                Authentication authentication) throws IOException, ServletException;
}
```

- 로그인 결과를 JSON으로 직접 응답
  - 접근 토큰, 갱신 토큰, 사용자 기본 정보(username, email, 권한목록 등)을 담아 json으로 응답

API 로그인

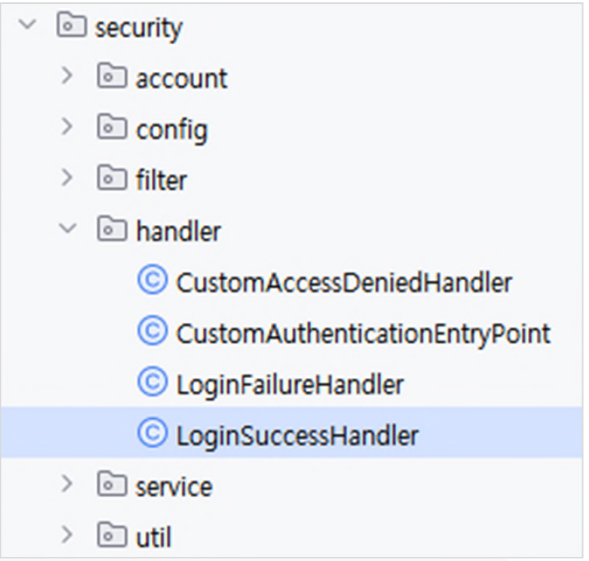
인증 실패 핸들러

- AuthenticationFailureHandler 인터페이스 구현체

```
public interface AuthenticationFailureHandler {
    void onAuthenticationFailure(
            HttpServletRequest request,
            HttpServletResponse response,
            AuthenticationException exception) throws IOException, ServletException;
}
```

- 기본 예외 처리로 이동
  - ApiExceptionAdvice에서 예외 응답하도록 유도

security.handler.LoginSuccessHandler.java




```
package org.scoula.security.handler;
...
import java.io.IOException;

@Log4j2
@Component
@RequiredArgsConstructor
public class LoginSuccessHandler implements AuthenticationSuccessHandler {
   private final JwtProcessor jwtProcessor;

   private AuthResultDTO makeAuthResult(CustomUser user) {
      String username = user.getUsername();
      // 토큰 생성
      String token = jwtProcessor.generateToken(username);
      // 토큰 + 사용자 기본 정보 (사용자명, ...)를 묶어서 AuthResultDTO 구성
      return new AuthResultDTO(token, UserInfoDTO.of(user.getMember()));
   }
```

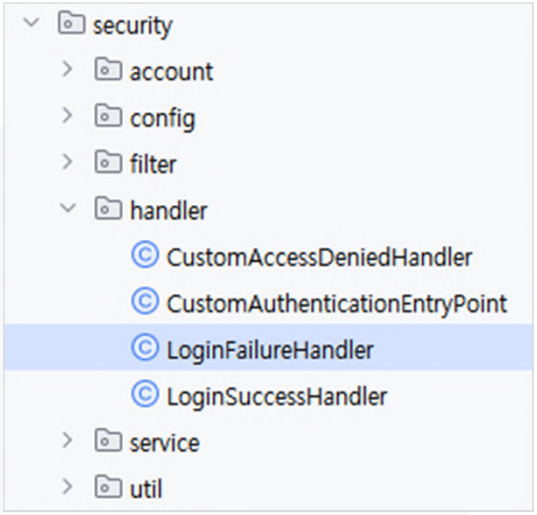
API 로그인

security.handler.LoginSuccessHandler.java

```
    @Override
    public void onAuthenticationSuccess(HttpServletRequest request, HttpServletResponse response,
                        Authentication authentication) throws IOException, ServletException {
        // 인증 결과 Principal
        CustomUser user = (CustomUser) authentication.getPrincipal();

        // 인증 성공 결과를 JSON으로 직접 응답
        AuthResultDTO result = makeAuthResult(user);
        JsonResponse.send(response, result);
    }
}
```

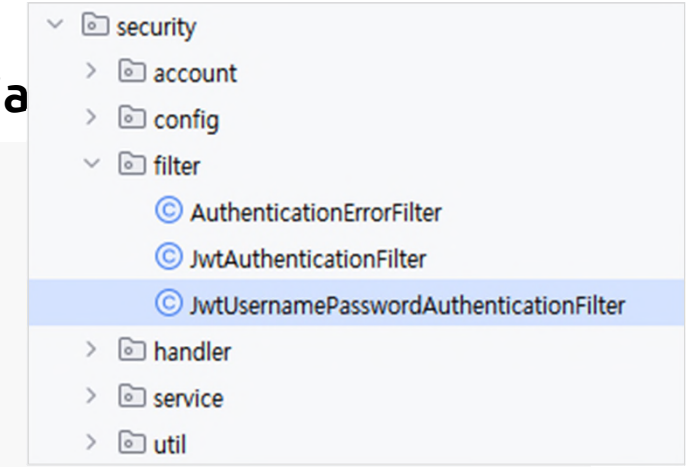
API 로그인

security.handler.LoginFailureHandler.java




```
package org.scoula.security.handler;
...
import java.io.IOException;

@Component
public class LoginFailureHandler implements AuthenticationFailureHandler {

    @Override
    public void onAuthenticationFailure(HttpServletRequest request, HttpServletResponse response,
                AuthenticationException exception) throws IOException, ServletException {

      JsonResponse.sendError(response, HttpStatus.UNAUTHORIZED, "사용자 ID 또는 비밀번호가 일치하지 않습니다.");
    }
}
```

API 로그인



security.filter.JwtUsernamePasswordAuthenticationFilter.ja

```
package org.scoula.security.filter;
...
import org.springframework.security.core.Authentication;
...

@Log4j2
@Component
public class JwtUsernamePasswordAuthenticationFilter extends UsernamePasswordAuthenticationFilter {
                                // 스프링 생성자 주입을 통해 전달
    public JwtUsernamePasswordAuthenticationFilter(
            AuthenticationManager authenticationManager,        // SecurityConfig가 생성된 이후에 등록됨
            LoginSuccessHandler loginSuccessHandler,
            LoginFailureHandler loginFailureHandler) {
        super(authenticationManager);

        setFilterProcessesUrl("/api/auth/login");                       // POST 로그인 요청 url
        setAuthenticationSuccessHandler(loginSuccessHandler);   // 로그인 성공 핸들러 등록
        setAuthenticationFailureHandler(loginFailureHandler);  // 로그인 실패 핸들러 등록
    }
```

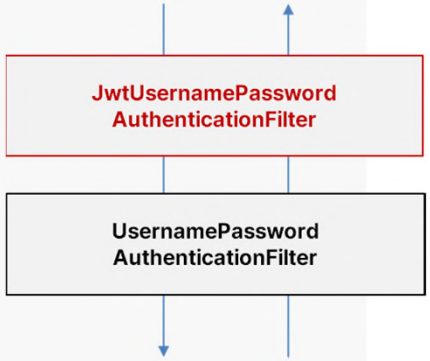
API 로그인

security.filter.JwtUsernamePasswordAuthenticationFilter.java

```
  // 로그인 요청 URL인 경우 로그인 작업 처리
  @Override
  public Authentication attemptAuthentication(HttpServletRequest request, HttpServletResponse response)
        throws AuthenticationException {

    // 요청 BODY의 JSON에서 username, password → LoginDTO
    LoginDTO login = LoginDTO.of(request);

    // 인증 토큰(UsernamePasswordAuthenticationToken) 구성
    UsernamePasswordAuthenticationToken authenticationToken =
          new UsernamePasswordAuthenticationToken(login.getUsername(), login.getPassword());

    // AuthenticationManager에게 인증 요청
    return getAuthenticationManager().authenticate(authenticationToken);
  }

}
```

security.config.SecurityConfig.java

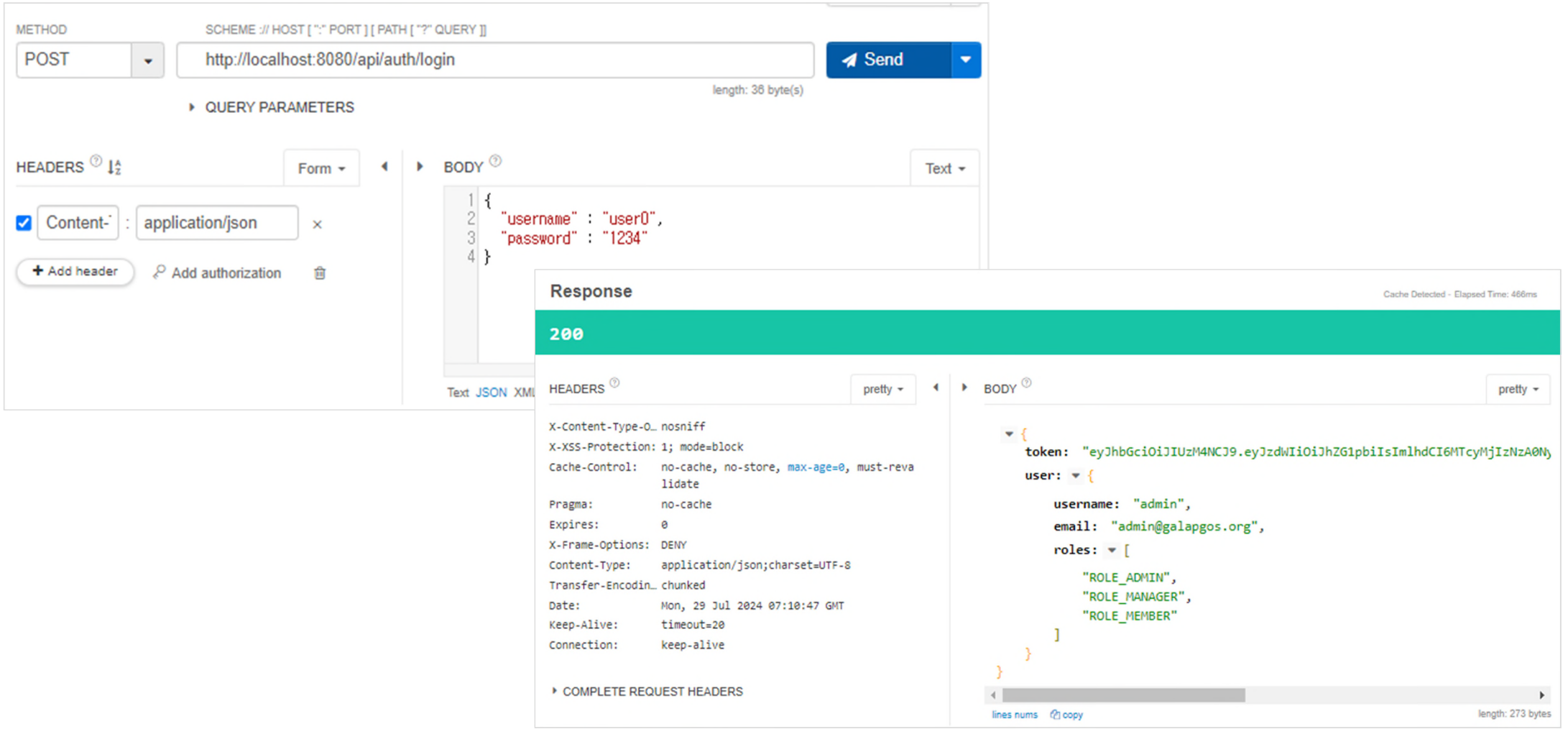
```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
  ...
  @Autowired
  private JwtUsernamePasswordAuthenticationFilter jwtUsernamePasswordAuthenticationFilter;
  ...
  @Override
  public void configure(HttpSecurity http) throws Exception {
    // 한글 인코딩 필터 설정
    http.addFilterBefore(encodingFilter(), CsrfFilter.class)
    // 로그인 인증 필터
      .addFilterBefore(jwtUsernamePasswordAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);

    http.httpBasic().disable()                    // 기본 HTTP 인증 비활성화
        .csrf().disable()      // CSRF 비활성화
        .formLogin().disable()  // formLogin 비활성화 □ 관련 필터 해제
        .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS); // 세션 생성 모드 설정
  }
```



- JwtUsernamePasswordAuthenticationFilter의 위치 설정
  - 기존 필터를 기준으로 설정

API 로그인

로그인 요청



API 로그인

로그인 요청