

2024년 상반기 K-디지털 트레이닝

UserDetails 사용하기

[KB] IT's Your Life

1 UserDetails 사용하기

✓ UserDetails 인터페이스

- Spring 인증에 필요한 필수 항목 정의 규약

UserDetails

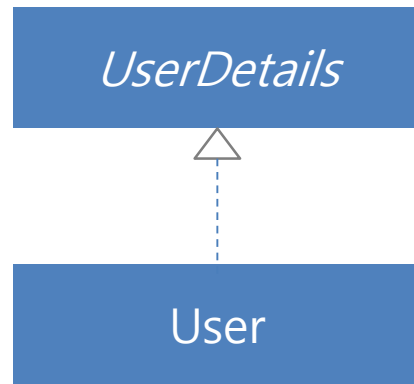
리턴값	메서드	설명
String	getUsername()	사용자 id 리턴
String	getPassword()	비밀번호 리턴
Collection <? extends GrantedAuthority>	getAuthorities()	권한 목록 리턴
boolean	isAccountNonExpired	계정 만료 여부 리턴. true: 유효, false: 만료
boolean	isAccountNonLocked	계정 잠김 여부 리턴. true: 유효, false: 잠김
boolean	isCredentialsNonExpired	신임장 만기 만료 여부 리턴: true: 유효, false: 만료
boolean	isEnabled	계정 사용가능 여부 리턴: true: 사용가능, false: 사용불가능

UserDetails.java

```
public interface UserDetails extends Serializable {  
    Collection<? extends GrantedAuthority> getAuthorities();  
  
    String getPassword();  
  
    String getUsername();  
  
    boolean isAccountNonExpired();  
  
    boolean isAccountNonLocked();  
  
    boolean isCredentialsNonExpired();  
  
    boolean isEnabled();  
}
```

✓ User

- 스프링에서 정의된 UserDetails 구현체
- 스프링 시큐리티에서 요구하는 최소 정보만 유지



User.java

```
public class User implements UserDetails, CredentialsContainer {  
    private static final long serialVersionUID = 580L;  
    private static final Log logger = LoggerFactory.getLog(User.class);  
  
    private String password;  
    private final String username;  
    private final Set<GrantedAuthority> authorities;  
  
    private final boolean accountNonExpired;  
    private final boolean accountNonLocked;  
    private final boolean credentialsNonExpired;  
    private final boolean enabled;
```

User.java

```
public User(String username, String password, Collection<? extends GrantedAuthority> authorities) {
    this(username, password, true, true, true, true, authorities);
}

public User(String username, String password, boolean enabled, boolean accountNonExpired,
    boolean credentialsNonExpired, boolean accountNonLocked,
    Collection<? extends GrantedAuthority> authorities) {
    Assert.isTrue(username != null && !"".equals(username) && password != null,
        "Cannot pass null or empty values to constructor");
    this.username = username;
    this.password = password;
    this.enabled = enabled;
    this.accountNonExpired = accountNonExpired;
    this.credentialsNonExpired = credentialsNonExpired;
    this.accountNonLocked = accountNonLocked;
    this.authorities = Collections.unmodifiableSet(sortAuthorities(authorities));
}
```

User.java

```
public Collection<GrantedAuthority> getAuthorities() { return this.authorities; }

public String getPassword() { return this.password; }

public String getUsername() { return this.username; }

public boolean isEnabled() { return this.enabled; }

public boolean isAccountNonExpired() { return this.accountNonExpired; }

public boolean isAccountNonLocked() { return this.accountNonLocked; }

public boolean isCredentialsNonExpired() { return this.credentialsNonExpired; }

public void eraseCredentials() {this.password = null; }
```

User.java

```
private static SortedSet<GrantedAuthority> sortAuthorities(Collection<? extends GrantedAuthority> authorities) {
    Assert.notNull(authorities, "Cannot pass a null GrantedAuthority collection");
    SortedSet<GrantedAuthority> sortedAuthorities = new TreeSet(new AuthorityComparator());
    Iterator var2 = authorities.iterator();

    while(var2.hasNext()) {
        GrantedAuthority grantedAuthority = (GrantedAuthority)var2.next();
        Assert.notNull(grantedAuthority, "GrantedAuthority list cannot contain any null elements");
        sortedAuthorities.add(grantedAuthority);
    }

    return sortedAuthorities;
}
```


User.java

```
public static UserBuilder builder() {  
    return new UserBuilder();  
}  
  
public static UserBuilder withUsername(String username) {  
    return builder().username(username);  
}  
  
public static UserBuilder withUserDetails(UserDetails userDetails) {  
    return  
withUsername(userDetails.getUsername()).password(userDetails.getPassword()).accountExpired(!userDetails.isAccountNonExpired()).accountLocked(!userDetails.isAccountNonLocked()).authorities(userDetails.getAuthorities()).credentialsExpired(!userDetails.isCredentialsNonExpired()).disabled(!userDetails.isEnabled());  
}
```

User.java

```
public static final class UserBuilder {  
    private String username;  
    private String password;  
    private List<GrantedAuthority> authorities;  
    private boolean accountExpired;  
    private boolean accountLocked;  
    private boolean credentialsExpired;  
    private boolean disabled;  
    private Function<String, String> passwordEncoder;  
  
    private UserBuilder() {  
        this.passwordEncoder = (password) -> {  
            return password;  
        };  
    }  
    ...  
}
```

User.java

```
public UserDetails build() {
    String encodedPassword = (String)this.passwordEncoder.apply(this.password);
    return new User(this.username, encodedPassword,
        !this.disabled, !this.accountExpired, !this.credentialsExpired, !this.accountLocked,
        this.authorities);
}

private static class AuthorityComparator implements Comparator<GrantedAuthority>, Serializable {
    ...
}
}
```

✓ UserDetailsService 인터페이스

○ 로그인 사용자의 상세 내역을 제공하는 규약 정의

```
public interface UserDetailsService {  
    UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;  
}
```

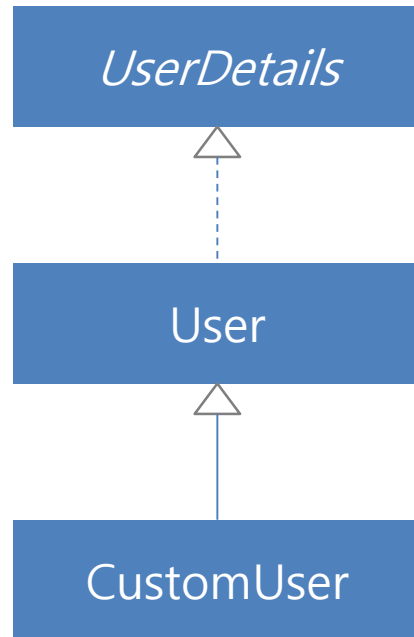
- 매개변수: 찾고자 하는 username 문자열
- 리턴값: 찾은 UserDetails 객체
- 예외: 찾고자 하는 username이 없는 경우 UsernameNotFoundException 발생

○ UserDetailsService 인터페이스 구현

- UserDetailsMapper 주입으로 멤버 정보 추출

✓ CustomUser

- User 클래스 상속
- 애플리케이션에서 필요한 사용자 추가 정보(MemberVO) 관리

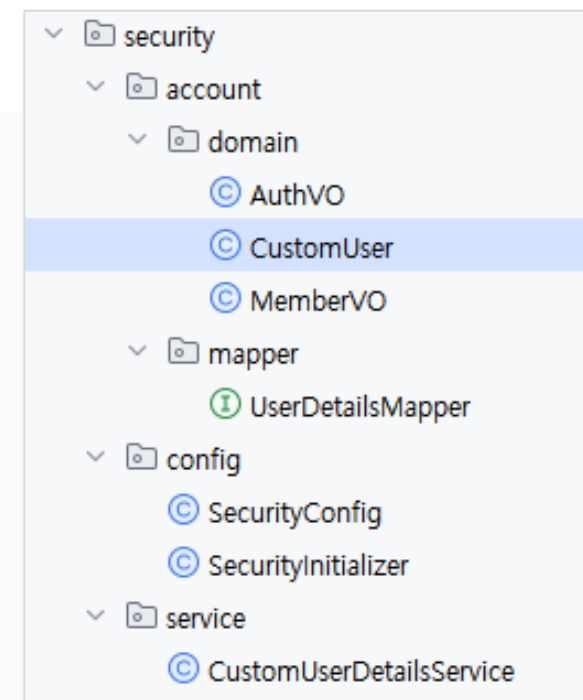


CustomUser.java

@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;  
    }  
}
```



CustomUserDetailsService.java

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

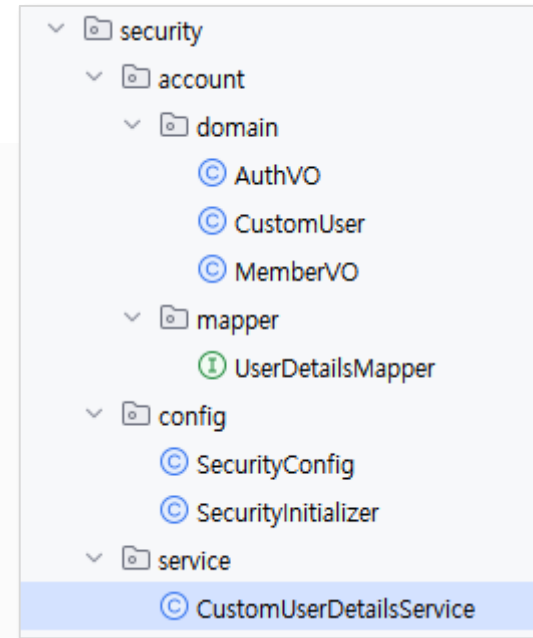
@Log4j
@Component
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {

    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);
    }
}
```



✓ 스프링 시큐리티 설정

- CustomUserDetailsService을 Bean으로 등록
- AuthenticationManagerBuilder에서 CustomUserDetailsService를 설정

SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@Log4j
@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());
    }
}
```

✓ 컨트롤러에서 UserDetails 얻기

○ Principal 주입

- getName() 메서드를 통해 username 얻을 수 있음

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

...

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

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

✓ 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**

✓ 컨트롤러에서 UserDetails 얻기

○ @AuthenticationPrincipal 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)])

✓ spring security tag lib 사용

```
<%@ taglib uri="http://www.springframework.org/security/tags" prefix="sec" %>
```

✓ <sec:authentication>

○ 인증이 된 경우 사용자 정보 출력

- <sec:authentication property="principal">
 - principal : UserDetailsService가 리턴한 객체
--> loadUserByUsername()에서 반환한 User 객체
 - 로그인 사용자명 출력

```
<sec:authentication property="principal.username"/>
```

✓ <sec:authorize access="">

○ 인증 여부 판단 및 접근 권한 체크

- **isAnonymous()** 로그인하지 않은 경우 true
- **isAuthenticated()** 로그인한 경우 true
- **isFullyAuthenticated()** remember-me 이외의 경로로 인증된 경우 true
- **isRememberMe()** remember-me로 인증된 사용자인 경우 true
- **hasRole([role])** 현재 주체에 지정된 역할이 있는지 여부.
역할이 'ROLE_'로 시작하지 않으면 자동 추가
- **hasAnyRole([role1,role2])** 제공된 역할 목록에 있는지 여부
- **hasAuthority([authority])** 현재 주체에 지정된 권한.
- **hasAnyAuthority([authority1,authority2])**
- **principal** 현재 사용자를 나타내는 주체 개체
- **authentication** SecurityContext Authentication.
- **permitAll** 항상 허용
- **denyAll** 항상 불허

✓ 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>
```

환영합니다.

[로그인](#)

환영합니다.

admin

로그아웃