# Spring Security

## build.gradle setting

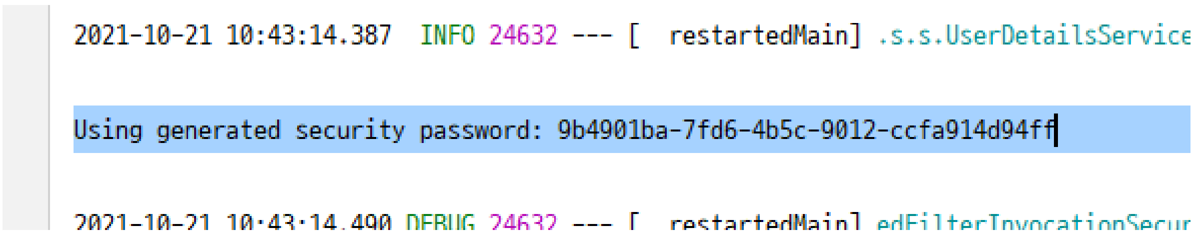
```
implementation 'org.springframework.boot:spring-boot-starter-security'
```

## application.properties

```
logging.level.org.springframework.security=INFO
```

## password 확인

설정하고 Application 실행시키면 기본 발급되는 password 확인 가능



```
2021-10-21 10:43:14.387  INFO 24632 --- [  restartedMain] .s.s.UserDetailsService

Using generated security password: 9b4901ba-7fd6-4b5c-9012-ccfa914d94ff

2021-10-21 10:43:14.490 DEBUG 24632 --- [  restartedMain] edFilterInvocationSecur
```

## CustomSecurityConfig 생성

```
@Configuration
@Log4j2
@EnableGlobalMethodSecurity(prePostEnabled = true)
@RequiredArgsConstructor
public class CustomSecurityConfig  extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        log.info("CustomSecurityConfig...configure.........");
        log.info("CustomSecurityConfig...configure.........");
        log.info("CustomSecurityConfig...configure.........");
        log.info("CustomSecurityConfig...configure.........");
    }
}
```

## 확인

```
Using generated security password: dc1ddcbb-1a43-49db-a00f-d90c6930841f

2021-10-21 10:50:57.787  INFO 26408 --- [  restartedMain] o.zerock.sb.config.CustomSecurityConfig  : CustomSecurityConfig...configure.........
2021-10-21 10:50:57.788  INFO 26408 --- [  restartedMain] o.zerock.sb.config.CustomSecurityConfig  : CustomSecurityConfig...configure.........
2021-10-21 10:50:57.788  INFO 26408 --- [  restartedMain] o.zerock.sb.config.CustomSecurityConfig  : CustomSecurityConfig...configure.........
2021-10-21 10:50:57.788  INFO 26408 --- [  restartedMain] o.zerock.sb.config.CustomSecurityConfig  : CustomSecurityConfig...configure.........
```

## Bcrypt 암호화 추가

```
public class CustomSecurityConfig  extends WebSecurityConfigurerAdapter {

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

}
```

## Member에 method 추가

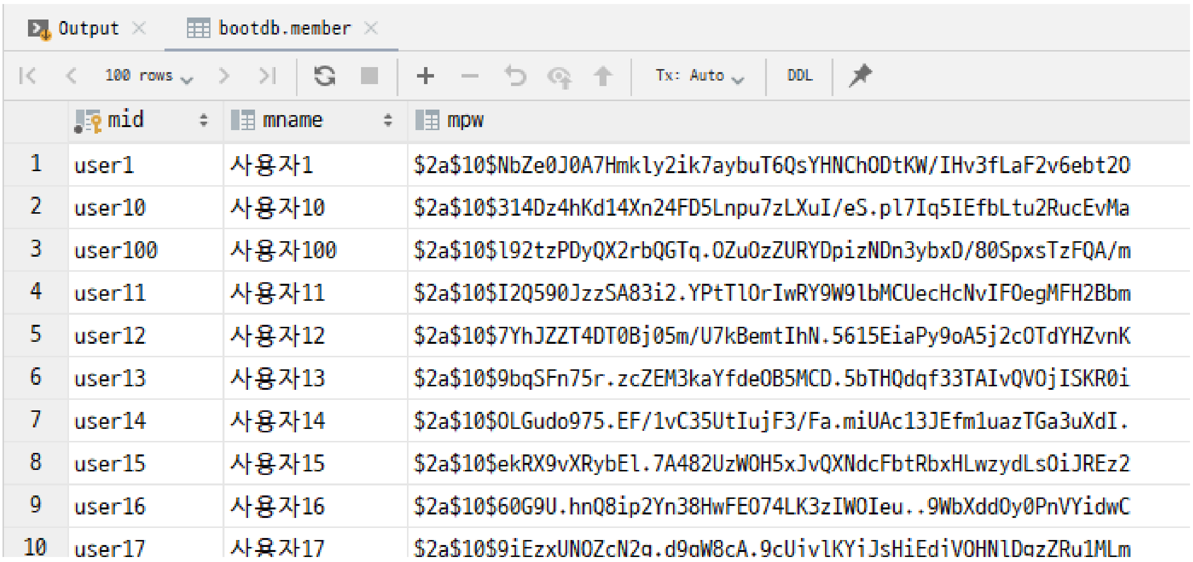
```
public class Member {

    //암호는 변경 가능해야하므로
    public void changePassword(String password) {
        this.mpw = password;
    }

}
```

## MemberRepository에서 test

```
@Autowired
private PasswordEncoder passwordEncoder;


@Test
public void updateMembers() {

    List<Member> memberList = memberRepository.findAll();

    //암호화 된 비밀번호로 변경
```

```
        memberList.forEach(member -> {
            member.changePassword(passwordEncoder.encode("1111"));
            memberRepository.save(member);
        });
    }
```

## 결과



| | mid | mname | mpw |
|---|---|---|---|
| 1 | user1 | 사용자1 | $2a$10$NbZe0J0A7Hmkly2ik7aybuT6QsYHNChODtKW/IHv3fLaF2v6ebt2O |
| 2 | user10 | 사용자10 | $2a$10$314Dz4hKd14Xn24FD5Lnpu7zLXuI/eS.pl7Iq5IEfbLtu2RucEvMa |
| 3 | user100 | 사용자100 | $2a$10$l92tzPDyQX2rbQGTq.OZuOzZURYDpizNDn3ybxD/80SpxsTzFQA/m |
| 4 | user11 | 사용자11 | $2a$10$I2Q590JzzSA83i2.YPtTlOrIwRY9W9lbMCUecHcNvIFOegMFH2Bbm |
| 5 | user12 | 사용자12 | $2a$10$7YhJZZT4DT0Bj05m/U7kBemtIhN.5615EiaPy9oA5j2cOTdYHZvnK |
| 6 | user13 | 사용자13 | $2a$10$9bqSFn75r.zcZEM3kaYfdeOB5MCD.5bTHQdqf33TAIvQVOjISKR0i |
| 7 | user14 | 사용자14 | $2a$10$OLGudo975.EF/1vC35UtIujF3/Fa.miUAc13JEfm1uazTGa3uXdI. |
| 8 | user15 | 사용자15 | $2a$10$ekRX9vXRybEl.7A482UzWOH5xJvQXNdcFbtRbxHLwzydLsOiJREz2 |
| 9 | user16 | 사용자16 | $2a$10$60G9U.hnQ8ip2Yn38HwFEO74LK3zIWOIeu..9WbXddOy0PnVYidwC |
| 10 | user17 | 사용자17 | $2a$10$9iEzxUNOZcN2q.d9qW8cA.9cUivlKYiJsHiEdiVOHNlDqzZRu1MLm |

더이상 Config 에서 UserDedetailsService 사용하지 않아도됨 - 자동으로 잡힘

## CustomSecurityConfig 수정

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    log.info("CustomSecurityConfig...configure.........");
    http.formLogin().loginPage("/customLogin").loginProcessingUrl("/login"); //인가/인증에 문제시 로그인 화면
    http.csrf().disable();
    http.logout();
}
```

```
@EnableGlobalMethodSecurity(prePostEnabled = true)
```

preAutorize 바로 사용 가능하도록 설정하는 것

### BoardController 수정

```
@PreAuthorize("isAuthenticated()")
@GetMapping("/register")
public void register() {
    log.info("----------- c register -----------");
    log.info("----------- c register -----------");
    log.info("----------- c register -----------");
}
```

인증된 사용자만 접근할 수 있도록 추가

`localhost:8080/board/register` 로 접속하면 `/customLogin` 으로 튕겨냄

### MemberController 생성

```
@Controller
@Log4j2
@RequiredArgsConstructor
public class MemberController {

    @GetMapping("/customLogin")
    public void loginInput(){
        log.info("custom Login Page....");
        log.info("custom Login Page....");
        log.info("custom Login Page....");
    }

}
```

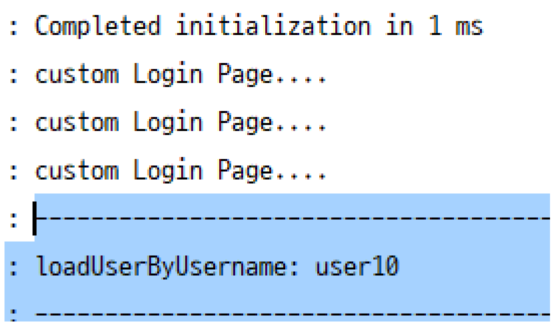
### customLogin.html 생성

```
<div class="wrapper fadeInDown">
    <div id="formContent">
        <!-- Tabs Titles -->

        <!-- Icon -->
        <div class="fadeIn first">
        </div>

        <!-- Login Form -->
        <form action="/login" method="post">
            <input type="text" id="login" class="fadeIn second" name="username"
placeholder="USER ID">
            <input type="text" id="password" class="fadeIn third"
name="password" placeholder="USER PW">
            <input type="submit" class="fadeIn fourth" value="Log In">
        </form>
        <!-- Remind Passowrd -->
        <div id="formFooter">
            <a class="underlineHover" href="#">Forgot Password?</a>
        </div>

    </div>
</div>
```

## CustomUserDetailsService 생성

```
@Service
@Log4j2
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        //로그인버튼 눌렀을 때 동작하는지 확인 먼저
        log.info("---------------------------------------------------------");
        log.info("loadUserByUsername: " + username);
        log.info("---------------------------------------------------------");

        return null;
    }

}
```

## 확인



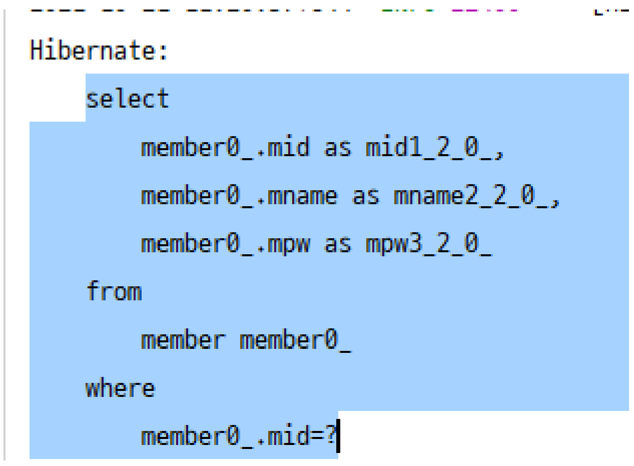
```
: Completed initialization in 1 ms
: custom Login Page....
: custom Login Page....
: custom Login Page....
: -------------------------------------
: loadUserByUsername: user10
: -------------------------------------
```

`return null`이라서 오류는 발생하지만 사용자 이름이 찍히는 것 확인

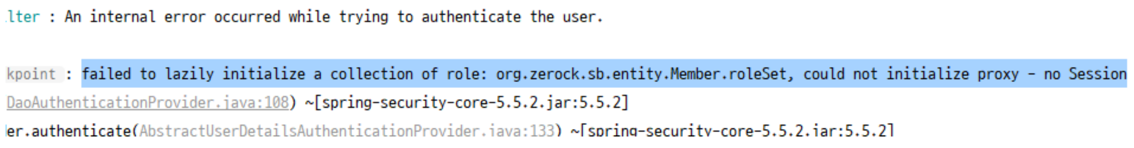
## 수정

```java
@Override
public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
    log.info("----------------------------------------------------------");
    log.info("loadUserByUsername: " + username);
    log.info("----------------------------------------------------------");

    Optional<Member> optionalMember = memberRepository.findById(username);

    //사용자의 계정이 존재하지 않을 경우 예외던지도록 설정
    Member member = optionalMember.orElseThrow(() -> new
UsernameNotFoundException("USER NOT FOUND"));

    log.info("Member: " + member);

    log.info("----------------------------------------------------------");

    return null;
}
```

## 결과



```
Hibernate:
    select
        member0_.mid as mid1_2_0_,
        member0_.mname as mname2_2_0_,
        member0_.mpw as mpw3_2_0_
    from
        member member0_
    where
        member0_.mid=?
```



```
lter : An internal error occurred while trying to authenticate the user.

kpoint : failed to lazily initialize a collection of role: org.zerock.sb.entity.Member.roleSet, could not initialize proxy - no Session
DaoAuthenticationProvider.java:108) ~[spring-security-core-5.5.2.jar:5.5.2]
er.authenticate(AbstractUserDetailsAuthenticationProvider.java:133) ~[spring-security-core-5.5.2.jar:5.5.2]
```

아까완 다르게 select문이 날아갔지만 no session 에러가 발생함 - **EAGER** 설정 필요

## @EntityGraph 사용해서 해결하기

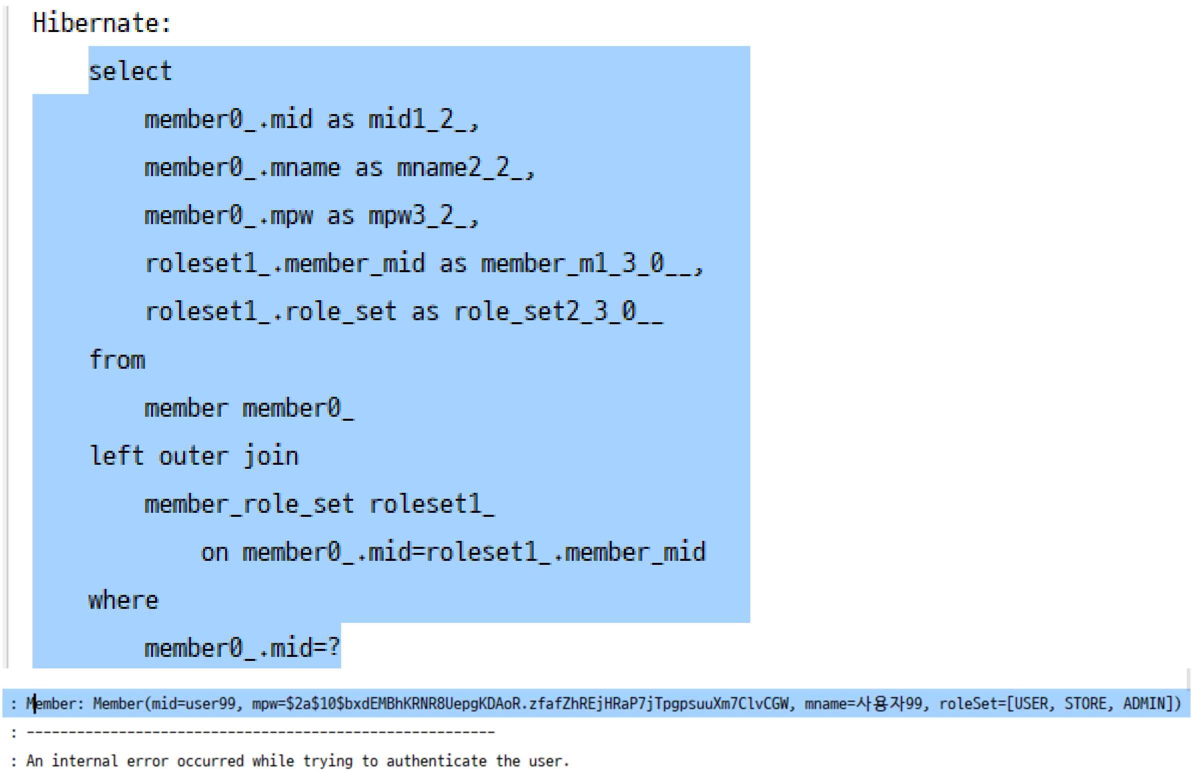
### MemberRepository 수정

```
public interface MemberRepository extends JpaRepository<Member, String> {

    //기본을 LAZY로 두고 필요할 때마다 설정
    @EntityGraph(attributePaths = "roleSet")
    @Query("select m from Member m where m.mid = :mid")
    Optional<Member> getMemberEager(String mid);

}
```

### CustomUserDetailsService 수정

```
Optional<Member> optionalMember = memberRepository.getMemberEager(username);
```

### 결과



```
Hibernate:
    select
        member0_.mid as mid1_2_,
        member0_.mname as mname2_2_,
        member0_.mpw as mpw3_2_,
        roleset1_.member_mid as member_m1_3_0__,
        roleset1_.role_set as role_set2_3_0__
    from
        member member0_
    left outer join
        member_role_set roleset1_
            on member0_.mid=roleset1_.member_mid
    where
        member0_.mid=?
: Member: Member(mid=user99, mpw=$2a$10$bxdEMBhKRNR8UepgKDAoR.zfafZhREjHRaP7jTpgpsuuXm7ClvCGW, mname=사용자99, roleSet=[USER, STORE, ADMIN])
: -------------------------------------------------------
: An internal error occurred while trying to authenticate the user.
```

LAZY loading이 걸려있음에도 join 등을 사용해서 사용자의 정보까지 잘 불러와지는 것 확인 가능

limit가 걸리지 않는 단점

### MemberDTO 생성

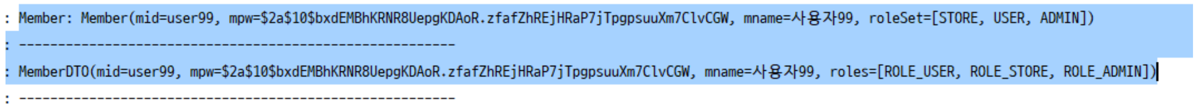
```java
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class MemberDTO implements UserDetails {

    private String mid;

    private String mpw;

    private String mname;

    private Set<SimpleGrantedAuthority> roles;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return roles;
    }

    @Override
    public String getPassword() {
        return mpw;
    }

    @Override
    public String getUsername() {
        return mid;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }

}
```

***UserDetails*** 상속해서 method 정의해주기

## CustomUserDetailsService 수정

```
@Override
public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
    log.info("--------------------------------------------------------");
    MemberDTO memberDTO = MemberDTO.builder()
            .mid(member.getMid())
            .mpw(member.getMpw())
            .mname(member.getMname())
            .roles(member.getRoleSet().stream().map(memberRole -> new
SimpleGrantedAuthority("ROLE_"+memberRole.name())).collect(Collectors.toSet()))
            .build();

    log.info(memberDTO);
    log.info("--------------------------------------------------------");

    return memberDTO;
}
```

## 결과



```
: Member: Member(mid=user99, mpw=$2a$10$bxdEMBhKRNR8UepgKDAoR.zfafZhREjHRaP7jTpgpsuuXm7ClvCGW, mname=사용자99, roleSet=[STORE, USER, ADMIN])
: -------------------------------------------------------
: MemberDTO(mid=user99, mpw=$2a$10$bxdEMBhKRNR8UepgKDAoR.zfafZhREjHRaP7jTpgpsuuXm7ClvCGW, mname=사용자99, roles=[ROLE_USER, ROLE_STORE, ROLE_ADMIN])
: -------------------------------------------------------
```

이제 로그인화면에서 로그인하면 넘어가고 Member, MemberDTO 출력 확인 가능