KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

# member 테이블 (심화 1)

## [KB] IT's Your Life

KB 국민은행

# member 테이블

## 다음과 같이 사용자 정보 테이블을 생성하세요.

```
-- 사용자 정보 테이블
drop table if exists tbl_member;

create table tbl_member
(
        username        varchar(50) primary key,        -- 사용자 id
        password        varchar(128) not null,          -- 암호화된 비밀번호
        email           varchar(50) not null,
        reg_date        datetime default now(),
        update_date datetime default now()
);

-- 사용자 권한 테이블
drop table if exists tbl_member_auth;

create table tbl_member_auth
(
        username        varchar(50) not null,           -- 사용자 id
        auth            varchar(50) not null,           -- 권한 문자열 ROLE_ADMIN, ROLE_MANAGER, ROLE_MEMBER 등
        primary key(username, auth),                    -- 복합키
        constraint fk_authorities_users foreign key (username) references tbl_member(username)
);
```

# member 테이블

- 앞에서 만든 사용자 정보 테이블과 권한 테이블에 데이터를 추가하세요.

# member 테이블

## 데이터 추가

```
-- 테스트 사용자 추가
insert into tbl_member(username, password, email)
values
		('admin', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lddCyn0nic5dFo3VfJYrXYC', 'admin@galapgos.org'),
		('user0', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lddCyn0nic5dFo3VfJYrXYC', 'user0@galapgos.org'),
		('user1', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lddCyn0nic5dFo3VfJYrXYC', 'user1@galapgos.org'),
		('user2', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lddCyn0nic5dFo3VfJYrXYC', 'user2@galapgos.org'),
		('user3', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lddCyn0nic5dFo3VfJYrXYC', 'user3@galapgos.org'),
		('user4', '$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lddCyn0nic5dFo3VfJYrXYC', 'user4@galapgos.org');


select * from tbl_member;
```

# member 테이블

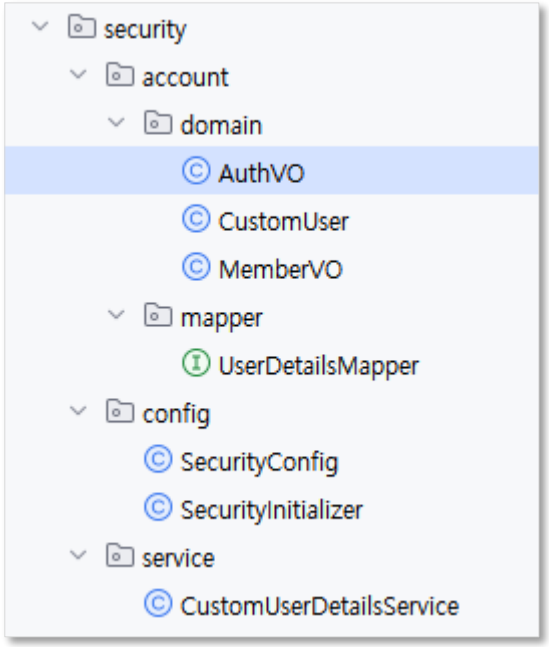
## 권한 데이터 추가

```
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

# member 테이블

- **권한 테이블에 대응하는 AuthVO 클래스를 정의하세요.**
  - GrantedAuthority 인터페이스 구현

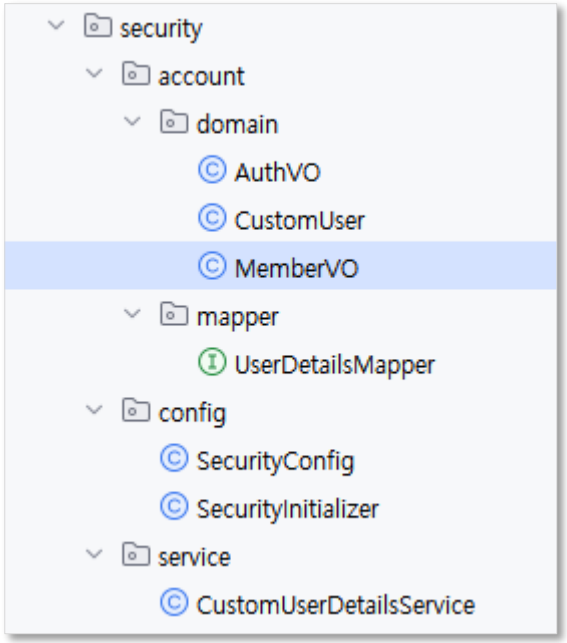


## AuthVO.java

```
@Data
public class AuthVO implements GrantedAuthority {
    private String username;
    private String auth;

    @Override
    public String getAuthority() {
        return auth;
    }
}
```

# member 테이블

- [x] 사용자 정보 테이블에 대응하는 MemberVO를 정의하세요.



# member 테이블

## MemberVO.java

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MemberVO {
		private String username;
		private String password;
		private String email;
		private Date regDate;
		private Date updateDate;

		private List<AuthVO> authList;		// 권한 목록, join 처리 필요

}
```

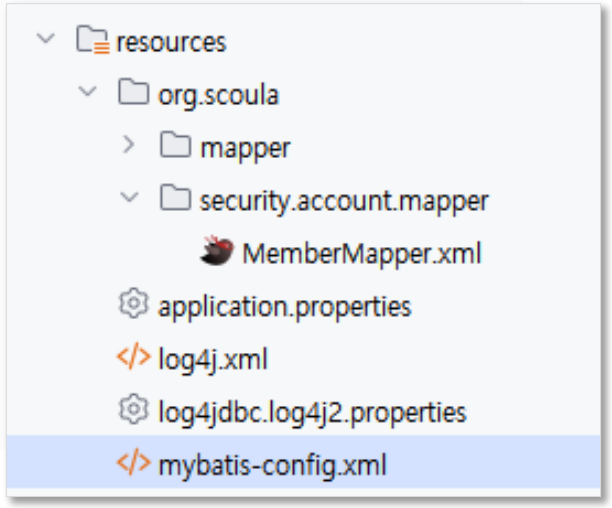
# member 테이블

- **UserDetailsMapper 인터페이스와 UserDetailsMapper.xml을 작성하세요.**
  - 메서드
    - MemberVO get(String username)

# member 테이블

## resources::mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>
   <settings>
      <setting name="mapUnderscoreToCamelCase" value="true" />
   </settings>

   <typeAliases>
      <package name="org.scoula.security.account.domain" />
   </typeAliases>

</configuration>
```

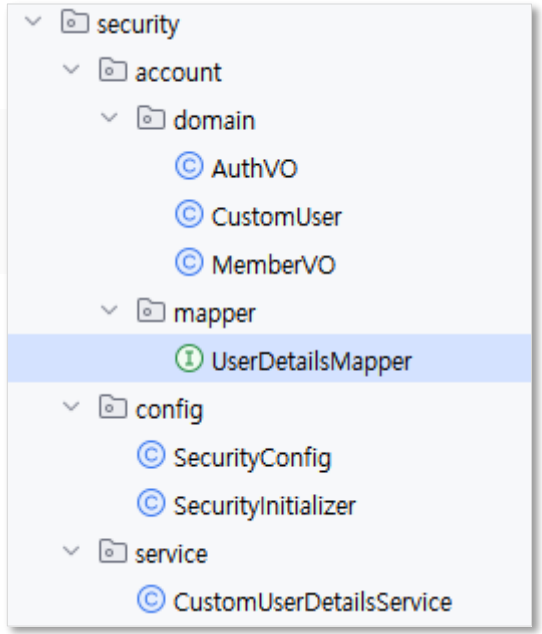


# member 테이블

## UserDetailsMapper.java

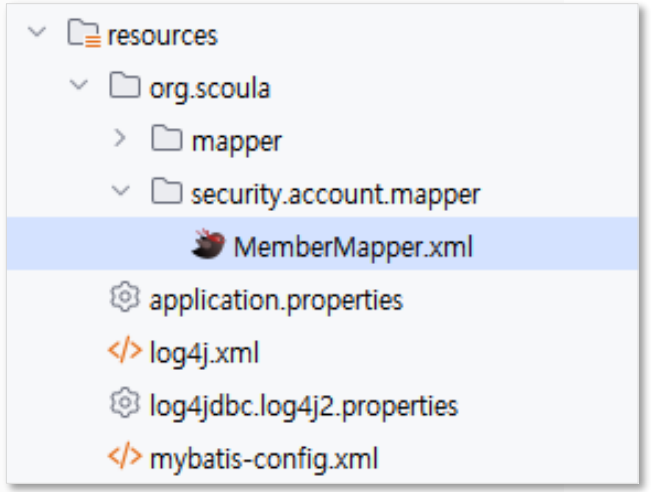
```
public interface UserDetailsMapper {
		public MemberVO get(String username);
}
```

- tbl_membe와 tbl_member_auth를 Join처리
  - MemberVO의 authList를 설정
  - MyBatis의 resultMap으로 Join 쿼리 결과를 MemberVO로 맵핑



## resources :: UserDetailsMapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="org.scoula.security.account.mapper.UserDetailsMapper">

  <resultMap id="authMap" type="AuthVO">
    <result property="username" column="username" />
    <result property="auth"                                    column="auth" />
  </resultMap>

  <resultMap id="memberMap"  type="MemberVO">
    <id property="username"                                                  column="username" />
    <result property="username"                column="username" />
    <result property="password"                column="password" />
    <result property="email"                                   column="email" />
    <result property="regDate"                                 column="reg_date" />
    <result property="updateDate"    column="update_date" />

    <collection property="authList" resultMap="authMap" />
  </resultMap>
```



# member 테이블

## resources :: UserDetailsMapper.xml

```
        <select id="get" resultMap="memberMap">
                select m.username, password, email, reg_date, update_date, auth
                from
                        tbl_member m left outer join tbl_member_auth a
                                on m.username = a.username
                where m.username = #{username}
        </select>
</mapper>
```

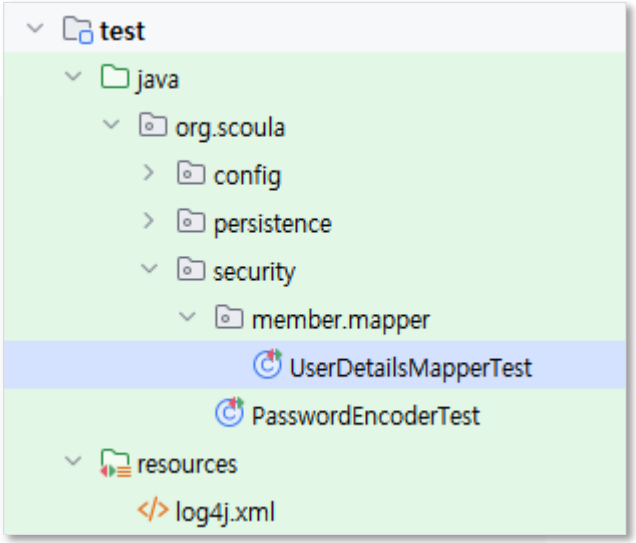
# member 테이블

- SecurityConfig에 UserDetailsMapper가 검색되도록 설정을 추가하고, 단위 테스트를 통해 Mapper의 동작을 확인하세요.

## SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@Log4j2
@MapperScan(basePackages  = {"org.scoula.security.account.mapper"})
public class SecurityConfig extends WebSecurityConfigurerAdapter {
…
}
```

## test :: UserDetailsMapperTest.java




```
@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes = { RootConfig.class, SecurityConfig.class })
@Log4j2
public class UserDetailsMapperTest {
        @Autowired
        private UserDetailsMapper mapper;

        @Test
        public void testGet() {
                MemberVO member = mapper.get("admin");
                log.info(member);

                for(AuthVO auth : member.getAuthList()) {
                        log.info(auth);
                }
        }
}
```

```
INFO : org.scoula.security.account.mapper.UserDetailsMapperTest - MemberVO(username=admin, password=$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lddCyn0nic5dFo3VfJYrXYC, email=admin@galapgos.org, regDate=Wed Jul 24 12:01:52 KST 2024, updateDate=Wed Jul 24 12:01:52 KST 2024, authList=[AuthVO(username=admin, auth=ROLE_ADMIN), AuthVO(username=admin, auth=ROLE_MANAGER), AuthVO(username=admin, auth=ROLE_MEMBER)])
INFO : org.scoula.security.account.mapper.UserDetailsMapperTest - AuthVO(username=admin, auth=ROLE_ADMIN)
INFO : org.scoula.security.account.mapper.UserDetailsMapperTest - AuthVO(username=admin, auth=ROLE_MANAGER)
INFO : org.scoula.security.account.mapper.UserDetailsMapperTest - AuthVO(username=admin, auth=ROLE_MEMBER)
```

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

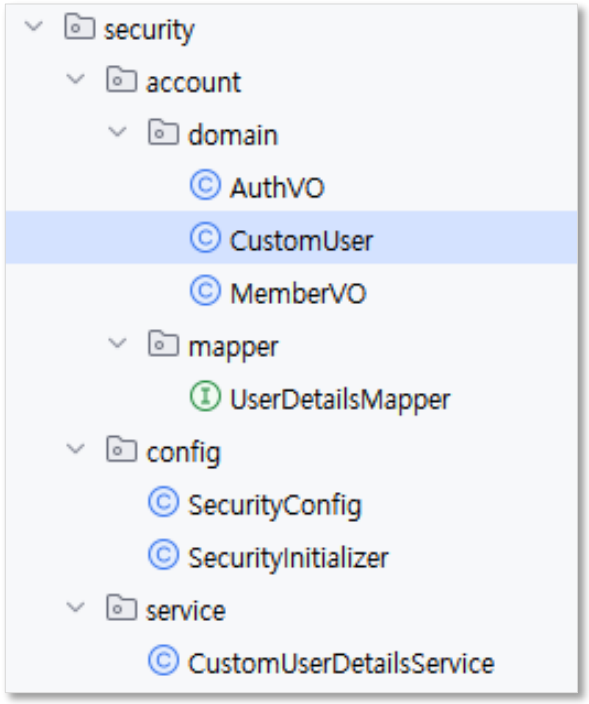
# UserDetails 사용하기 (심화 2)

## [KB] IT's Your Life

KB 국민은행

# UserDetails 사용하기

- **CustomUser 클래스를 정의하세요.**
  - User 클래스 상속



# UserDetails 사용하기

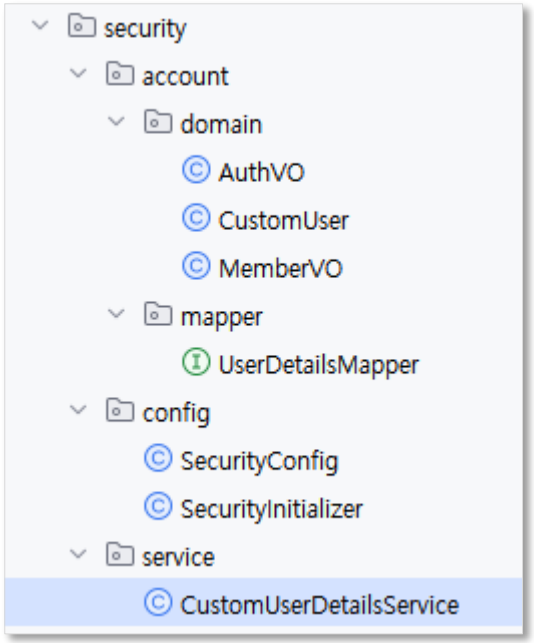
## CustomUser.java

```
@Getter
@Setter
public class CustomUser extends User {
   private MemberVO member;                    // 실질적인 사용자 데이터

   public CustomUser(String username, String password,
                     Collection<? extends GrantedAuthority> authorities) {
      super(username, password, authorities);
   }

   public CustomUser(MemberVO vo) {
      super(vo.getUsername(), vo.getPassword(), vo.getAuthList());
      this.member = vo;
   }

}
```

# UserDetails 사용하기

## CustomUserDetailsService 클래스를 정의하세요.

- UserDetailsService 인터페이스 구현



# UserDetails 사용하기

## CustomUserDetailsService.java

```
package org.scoula.security.service;
…

@Log4j2
@Compnent
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {

        private final UserDetailsMapper mapper;

        @Override
        public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

                MemberVO vo = mapper.get(username);
                if(vo == null) {
                        throw new UsernameNotFoundException(username + "은 없는 id입니다.");
                }

                return new CustomUser(vo);
        }

}
```

# UserDetails 사용하기



- AuthenticationManager에서 인메모리 사용자 정보 대신 UserDetailsService를 이용하도록 수정하세요.

# UserDetails 사용하기

## SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@Log4j2
@MapperScan(basePackages  = {"org.scoula.security.account.mapper"})
@ComponentScan(basePackages  = {"org.scoula.security"})
@RequiredArgsConstructor
public class SecurityConfig extends WebSecurityConfigurerAdapter {

   private final UserDetailsService userDetailsService;

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

 @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
                  // in memory user 정보 삭제  → UserDetailsService와 같이 사용 불가
   auth
       .userDetailsService(userDetailsService)
      .passwordEncoder(passwordEncoder());
  }
}
```

# UserDetails 사용하기



- **컨트롤러에 UserDetail을 얻는 방법 사용하여 확인하세요.**
  - Principal 주입
  - Authentication 주입
  - @AuthenticationPrincipal 주입

# UserDetails 사용하기

## 컨트롤러에서 UserDetails 얻기

### Principal 주입

- getName() 메서드를 통해 username 얻을 수 있음

```
...
import java.security.Principal;

...

@GetMapping("/member")
public void doMember(Pricipal principal) {
   log.info("username = " + principal.getName());
}
```

INFO : org.scoula.controller.SecurityController - username = **user0**

# UserDetails 사용하기

## 컨트롤러에서 UserDetails 얻기

### Authentication 주입

- getPrincipal()을 통해 UserDetails 추출

```
...
import org.springframework.security.core.Authentication;
...

@GetMapping("/member")
public void doMember(Authentication authentication) {
   UserDetails userDetails = (UserDetails)authentication.getPrincipal();

   log.info("username = " + userDetails.getUsername());

}
```

INFO : org.scoula.controller.SecurityController - username = **user0**

# UserDetails 사용하기

## 컨트롤러에서 UserDetails 얻기

### @AuthenticationPincipal 애노테이션

- CustomUser 주입을 요구할 수 있음

```
...
import org.springframework.security.core.annotation.AuthenticationPrincipal;
...

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

INFO : org.scoula.controller.SecurityController - username = MemberVO(username=admin, password=$2a$10$EsIMfxbJ6NuvwX7MDj4WqOYFzLU9U/lddCyn0nic5dFo3VfJYrXYC, email=admin@galapgos.org, regDate=Wed Jul 24 12:01:52 KST 2024, updateDate=Wed Jul 24 12:01:52 KST 2024, authList=[AuthVO(username=admin, auth=ROLE_ADMIN), AuthVO(username=admin, auth=ROLE_MANAGER), AuthVO(username=admin, auth=ROLE_MEMBER)])

# UserDetails 사용하기



- **index.jsp에서 로그인 여부에 따라 화면을 재구성하세요.**
  - 로그인 한 경우
    - username 출력
    - 로그아웃 링크 출력
  - 로그인 안 한 경우
    - 로그인 링크 출력

# UserDetails 사용하기

## index.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://www.springframework.org/security/tags" prefix="sec" %>
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<h1>환영합니다.</h1>

<sec:authorize access="isAnonymous()"> <!-- 로그인 안한 경우 -->
  <a href="/security/login">로그인</a>
</sec:authorize>

<sec:authorize access="isAuthenticated()"><!-- 로그인 한 경우 -->
  <sec:authentication property="principal.username"/>
  <form action="/security/logout" method="post">
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
    <input type="submit" value="로그아웃"/>
  </form>
</sec:authorize>

</body>
</html>
```