



[한국ICT인재개발원] 스프링 프레임워크

12. 스프링 시큐리티 심화(.jsp파일, 자동로그인, 어노테이션)

前) 광고데이터 분석 1년

前) IT강의 경력 2년 6개월

前) 머신러닝을 활용한 데이터 분석 프로젝트반 운영 1년

前) 리그오브 레전드 데이터 분석 등...

現) 국비반 강의 진행중

기초적인 내용을 배웠다면 이제는 실질적으로 내가 만든 웹사이트에 스프링 시큐리티를 적용하는 방법을 배워보겠습니다.

먼저, `jsp`에서 스프링 시큐리티 옵션을 쓰는법을 배워본 다음,
자동로그인 체크와 어노테이션을 이용한 적용까지 배우면
웹 사이트에 스프링 시큐리티를 좀 더 편하게 적용할 수 있습니다.

admin.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://www.springframework.org/security/tags" prefix="sec" %>
4 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
5 <!DOCTYPE html>
6 <html>
7 <head>
8 <meta charset="UTF-8">
9 <title>Insert title here</title>
```

먼저 admin.jsp파일 내부에서 sec 태그를 사용해보겠습니다.

c태그라이브러리처럼 taglib설정을 해 주면 됩니다.

admin.jsp

```
<body>
  <h1>admin 주소</h1>
  <h2>다양한 페이지 정보</h2>

  <p>principal : <sec:authentication property="principal"/></p>
  <p>MemberVO : <sec:authentication property="principal.member"/></p>
  <p>사용자의 이름 : <sec:authentication property="principal.member.userName"/></p>
  <p>사용자의 아이디 : <sec:authentication property="principal.member.userid"/></p>
  <p>사용자 권한목록 : <sec:authentication property="principal.member.authList"/></p>

  <hr>
  <a href="/customLogout">로그아웃페이지 이동</a>
</body>
```

body태그 내부는 다음과 같이 고쳐주세요

principal이 받아오는것이 바로 UserDetailsService에서 받아온 정보입니다.

```
CustomUserDetailsService.java ✕  
  
@Log4j  
public class CustomUserDetailsService implements UserDetailsService {  
  
    @Autowired  
    private MemberMapper mapper;  
  
    @Override  
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
  
        Log.warn("유저 이름 확인: " + username);  
        MemberVO vo = mapper.read(username);  
        Log.info("확인된 유저이름으로 얻어온 정보 : " + vo);  
  
        return vo == null? null : new CustomUser(vo);  
    }  
}
```

위와 같이 username을 토대로 얻어온 유저 정보를

MemberVO로 만들어서 리턴을 한 결과물을 화면에 뿌려줍니다.

세팅이 되어있다면 그냥 로그인한 계정의 MemberVO를 보내준다고 보시면 됩니다.

admin 주소

다양한 페이지 정보

principal : org.ict.domain.CustomUser@ce2b2b4e: Username: user25; Password: [PROTECTED];
Enabled: true; AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked: true;
Granted Authorities: ROLE_ADMIN

MemberVO : MemberVO(userid=user25,
userpw=\$2a\$10\$0h0JbTQhiZ5KqBqEaJs5T.iI2TNe0fcSLRcphUmAmGipdGlqEOccS, userName=운영
자25, enabled=false, regDate=2021-10-03, updateDate=2021-10-03, authList=
[AuthVO(userid=user25, auth=ROLE_ADMIN)])

사용자의 이름 : 운영자25

사용자의 아이디 : user25

사용자 권한목록 : [AuthVO(userid=user25, auth=ROLE_ADMIN)]

[로그아웃페이지 이동](#)

로그인 하면 이제 위와 같이 principal에는 전체 데이터가 담겨있고,
principal.member에는 계정에 관련된 전체 정보가
principal.member.변수명 을 .jsp에서 출력하면 vo의 멤버변수가 출력됩니다.

표현식	역할
hasRole([role]) hasAuthority([authority])	로그인한 사용자가 제시한 권한이 있으면 true , 없으면 false
hasAnyRole([role,role2]) hasAnyAuthority([authority])	제시된 권한중 하나만 가지고 있어도 true , 하나도 없으면 false
principal	현재 접속 사용자의 전체적인 정보(5~7페이지에서 이미 확인함)
permitAll	모든 사용자에게 허용
denyAll	모든 사용자에게 거부
isAnonymous()	익명 사용자이면 true (로그인 안한 사용자도 익명으로 간주)
isAuthenticated()	인증된 사용자면 true
isFullyAuthenticated()	자동로그인으로 접속하지 않은 사용자이면서 권한이 맞으면 true

이제 스프링 시큐리티에서 자주 쓰는 표현식에 대해 보겠습니다.
조건식으로 사용하며 **sec**태그와 조합해 쓰는것이 특징입니다.

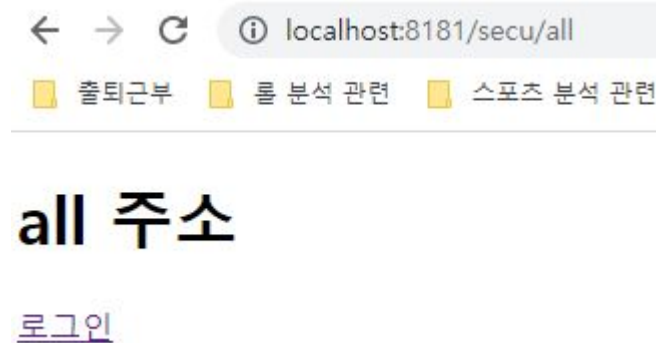

```
2  pageEncoding="UTF-8"%>
3  <%@ taglib uri="http://www.springframework.org/security/tags" prefix="sec" %>
4  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
5  <!DOCTYPE html>
6  <html>
7  <head>
8  <meta charset="UTF-8">
9  <title>Insert title here</title>
10 </head>
11 <body>
12     <h1>all 주소</h1>
13
14     <sec:authorize access="isAnonymous()">
15         <!-- 로그인 안한(익명) 사용자인 경우 -->
16         <a href="/customLogin">로그인</a>
17     </sec:authorize>
18     <sec:authorize access="isAuthenticated()">
19         <!-- 로그인 한(인증된) 사용자인 경우 -->
20         <a href="/customLogout">로그아웃</a>
21     </sec:authorize>
22 </body>
```

추가

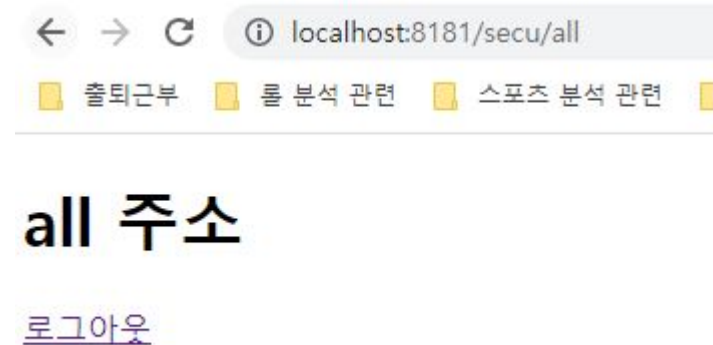
*all.jsp

/secu/all 의 주소로 접속했을때 이제 로그인 안한 사용자에게는 로그인창 링크를, 그리고 로그인한 사용자에게는 로그아웃 링크를 보여주도록 코드를 위와 같이 고칩니다.

```
<sec:authorize access="isAnonymous()">
<!-- 로그인 안한(익명) 사용자인 경우 -->
    <a href="/customLogin">로그인</a>
</sec:authorize>
```



```
<sec:authorize access="isAuthenticated()">
<!-- 로그인 한(인증된) 사용자인 경우 -->
    <a href="/customLogout">로그아웃</a>
</sec:authorize>
```



이제 **sec:authorize** 태그에 의해 조건식처럼 로그인한 사용자와 그렇지 않은 사용자간 보여지는 내용이 살짝 달라집니다.

```
CREATE TABLE persistent_logins (  
    username varchar(64) not null,  
    series varchar(64) primary key,  
    token varchar(64) not null,  
    last_used timestamp not null  
);
```

이제 자동로그인 기능을 구현해보겠습니다.

한 번 로그인하면 일정 시간동안 로그인로직 없이도 로그인유지가 되도록
만드는것으로 먼저 테이블을 추가합니다.

위 테이블 양식으로 만드는게 스프링시큐리티의 공식 문서 권장사항입니다.

```
<security:form-login login-page="/customLogin"
    authentication-success-handler-ref="customLoginSuccess"/>

<!-- <security:access-denied-handler error-page="/accessError" /> -->
<security:access-denied-handler ref="customAccessDenied"/>

<security:logout logout-url="/customLogout" invalidate-session="true" />
<security:remember-me data-source-ref="dataSource" token-validity-seconds="604800"/>
</security:http>
```

↑ 값은 기본 유지주기 추가

테이블이 새로 생성되었다면

security-context.xml을 위와 같이 수정합니다. 604800초만큼 자동로그인이 유지되도록 해 주는것으로 설정은 자유롭게 바꿔주세요.


```
<body>
  <h1>사용자 생성 로그인 폼</h1>
  <h2><c:out value="${error }" /></h2>
  <h2><c:out value="${logout }" /></h2>

  <form action="/login" method='post'>
    아이디: <input type="text" name="username" value="admin"><br/>
    비밀번호: <input type="text" name="password" value="admin"><br/>
    자동로그인 : <input type="checkbox" name="remember-me"><br/>
    <input type="submit" value="로그인하기">
    <input type="hidden" name="${_csrf.parameterName }" value="${_csrf.token }" />
  </form>
</body>
```

이제 로그인화면에 checkbox를 구현합니다.
name속성에는 remember-me를 주면 됩니다.

customLogin.jsp를 위와 같이 수정합니다.

admin 주소

다양한 페이지 정보

principal : org.ict.domain.CustomUser@ce2b2b4e: Username: user25; Password: [PROTECTED];
Enabled: true; AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked: true;
Granted Authorities: ROLE_ADMIN

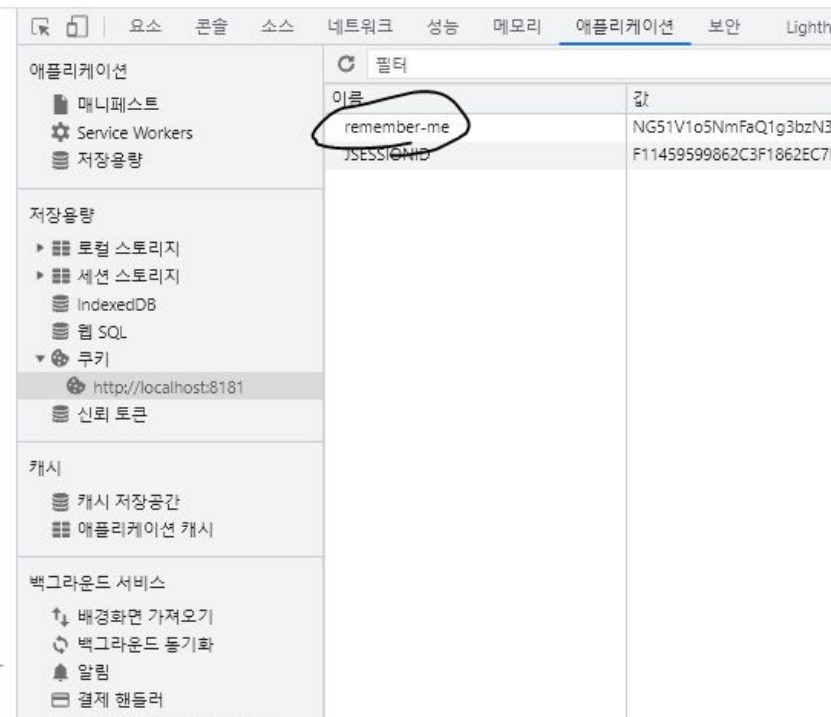
MemberVO : MemberVO(userid=user25,
userpw=\$2a\$10\$0h0JbTQhiZ5KqBqEaJs5T.il2TNe0fcSLRcphUmAmGipdGlqEOccS, userName=운
영자25, enabled=false, regDate=2021-10-03, updateDate=2021-10-03, authList=
[AuthVO(userid=user25, auth=ROLE_ADMIN)])

사용자의 이름 : 윤영자25

사용자의 아이디 : user25

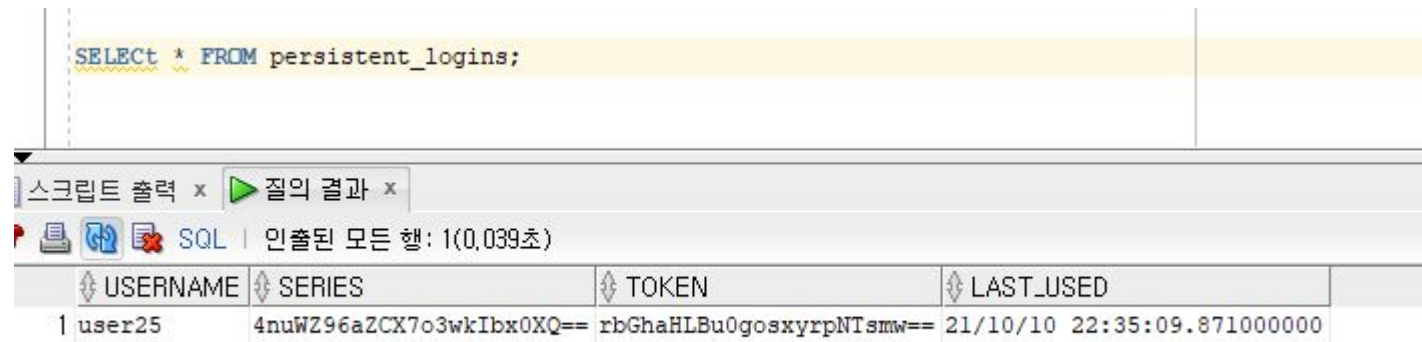
사용자 권한목록 : [AuthVO(userid=user25, auth=ROLE_ADMIN)]

[로그아웃페이지 이동](#)



이제 로그인시 f12 -> application -> storage 에서 세션발급여부를 검사하면

JSESSIONID에 이어서 remeber-me라는 쿠키가 추가로 발급되는것을 볼 수 있습니다.



```
SELECT * FROM persistent_logins;
```

	USERNAME	SERIES	TOKEN	LAST_USED
1	user25	4nuWZ96aZCX7o3wkIbx0XQ==	rbGhaHLBu0gosxyrpNTsmw==	21/10/10 22:35:09.871000000

그리고 DB쪽의 `persistent_logins` 테이블을 조회해보면

위와 같이 특정 유저의 마지막 접속 시점이 기록으로 자동기록됩니다.

admin 주소

다양한 페이지 정보

principal : org.ict.domain.CustomUser@ce2b2b4e: Username: user25; Password: [PROTECTED];
Enabled: true; AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked: true;
Granted Authorities: ROLE_ADMIN

MemberVO : MemberVO(userid=user25,
userpw=\$2a\$10\$0h0JbTQhiZ5KqBqEaJs5T.il2TNe0fcSLRcphUmAmGipdGlqEOccS, userName=운
영자25, enabled=false, regDate=2021-10-03, updateDate=2021-10-03, authList=
[AuthVO(userid=user25, auth=ROLE_ADMIN)])

사용자의 이름 : 운영자25

사용자의 아이디 : user25

사용자 권한목록 : [AuthVO(userid=user25, auth=ROLE_ADMIN)]

[로그아웃페이지 이동](#)

이제 브라우저를 다 꺾다 쳐도 로그인이 유지되는지

/secu/admin으로 브라우저를 완전히 꺾다가 접속해보겠습니다.


```
<!-- <security:access-denied-handler error-page="/accessError" /> -->
<security:access-denied-handler ref="customAccessDenied"/>

<security:logout logout-url="/customLogout" invalidate-session="true"
    delete-cookies="remember-me, JSESSIONID"/>

<security:remember-me data-source-ref="dataSource" token-validity-seconds="604800"/>

</security:http>
```

로그아웃시 remember-me 쿠키를 파기해야합니다.

security-context.xml에 위와 같이 추가합니다.

```
<security:http>
```

```
<!-- /secu/all 접속시 모든 유저에게 접근 허용 -->
```

```
<security:intercept-url pattern="/secu/all" access="permitAll" />
```

```
<!-- /secu/member는 member에게만 접근 허용 -->
```

```
<security:intercept-url pattern="/secu/member" access="hasRole('ROLE_MEMBER')" />
```

```
<!-- /secu/admin은 admin에게만 접근 허용 -->
```

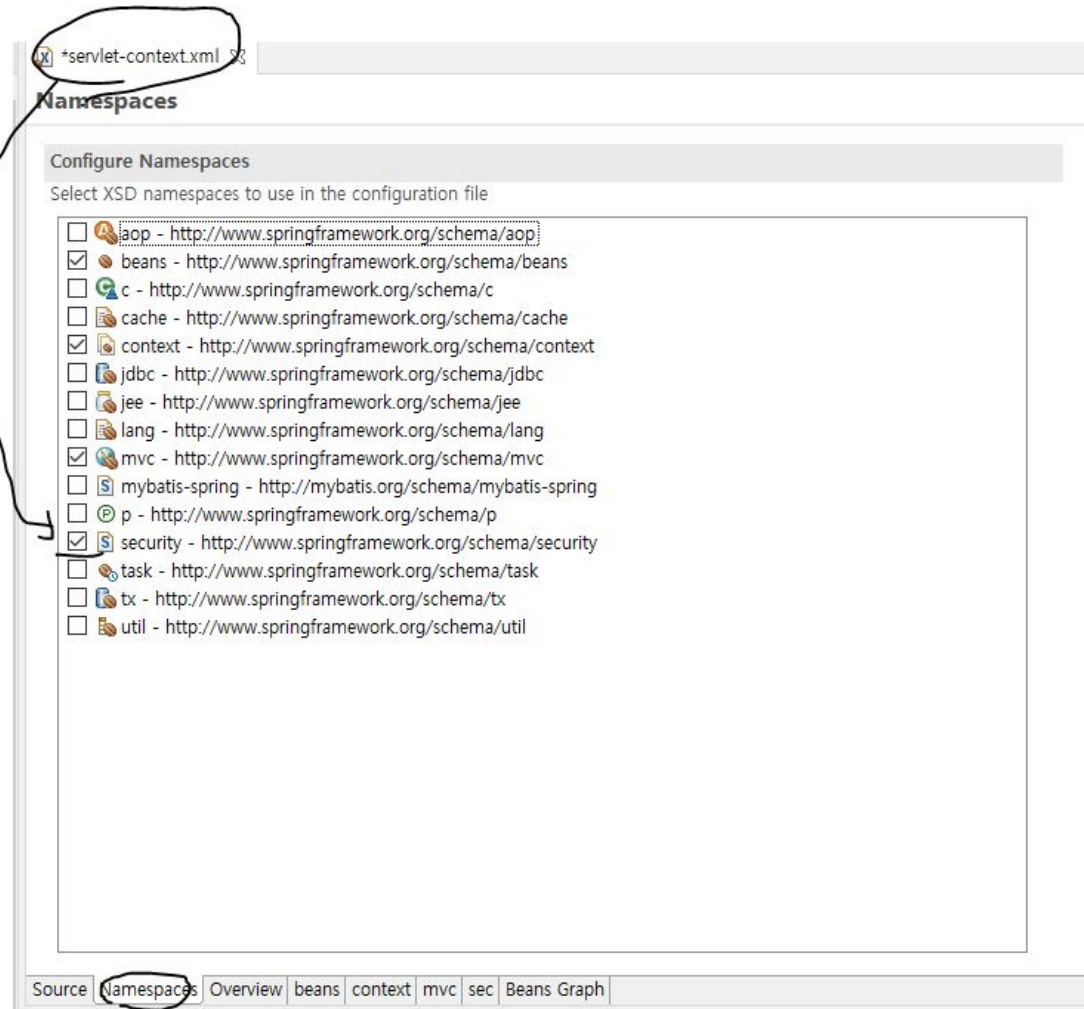
```
<security:intercept-url pattern="/secu/admin" access="hasRole('ROLE_ADMIN')" />
```

```
<security:form-login login-page="/customLogin"
```

```
authentication-success-handler-ref="customLoginSuccess"/>
```

모든 주소를 하나하나 security-context.xml에 security-intercept-url 태그를 추가해 시큐리티를 적용하는것은 복잡하고 관리도 어려운 일입니다.

따라서 이제는 어노테이션으로 처리하는 방법에 대해 배워봅시다.



먼저 **servlet-context.xml**의 **Namespaces**탭에서
security를 체크해줍니다.

```
3 XMLSchema-instance"
4 ework.org/schema/beans"
5 amework.org/schema/context"
6 ramework.org/schema/security"
7 ingframework.org/schema/security http://www.springframework.org/schema/security/spring-security-5.0.xsd
8 /schema/mvc https://www.springframework.org/schema/mvc/spring-mvc.xsd
9 /schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd
10 /schema/context https://www.springframework.org/schema/context/spring-context.xsd">
11
12 fines this servlet's request-processing infrastructure -->
--
```

-5.0
제거

다음 **servlet-context.xml**에도 역시 **-5.0**을 제거해줍니다.

이 작업은 **security-conetxt.xml**을 생성할때도 똑같이 해준 작업입니다.


```
        <beans:property name="suffix" value=".jsp" />
    </beans:bean>

    <context:component-scan base-package="org.ict.controller" />
    <security:global-method-security pre-post-annotations="enabled"
                                   secured-annotations="enabled"/>
</beans:beans>
```

제거가 완료되었다면 **security:global-method-security** 태그를 추가하고
위 두 속성을 모두 **enabled**로 고쳐줍니다.

```
<security:http>
```

```
<!-- /secu/all 접속시 모든 유저에게 접근 허용 -->
```

```
<security:intercept-url pattern="/secu/all" access="permitAll" />
```

```
<!-- /secu/member는 member에게만 접근 허용 -->
```

```
<!-- <security:intercept-url pattern="/secu/member" access="hasRole('ROLE_MEMBER')" /> -->
```

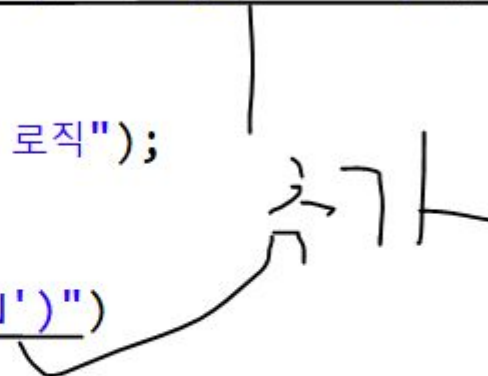
```
<!-- /secu/admin은 admin에게만 접근 허용 -->
```

```
<!-- <security:intercept-url pattern="/secu/admin" access="hasRole('ROLE_ADMIN')" /> -->
```

이제 security-context.xml 내부의 security:intercept-url을 주석처리하면

검증요건 없이 접속이 잘 되는걸 확인할 수 있습니다.

```
@PreAuthorize("hasAnyRole('ROLE_ADMIN','ROLE_MEMBER')")  
@GetMapping("/member")  
public void doMember() {  
    Log.info("회원들이 접속 가능한 member 로직");  
}  
  
@PreAuthorize("hasAnyRole('ROLE_ADMIN')")  
@GetMapping("/admin")  
public void doAdmin() {  
    Log.info("운영자만 접속 가능한 admin 로직");  
}
```



SecurityController로 와서, /member와 /admin패턴에

위와 같이 @PreAuthorize 어노테이션과 표현식을 추가합니다.

본 교안 8페이지에 의하면 저 권한중 하나를 가져야 로그인 가능합니다.

admin 주소

다양한 페이지 정보

principal : org.ict.domain.CustomUser@ce2b2b4e: Username: user25; Password: [PROTECTED]; Enabled: true; AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked: true; Granted Authorities: ROLE_ADMIN

MemberVO : MemberVO(userid=user25, userpw=\$2a\$10\$0h0JbTQhiZ5KqBqEaJs5T.il2TNe0fcSLRcphUmAmGipdGlqEOccS, userName=운영자25, enabled=false, regDate=2021-10-03, updateDate=2021-10-03, authList=[AuthVO(userid=user25, auth=ROLE_ADMIN)])

사용자의 이름 : 운영자25

사용자의 아이디 : user25

사용자 권한목록 : [AuthVO(userid=user25, auth=ROLE_ADMIN)]

[로그아웃페이지 이동](#)

이제 로그인에 잘 되는걸 확인해주시면 됩니다.