

2024년 상반기 K-디지털 트레이닝

SPRING Security2-2

[KB] IT's Your Life

- [spring_security2-2문제.zip](#) 프로젝트 파일을 다운로드(클릭) 받아 본인의 설정을 변경한 후, 프로젝트를 완성하시오.
- SecurityConfig.java, SecurityController.java, UserDetailsMapper.xml 파일 내 <코드 구현 부분>을 각 요구사항에 맞게 구현하시오.



• 테이블 생성

-- 사용자 정보 테이블

drop table if exists tbl_member;

create table **tbl_member**

(

username	varchar(50) primary key,
password	varchar(128) not null,
email	varchar(50) not null,
reg_date	datetime default now(),
update_date	datetime default now()

-- 사용자 id

-- 암호화된 비밀번호

);

-- 사용자 권한 테이블

drop table if exists tbl_member_auth;

create table **tbl_member_auth**

(

username	varchar(50) not null,
auth	varchar(50) not null,
primary key(username, auth),	
constraint fk_authorities_users foreign key (username) references tbl_member(username)	

-- 사용자 id

-- 권한 문자열 ROLE_ADMIN, ROLE_MANAGER, ROLE_MEMBER 등

-- 복합키

);

- 데이터 추가

-- 테스트 사용자 추가

```
insert into tbl_member(username, password, email)
```

```
values
```

```
    ('admin', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lldCyn0nic5dFo3VfJYrXYC', 'admin@galapgos.org'),  
    ('user0', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lldCyn0nic5dFo3VfJYrXYC', 'user0@galapgos.org'),  
    ('user1', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lldCyn0nic5dFo3VfJYrXYC', 'user1@galapgos.org'),  
    ('user2', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lldCyn0nic5dFo3VfJYrXYC', 'user2@galapgos.org'),  
    ('user3', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lldCyn0nic5dFo3VfJYrXYC', 'user3@galapgos.org'),  
    ('user4', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lldCyn0nic5dFo3VfJYrXYC', 'user4@galapgos.org');
```

```
select * from tbl_member;
```

- 권한 데이터 추가

```
insert into tbl_member_auth(username, auth)
values
```

```
    ('admin','ROLE_ADMIN'),
    ('admin','ROLE_MANAGER'),
    ('admin','ROLE_MEMBER'),
    ('user0','ROLE_MANAGER'),
    ('user0','ROLE_MEMBER'),
    ('user1','ROLE_MEMBER'),
    ('user2','ROLE_MEMBER'),
    ('user3','ROLE_MEMBER'),
    ('user4','ROLE_MEMBER');
```

```
select * from tbl_member_auth order by auth;
```

- ADMIN(최고 관리자)
 - ROLE_ADMIN, ROLE_MANAGER, ROLE_MEMBER
- MANAGER(일반 관리자)
 - ROLE_MANAGER, ROLE_MEMBER
- MEMBER(일반 회원)
 - ROLE_MEMBER

- **요구사항1** : SecurityConfig.java
 - `UserDetailsService`와 `PasswordEncoder` 사용:
 - 코드 하단에 `auth.userDetailsService(userDetailsService)`와 `auth.passwordEncoder(passwordEncoder())`가 추가
 - `UserDetailsService`를 사용하여 사용자 정보를 데이터베이스나 다른 외부 저장소에서 불러오도록 설정
 - `PasswordEncoder`를 사용하여 비밀번호를 암호화된 형태로 저장 및 검증

- **요구사항2 : SecurityController.java**
 - **doMember** 메서드 (/member 경로)
 - 매핑된 경로: /member
 - 사용자 정보 획득: `Authentication` 객체를 통해 `UserDetails`를 얻어옴.
 - 로그 출력: `userDetails.getUsername()`을 통해 사용자의 사용자명(username)을 로그에 기록.
 - **doAdmin** 메서드 (/admin 경로)
 - 매핑된 경로: /admin
 - 사용자 정보 획득: `@AuthenticationPrincipal` 어노테이션을 사용하여 커스텀 사용자 객체(`CustomUser`)를 직접 주입받음.
 - 로그 출력: `customUser.getMember()`를 호출하여 얻은 `MemberVO` 객체를 로그에 기록.

- **요구사항3 : UserDetailsMapper.xml**
 - resultMap 정의 (memberMap):
 - 객체 타입: MemberVO
 - 매핑 필드:
 - username 필드를 username 컬럼에 매핑
 - password 필드를 password 컬럼에 매핑
 - email 필드를 email 컬럼에 매핑
 - regDate 필드를 reg_date 컬럼에 매핑
 - updateDate 필드를 update_date 컬럼에 매핑
 - 컬렉션 매핑:
 - authList 필드를 authMap resultMap으로 매핑 (연관된 권한 정보를 포함)
 - select 쿼리 (get):
 - 쿼리 내용:
 - tbl_member 테이블에서 username, password, email, reg_date, update_date 필드와 함께 tbl_member_auth 테이블과의 조인 결과로 auth 필드를 선택
 - 조인 조건:
 - tbl_member의 username과 tbl_member_auth의 username을 외부 조인(LEFT OUTER JOIN)으로 연결
 - 조건:
 - m.username = #{username}을 조건으로, 특정 사용자 이름에 대한 정보를 조회

수고하셨습니다!

