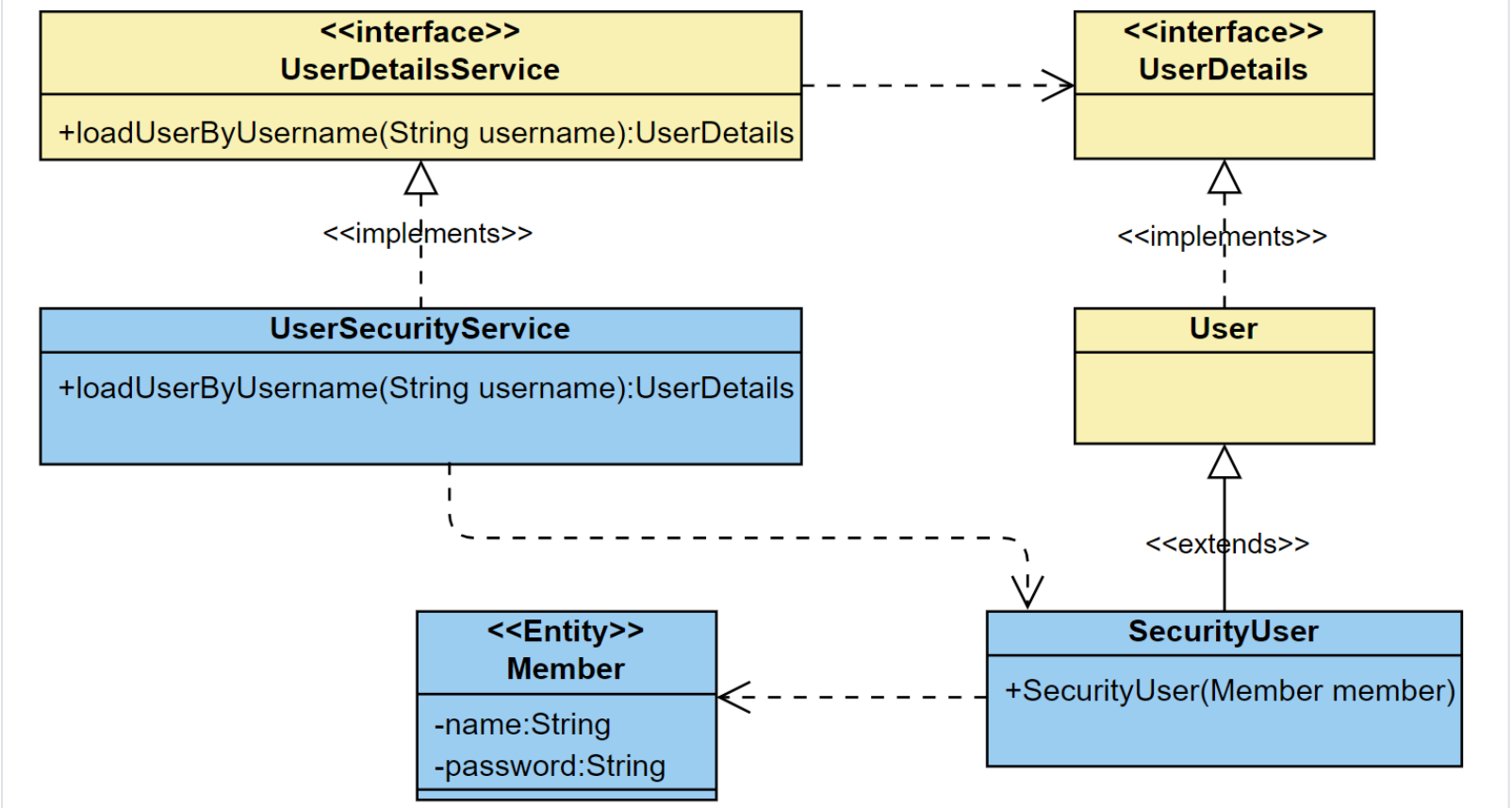
스프링부트 시큐리티를 이용한 *Login* 구현하기

0. 스프링부트 시큐리티 구조



1. 스프링부트 시큐리티 dependency 설정

//파일위치: ../pom.xml

```
    <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
```

스프링부트 시큐리티 프레임워크를 사용하기 위해 위의 dependency를 `pom.xml`에 설정한다.

2. login.html

`login.html`은 다음과 같으며 `/templates` 폴더에 위치한다.

//파일위치: ../templates/login.html

```
<div class="container my-3">
<form th:action="@{/login}" method="post">
    <p style="font-size: 40px;" class="text-left">로그인</p>
    <div th:if="${param.error}">
        <p class="text-warning bg-dark text-center">사용자ID 또는 비밀번호를 확인해 주세요.</p>
    </div>
    <div class="mb-3 col-sm-6">
        <label for="username" class="form-label">사용자ID</label>
        <input type="text" name="username" id="username" class="form-control">
    </div>
    <div class="mb-3  col-sm-6" >
        <label for="password" class="form-label">비밀번호</label>
        <input type="password" name="password" id="password" class="form-control">
    </div>
    <div class="mb-3  col-sm-6">
        <button type="submit" class="btn btn-primary">로그인</button>
    </div>
</form>
</div>
```

3. SecurityController

`SecurityController`에서 `/login` 매핑을 처리한다.

```
//파일위치: ../controller/SecurityController.java
@GetMapping("/login") //templates/login.html 실행
public String login() {
    return "/login";
}
```

로그인을 처리하는 @PostMapping 방식의 메서드는 스프링 시큐리티가 담당하므로 구현할 필요 없다.

4. Member

//파일위치: ../domain/Member.java

```java
@Getter
@Setter
@ToString
@Entity
@Table(name="member_login")
public class Member {
    @Id
    private String id;
    private String name;
    private String password;
    @Enumerated(EnumType.STRING)
    private Role role;
}
```

5. Role

//파일위치: ../domain/Role.java

```java
@Getter
public enum Role {
    ADMIN("ROLE_ADMIN"),
    MEMBER("ROLE_MEMBER"),
    MANAGER("ROLE_MANAGER");

    Role(String value) {
        this.value = value;
    }

    private String value;
}
```

스프링 시큐리티는 **인증(Authentication)** 과 **인가(Authorization)** 를 관리한다.
사용자Id와 패스워드로 **인증**이 된 후 특정 페이지를 접근하기 위해 **인가(권한)**가 필요하다.
`Role`클래스는 ADMIN, MEMBER, MANAGER 3개의 권한을 제공한다. 권한에 따라 접근할 수 있는 서버 자원(Resource)가 다르다.

6. MemberRepository

//파일위치: ../repository/MemberRepository.java

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

7. MemberRepositoryTest

//파일위치: /test/...package.../MemberRepositoryTest.java

```java
@SpringBootTest
class MemberRepositoryTest {

    @Autowired
    private MemberRepository memberRepository;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Test
    @DisplayName("Member 데이터 생성")
    void save() {

        // admin 계정
        // given
        Member admin = new Member();
        admin.setId("admin");
        admin.setName("admin");
        admin.setRole(Role.ADMIN);
        admin.setPassword(passwordEncoder.encode("1234"));

        // when
        Member savedMember1 = memberRepository.save(admin);
        // then
        assertEquals(admin.getName(), savedMember1.getName());


        //member 계정
        Member member = new Member();
        member.setId("member");
        member.setName("member");
        member.setRole(Role.MEMBER);
        member.setPassword(passwordEncoder.encode("1234"));
        // when
        Member savedMember2 = memberRepository.save(member);
        // then
        assertEquals(member.getName(), savedMember2.getName());


        // manager 계정
        // given
        Member manager = new Member();
        manager.setId("manager");
        manager.setName("manager");
        manager.setRole(Role.MANAGER);
        manager.setPassword(passwordEncoder.encode("1234"));

        // when
        Member savedMember3 = memberRepository.save(manager);
        // then
        assertEquals(manager.getName(), savedMember3.getName());
    }
}
```

8. SecurityUser

//파일위치: ../config/SecurityUser.java

```java
public class SecurityUser extends User {
    private static final long serialVersionUID = 1L;

    public SecurityUser(Member member) {
        super(member.getId(), member.getPassword(),
                AuthorityUtils.createAuthorityList(member.getRole().toString()));
        System.out.println(member.getId());
    }
}
```

9. UserSecurityService

//파일위치: ../service/UserSecurityService.java

```
@Service
public class UserSecurityService implements UserDetailsService {

    @Autowired
    private MemberRepository memberRepo;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        Optional<Member> optional = memberRepo.findById(username);
        if (optional.isEmpty()) {
            throw new UsernameNotFoundException(username + " 사용자 없음");
        } else {
            Member member = optional.get();
            System.out.println(member.getRole());
            return new SecurityUser(member);
        }
    }
}
```

UserSecurityService는 스프링 시큐리티가 제공하는 UserDetailsService 인터페이스를 구현(implements) 한다.
스프링 시큐리티의 UserDetailsService는 loadUserByUsername 메서드를 구현하도록 강제하는 인터페이스이다

`public UserDetails loadUserByUsername(String username)`

10. login URL 등록

로그인 페이지의 URL은 `/login`이고 로그인 성공시에 이동하는 디폴트 페이지는 `/loginSuccess`이다.
`login.html`과 `loginSuccess.html`은 `SecurityConfig.java`에 등록한다.

```
        //파일위치: ../config/SecurityConfig.java
        http
        .formLogin()
        .loginPage("/login") // 로그인 페이지 login.html
        .defaultSuccessUrl("/loginSuccess", true)  //로그인 성공시 이동 url
loginSuccess.html
```

11. SecurityConfig

//파일위치: ../config/SecurityConfig.java

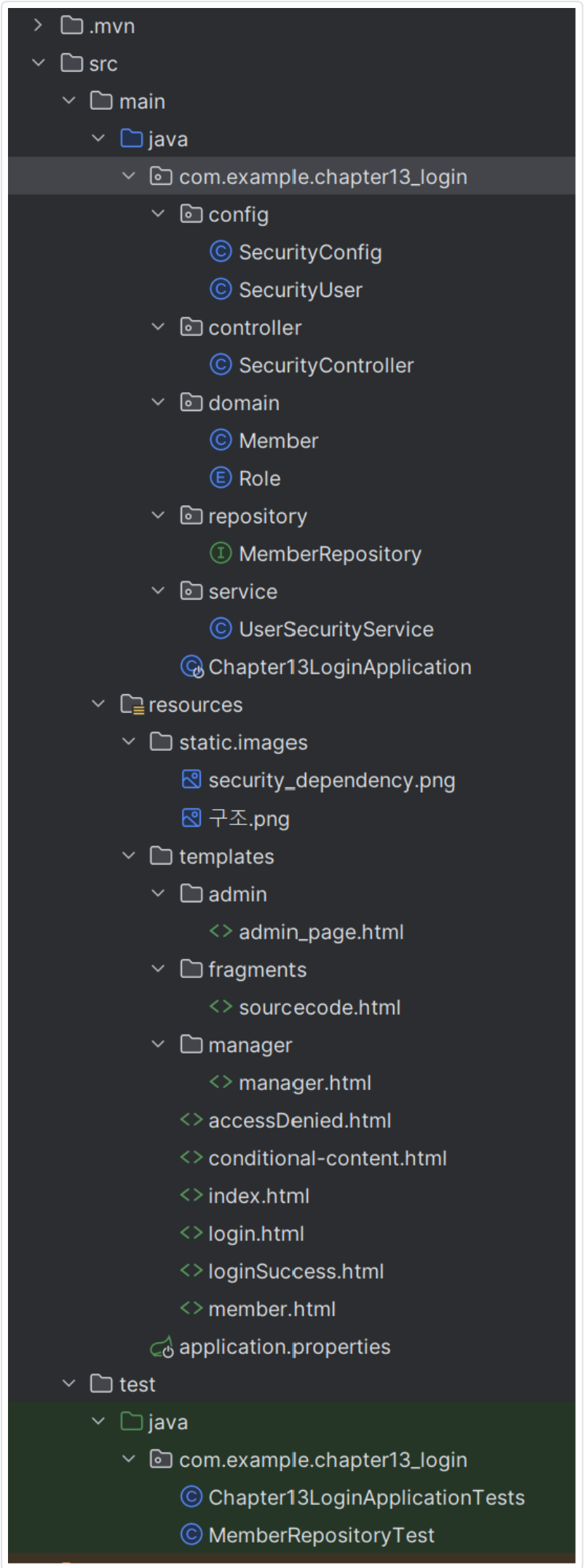
```java
@Configuration
public class SecurityConfig {
    @Autowired
    private UserSecurityService boardUserDetailsService;

    //spring security 5.0 사용
    @Bean
    public SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {
        http
                .formLogin()
                .loginPage("/login")
                .defaultSuccessUrl("/loginSuccess", true)  //로그인 성공시 이동 url
                .and()
                .logout()
                .invalidateHttpSession(true)
                .logoutSuccessUrl("/logout")
                .and()
                .authorizeRequests(authorizeRequests ->
                        authorizeRequests
                                .antMatchers("/", "/login").permitAll()
                                .antMatchers("/admin/**").hasRole("ADMIN")
                                .antMatchers("/manager/**").hasRole("MANAGER")
                                .anyRequest().authenticated()  //그 외 request는 모두 authenticated되어야함
                )
                .csrf().disable()
                .exceptionHandling(exceptionHandling ->
                        exceptionHandling
                                .accessDeniedPage("/accessDenied")
                )
                .userDetailsService(boardUserDetailsService);

        return http.build();
    }

    //패스워드 암호화 처리
    @Bean
    public PasswordEncoder passwordEncoder() {
        return PasswordEncoderFactories.createDelegatingPasswordEncoder();
    }
}
```

12. 인텔리제이 모듈 구조



13. `member_login` 테이블

springboot.member_login: 3 행 (총) (대략적)

id	name	password	role
admin	admin	{bcrypt}$2a$10$2...	ADMIN
manager	manager	{bcrypt}$2a$10$A...	MANAGER
member	member	{bcrypt}$2a$10$cJ...	MEMBER