

2024년 상반기 K-디지털 트레이닝

로그인과 로그아웃 처리

[KB] IT's Your Life

✓ Spring Security 설정 후 POST 요청

○ 한글 문자 인코딩 발생

- WebConfig에서 등록한 문자 인코딩 필터보다 먼저 Security Filter가 먼저 동작
- Security Filter에서 POST body가 resolve됨 → 한글 깨짐
- Spring Security Filter 체인에서 문자 인코딩 필터를 CsrfFilter 보다 앞에 등록 필요

✓ SecurityController.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    // 문자셋 필터
    public CharacterEncodingFilter encodingFilter() {
        CharacterEncodingFilter encodingFilter = new CharacterEncodingFilter();
        encodingFilter.setEncoding("UTF-8");
        encodingFilter.setForceEncoding(true);
        return encodingFilter;
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.addFilterBefore(encodingFilter(), CsrfFilter.class);
    }
    ...
}
```

✓ controller/SecurityController.java

```
package org.scoula.controller;

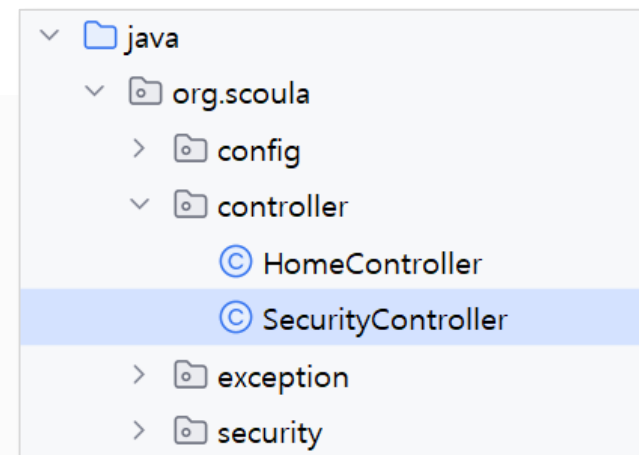
...

@Log4j
@RequestMapping("/security")
@Controller
public class SecurityController {

    @GetMapping("/all")    // 모두 접근 가능
    public void doAll() {
        log.info("do all can access everybody");
    }

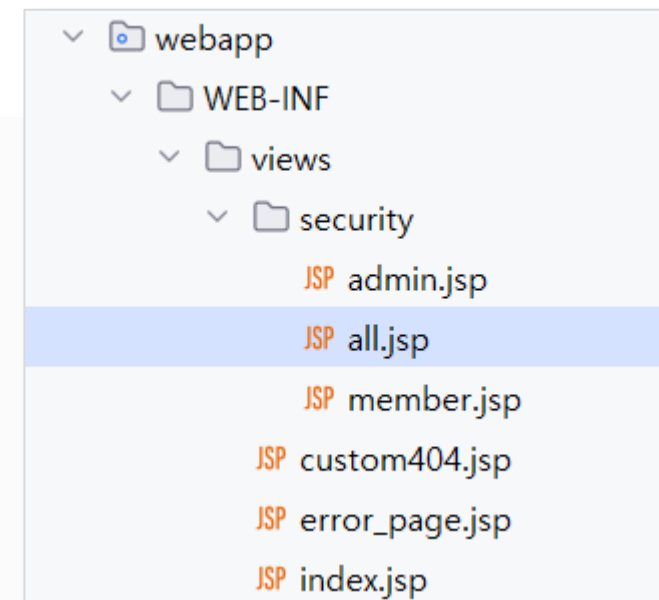
    @GetMapping("/member") // MEMBER 또는 ADMIN 권한 필요
    public void doMember() {
        log.info("logged member");
    }

    @GetMapping("/admin") // ADMIN 권한 필요
    public void doAdmin() {
        log.info("admin only");
    }
}
```



✓ views/security 폴더에 all.jsp, member.jsp, admin.jsp 생성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>/security/all page</h1>
</body>
</html>
```



- <h1> 부분은 각 페이지에 맞게 수정

✓ 경로별 인증/권한 설정

- `public void configure(HttpSecurity http)`이 http 인자로 설정
 - 내부적으로 builder 패턴적용되어 있음.
 - 대부분 메서드의 리턴값이 `HttpSecurity`임 → 메서드 체이닝으로 설정해 나감

```
http.authorizeRequests()
    // 모두 허용
    .antMatchers("/security/all").permitAll()
    // 특정 역할에게만 허용
    .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
    .antMatchers("/security/member").access("hasRole('ROLE_MEMBER')")
    // 로그인 사용자에게 허용
    .antMatchers("/board/write",
        "/board/modify",
        "/board/delete").authenticated();
```

✓ SecurityConfig.java

```
...

@Configuration
@EnableWebSecurity
@Log4j
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    public void configure(HttpSecurity http) throws Exception {

        // 경로별 접근 권한 설정
        http.authorizeRequests()
            .antMatchers("/security/all").permitAll()
            .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
            .antMatchers("/security/member").access("hasRole('ROLE_MEMBER')");
    }
}
```

- ✓ <http://localhost:8080/security/all>
- ✓ <http://localhost:8080/security/member>
- ✓ <http://localhost:8080/security/admin>
- 에러 발생보다는 로그인 페이지로 리다이렉트하는 것이 좋음
--> 로그인 설정 필요

/security/all page

HTTP 상태 403 – 금지됨

타입 상태 보고

메시지 Access Denied

설명 서버가 요청을 이해했으나 승인을 거부합니다.

VMware tc Runtime 9.0.33.A.RELEASE

✓ 로그인 설정

○ HttpSecurity http

- form 기반의 로그인 설정

```
http.formLogin()           // 로그인 폼 설정 시작
    .loginPage("/customLogin") // 로그인 요청 폼 GET url 설정
    .loginProcessingUrl("/login"); // 로그인 POST 요청 url 설정
```

✔ SecurityConfig.java

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

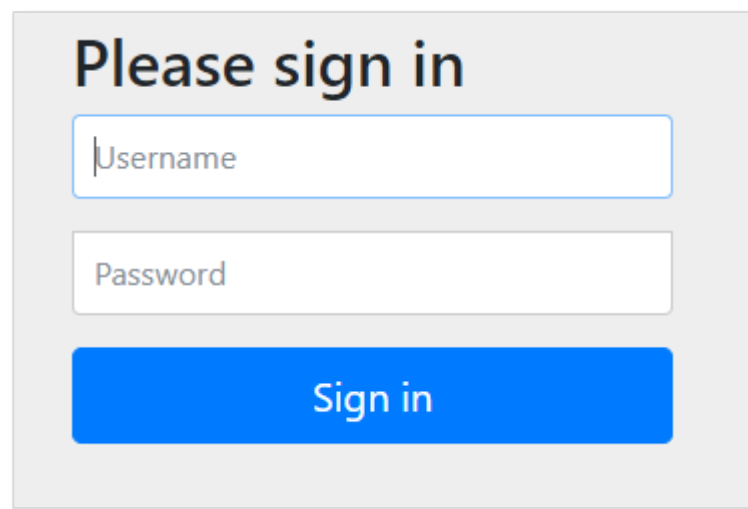
    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/security/all").permitAll()
            .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
            .antMatchers("/security/member").access("hasAnyRole('ROLE_MEMBER', 'ROLE_ADMIN')");

        http.formLogin(); // form 기반 로그인 활성화, 나머지는 모두 디폴트
    }
}
```

- ✓ <http://localhost:8080/security/member>
- ✓ <http://localhost:8080/security/admin>

→ /login으로 리다이렉트

<http://localhost:8080/login>



Please sign in

Username

Password

Sign in

✓ 인증 정보 설정

○ `protected void configure(AuthenticationManagerBuilder auth)`

- 사용자 정보를 어디서(메모리, 파일, db 등) 얻을지 설정
- 메모리에 사용자 정보 구축하는 경우
 - 주로 테스트용

```
auth.inMemoryAuthentication() // 메모리에서 사용자 정보 설정
    .withUser("admin")         // username, 사용자 id
    .password("{noop}1234") // 비밀번호, {noop}는 암호화 없음 의미
    .roles("ADMIN","MEMBER");  // ROLE_ADMIN 역할 설정
```

✓ SecurityConfig.java

```
@Override
protected void configure(AuthenticationManagerBuilder auth)
    throws Exception {
    log.info("configure .....");

    auth.inMemoryAuthentication()
        .withUser("admin")
        .password("{noop}1234")
        .roles("ADMIN","MEMBER");    // ROLE_ADMIN

    auth.inMemoryAuthentication()
        .withUser("member")
        .password("{noop}1234")
        .roles("MEMBER"); // ROLE_MEMBER
    }
}
```

- admin 또는 member로 로그인 가능

✓ 로그인 페이지 커스터마이징

- 기본적으로 제공되는 localhost:8080/login 대신 새로운 로그인 페이지 운영

- 로그인 설정

http.formLogin()	// 로그인 설정 시작
.loginPage("/security/login")	// 로그인 페이지 GET URL → security/login 뷰(jsp) 정의
.loginProcessingUrl("/security/login")	// 로그인 POST URL → login form의 action에 지정
.defaultSuccessUrl("/");	// 로그인 성공 시 이동(redirect)할 페이지

✓ config/SecurityConfig.java

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

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/security/all").permitAll()
            .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
            .antMatchers("/security/member").access("hasAnyRole('ROLE_MEMBER', 'ROLE_ADMIN')");

        http.formLogin()
            .loginPage("/security/login")
            .loginProcessingUrl("/security/login")
            .defaultSuccessUrl("/");
    }
}
```

✓ controller/SecurityController.java

```
...
@RequestMapping("/security")
public class SecurityController {
    ...

    @GetMapping("/login")
    public void login() {
        log.info("login page");
    }
}
```


✓ CSRF(Cross Site Request Forgery) 공격

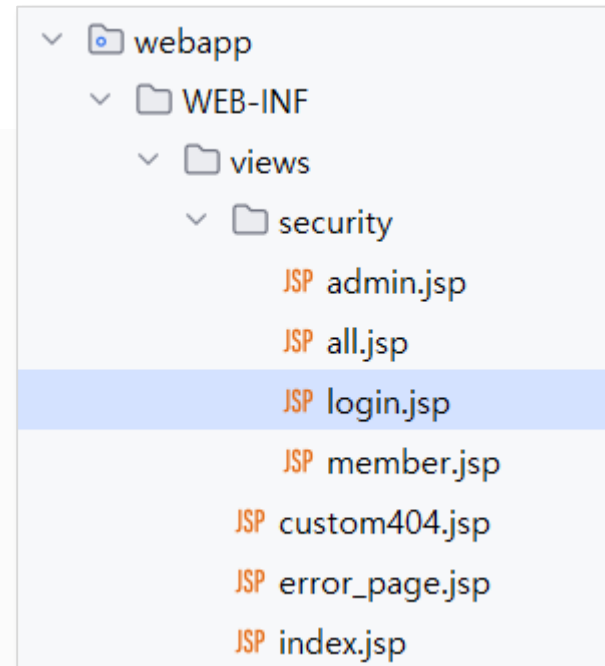
- 인터넷 사용자(희생자)가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 만드는 공격
- POST 요청을 위조하여 전송하는 것
- 방어책
 - CSRF 토큰 운영
 - form 페이지 GET요청 시 form 내에 인증 토큰을 심어서 전송
 - 인증 토큰이 있는 경우에만 정당한 POST 요청으로 인식
 - 해당 토큰이 없으면 에러를 발생시킴
- spring-security 사용시 디폴트로 사용하는 것으로 설정됨

✓ 로그인 form

- 요청 파라미터의 이름이 정해져 있음
 - username: 사용자 id
 - password: 비밀번호

✓ security/login.jsp

```
<body>
  <h1>login</h1>
  <form name='f' action='/security/login' method='POST'>
    <input type="hidden" name="_csrf.parameterName" value="_csrf.token" />
    <table>
      <tr>
        <td>User:</td>
        <td><input type='text' name='username' value=''></td>
      </tr>
      <tr>
        <td>Password:</td>
        <td><input type='password' name='password' /></td>
      </tr>
      <tr>
        <td colspan='2'>
          <input name="submit" type="submit" value="Login" />
        </td>
      </tr>
    </table>
  </form>
</body>
</html>
```



login

User:

Password:

```
<form name='f' action='/security/login' method='POST'>
  <input type="hidden" name="_csrf" value="d5bab99f-8a5d-46ad-990e-8b5504bce41b" />
  <table>
    ...
```

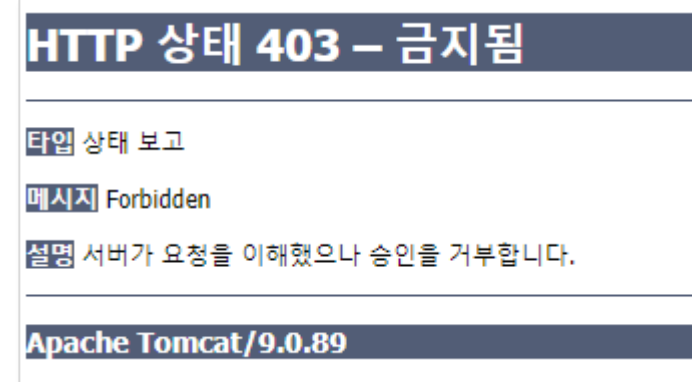
✓ 확인

- <http://localhost:8080/security/member> 요청
 - 로그인하지 않은 상태이므로 /security/login으로 리다이렉트 됨
 - member로 로그인



A screenshot of a web login form. The title is "login". It contains two input fields: "User:" and "Password:". Below the password field is a "Login" button.

- 로그인 성공 시
 - 원래 요청 url로 리다이렉트됨
 - <http://localhost:8080/security/member>



- <http://localhost:8080/security/admin>을 요청하고, member로 로그인 하면 → 403 에러
- member로 로그인한 상태에서 <http://localhost:8080/security/admin>을 요청하면 → 403 에러

✓ 접근 권한 설정 페이지 접근 시

- 권한이 맞으면 요청한 페이지로 진입
- 권한이 맞지 않으면
 - 로그인 하지 않은 경우 login 페이지로 이동
→ 로그인 이후 해당 페이지로 들어감
 - 로그인 된 상태라면 403에러 발생

✓ 로그아웃 설정

○ 로그아웃 시 해야 할 일

- 세션 무효화(invalidate)
- 쿠키 제거: JSESSION-ID, remember-me

```
http.logout()                // 로그아웃 설정 시작
    .logoutUrl("/security/logout") // 로그아웃 호출 url
    .invalidateHttpSession(true)  // 세션 invalidate
    .deleteCookies("remember-me", "JSESSION-ID") // 삭제할 쿠키 목록
    .logoutSuccessUrl("/");       // 로그아웃 이후 이동할 페이지
```

○ logout url을 post로 요청 !!

✓ config/SecurityConfig.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    ...
    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/security/all").permitAll()
            .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
            .antMatchers("/security/member").access("hasAnyRole('ROLE_MEMBER', 'ROLE_ADMIN')");

        http.formLogin()
            .loginPage("/security/login")
            .loginProcessingUrl("/security/login")
            .defaultSuccessUrl("/");

        http.logout()                // 로그아웃 설정 시작
            .logoutUrl("/security/logout") // POST: 로그아웃 호출 url
            .invalidateHttpSession(true) // 세션 invalidate
            .deleteCookies("remember-me", "JSESSION-ID") // 삭제할 쿠키 목록
            .logoutSuccessUrl("/security/logout"); // GET: 로그아웃 이후 이동할 페이지
    }
    ...
}
```

✓ controller.SercurityController.java

```
@RequestMapping("/security")
public class SecurityController {

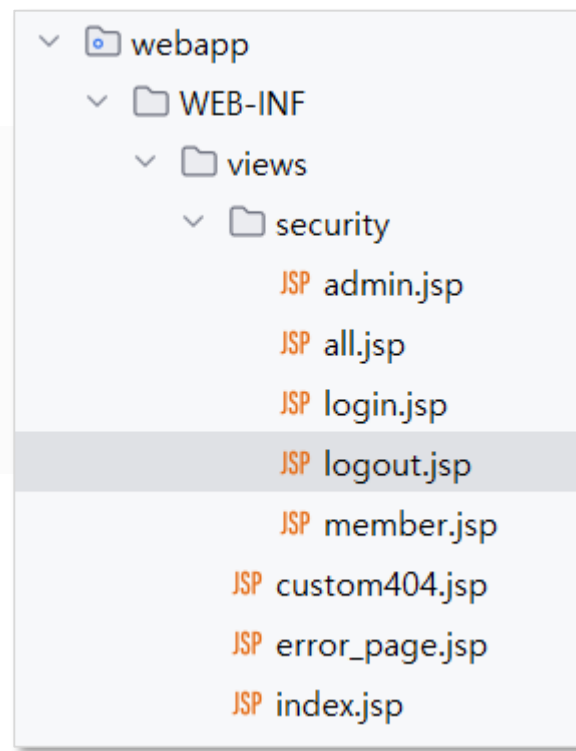
    ...

    @GetMapping("/logout")
    public void logout() {
        log.info("logout page");
    }
}
```


✓ security/logout.jsp

```
<body>
  <h1>
    로그 아웃됨
  </h1>

  <a href="/">홈으로</a>
</body>
```



✓ member.jsp, admin.jsp

```
..  
<body>  
  <h1>/security/member page</h1>  
  
  <form action="/security/logout" method="post">  
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />  
    <input type="submit" value="로그아웃"/>  
  </form>  
</body>  
</html>
```

/security/member page

로그아웃

redirect: /security/logout

http://localhost:8080/security/logout

로그 아웃됨

홈으로

✓ PasswordEncoder 인터페이스

○ 비밀번호는 반드시 암호화해서 처리해야 함

메서드	설명
String encode(String rawPassword)	암호화되지 않은 비밀번호(rawPassword)를 암호화해서 리턴함 같은 값을 암호화해도 매번 다른 값을 리턴 equals()로 비교할 수 없음
boolean matches(String rawPssword, String encodedPassword)	사용자가 입력한 암호화되지 않은 비밀번호(rawPassword)와 암호화된 비밀번호 (DB. 저장값)가 일치하는지 검사 같으면 true, 다르면 false 리턴

○ 구현체

- BCryptPasswordEncoder

✓ config/SecurityConfig.java

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

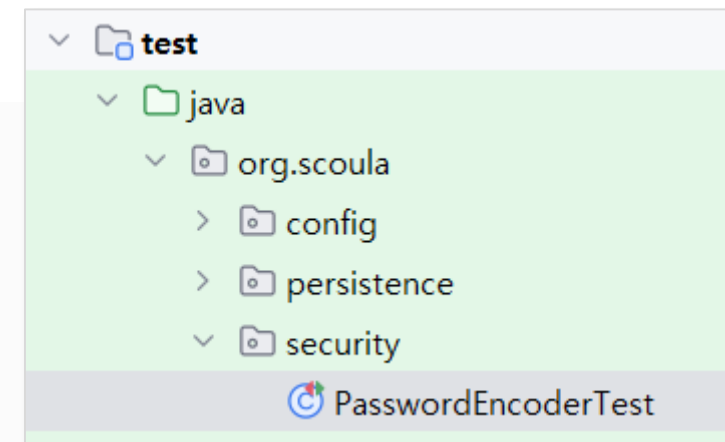
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    ...
}
```

✓ tests:: PasswordEncoderTest.java

```
@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes = {
    RootConfig.class,
    SecurityConfig.class
})
@Log4j
public class PasswordEncoderTest {

    @Autowired
    private PasswordEncoder pwEncoder;
```



✓ tests: PasswordEncoderTests.java

```
@Test
public void testEncode() {
    String str = "1234";

    String enStr = pwEncoder.encode(str);    // 암호화
    log.info("password: " + enStr);

    String enStr2 = pwEncoder.encode(str);    // 암호화
    log.info("password: " + enStr2);

    log.info("match :" + pwEncoder.matches(str, enStr));    // 비밀번호 일치 여부 검사
    log.info("match :" + pwEncoder.matches(str, enStr2));    // 비밀번호 일치 여부 검사
}
}
```

```
INFO : org.galapagos.security.SecurityTest -
password: $2a$10$VJK.3K/W3PhSu53.FVm7W0EzFZPLGTw5.iiCZXgKTHPkhK419Jdz2
INFO : org.galapagos.security.SecurityTest -
password: $2a$10$ME3YMFVYP.Wi1YTL5ghU9.yPyuEkEBXpQl9FHBsZ9BpP0tef8hj72
INFO : org.galapagos.security.SecurityTest - match :true
INFO : org.galapagos.security.SecurityTest - match :true
```

같은 문자열이어도
매번 다르게 암호화됨

✓ config/SecurityConfig.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {
...
    @Override
    protected void configure(AuthenticationManagerBuilder auth)
        throws Exception {
        auth.inMemoryAuthentication()
            .withUser("admin")
//      .password("{noop}1234")
//      .password("$2a$10$EsIMfxbJ6NuvwX7MDj4Wq0YFzLU9U/lldCyn0nic5dFo3VfJYrXYC")
            .roles("ADMIN","MEMBER");    // ROLE_ADMIN, ROLE_MEMBER

        auth.inMemoryAuthentication()
            .withUser("member")
//      .password("{noop}1234")
//      .password("$2a$10$EsIMfxbJ6NuvwX7MDj4Wq0YFzLU9U/lldCyn0nic5dFo3VfJYrXYC")
            .roles("MEMBER"); // ROLE_MEMBER
    }
...
}
```