

7장. 스프링부트 시큐리티



security



Spring Boot

▪ 인증과 인가

인증(Authentication)과 인가(Authorization)

인증을 통해 사용자를 식별하고, 인가를 통해 시스템 자원에 대한 접근을 통제한다.


어떤 직원이 회사 건물에 들어가기 위해서는 사원증이나 RFID 카드를 이용해서 반드시 인증에 통과해야 한다. 인증에 실패한 사람이 회사 건물에 들어가려고 하면 당연히 보안 직원이 제제할 것이다.

그리고 직급과 직무에 따라 부여된 권한이 다르기 때문에 회사 내에서 열람할 수 있는 문서의 종류도 제한되어 있다.

이렇게 직원이 특정 자원에 접근할 때 적절한 권한이 있는지 확인하는 과정을 인가라고 할 수 있다.



▪ BootSecurity 프로젝트 생성

New Spring Starter Project 

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

```
# Security log level Setting  
logging.level.org.springframework.security=debug
```



스프링부트 시큐리티

- 시큐리티 적용

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

```
<dependency>  
  <groupId>org.springframework.security</groupId>  
  <artifactId>spring-security-test</artifactId>  
  <scope>test</scope>  
</dependency>
```

```
2022-09-24 07:31:51.249 WARN 1424 --- [ restartedMain] ion$DefaultTemplateResol  
2022-09-24 07:31:51.331 WARN 1424 --- [ restartedMain] .s.s.UserDetailsService/
```

Using generated security password: e3eb1a8b-5d57-4f49-afba-49b18712a0ff

This generated password is for development use only. Your security configuration



스프링부트 시큐리티

- 시큐리티 적용

/static/hello.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="utf-8">
  <title>Hello~</title>
</head>
<body>
  <h2>Hello~ 스프링 부트!!</h2>
</body>
</html>
```



스프링부트 시큐리티

- 시큐리티 적용

Please sign in

Please sign in

자격 증명에 실패하였습니다.

← → ↻ ⓘ localhost:8080/hello.html

Hello~ 스프링 부트!!



스프링부트 시큐리티

- 시큐리티 커스터마이징

1. 시큐리티 설정파일

```
package com.boot.config;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter{

    @Override
    protected void configure(HttpSecurity http) throws Exception {

    }

}
```

SecurityConfig 클래스가 빈으로 등록되기만 해도 애플리케이션에서는 더 이상 로그인을 강제하지 않는다.



- 시큐리티 커스터마이징

2. 시큐리티 화면 구성

요청 URL	의미
<code>"/</code>	인증을 하지 않은 모든 사용자가 접근할 수 있다.



스프링부트 시큐리티

- SecurityController 클래스

```
@Slf4j
@Controller
public class SecurityController {

    @GetMapping("/")
    public String index() {
        Log.info("index 요청입니다.");
        return "index";
    }

    @GetMapping("/member")
    public void forMember() {
        Log.info("Member 요청입니다.");
    }
}
```



스프링부트 시큐리티

▪ SecurityConfig 재정의

```
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter{

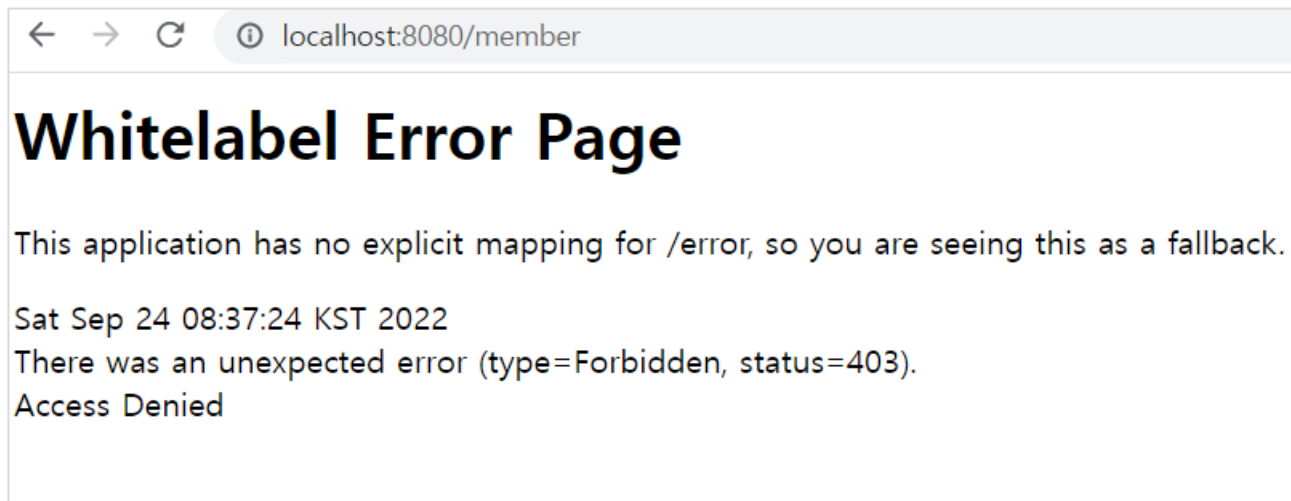
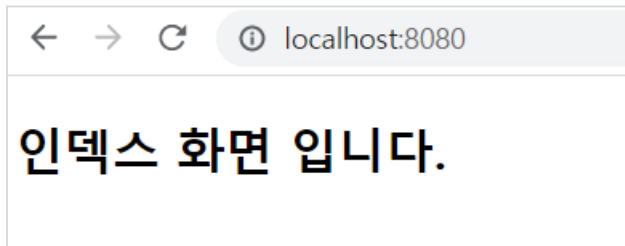
    @Override
    protected void configure(HttpSecurity security) throws Exception {
        /*security.authorizeRequests().antMatchers("/").permitAll();
        security.authorizeRequests().antMatchers("/member/**").authenticated();*/

        security.authorizeRequests()
            .antMatchers("/").permitAll()
            .antMatchers("/member/**").authenticated();

        security.csrf().disable();
    }
}
```



- SecurityConfig 재정의



스프링부트 시큐리티

- SecurityConfig 재정의

```
security.csrf().disable();    //csrf 비활성화
//security.formLogin();      //스프링부트 제공 로그인 폼 실행
security.formLogin().loginPage("/login").defaultSuccessUrl("/loginSuccess", true);
```



스프링부트 시큐리티

- 사용자 인증하기

localhost:8181/

게시판 프로그램입니다.



[글 목록](#) [로그인](#)



스프링부트 시큐리티

- 사용자 인증하기

localhost:8181/member

로그인

아이디	<input type="text" value="user"/>
비밀번호	<input type="password" value="....."/>
<input type="button" value="로그인"/>	

로그인 인증을 성공했습니다.

[INDEX 페이지로 이동](#)

[MEMBER 페이지로 이동](#)



스프링부트 시큐리티

- 메모리 사용자 인증하기

```
@Autowired
public void authenticate(AuthenticationManagerBuilder auth) throws Exception {
    //메모리 사용자 인증하기
    auth.inMemoryAuthentication()
        .withUser("manager")
        .password("{noop}manager123")
        .roles("MANAGER");

    auth.inMemoryAuthentication()
        .withUser("admin")
        .password("{noop}admin123")
        .roles("ADMIN");
}
```



스프링부트 시큐리티

■ 메모리 사용자 인증하기

localhost:8181/member로 요청
manager/manager123 -> 로그인

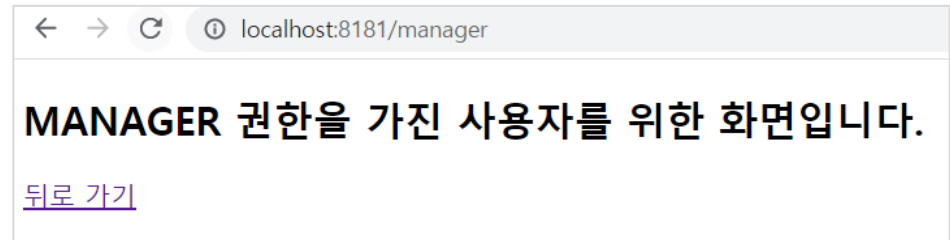
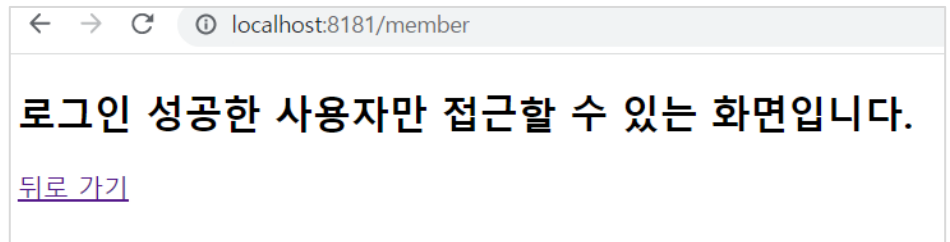
로그인 인증을 성공했습니다.

[INDEX 페이지로 이동](#)

[MEMBER 페이지로 이동](#)

[MANAGER 페이지로 이동](#)

[ADMIN 페이지로 이동](#)



스프링부트 시큐리티

- 접근 권한 없음 페이지 처리

SecurityConfig.java

```
//접근 권한 없음 페이지 처리  
security.exceptionHandling().accessDeniedPage("/accessDenied");
```

SecurityController.java

```
//접근 권한 없음 페이지  
@GetMapping("/accessDenied")  
public void accessDenied() {  
    Log.info("accessDenied 접근 거부");  
}
```



스프링부트 시큐리티

- 접근 권한 없음 페이지 처리

accessDenied.html

```
<link rel="stylesheet" href="/static/css/style.css">
</head>
<body>
  <div id="container">
    <h3>페이지에 대한 접근 권한이 없습니다.</h3>
    <h4>다시 로그인을 원하시면 <a th:href="@{/login}">여기(클릭)</a></h4>
  </div>
</body>
```

localhost:8181/admin

페이지에 대한 접근 권한이 없습니다.

다시 로그인을 원하시면 [여기\(클릭\)](#)



스프링부트 시큐리티

- 로그아웃 처리하기

SecurityConfig.java

```
//로그아웃 후 로그인 페이지로 이동  
security.logout().invalidateHttpSession(true).logoutSuccessUrl("/login");
```

manager.html

```
<h2>MANAGER 권한을 가진 사용자를 위한 화면입니다.</h2>  
<a th:href="@{/loginSuccess}">뒤로 가기</a><br>  
  
<form action="logout" method="get">  
    <input type="submit" value="로그아웃">  
</form>
```

MANAGER 권한을 가진 사용자를 위한 화면입니다.

[뒤로 가기](#)

로그아웃



스프링부트 시큐리티

- 데이터베이스 연동하기

Member 테이블 생성

기본 옵션 인덱스 (1) 외래 키 (0) 제약 조건 확인 (0) 분할 CREATE 코드

이름: member

코멘트:

필드: + 추가 × 제거 ▲ 위로 ▼ 아래로

#	이름	데이터 유형	길이/설정	부호 ...	NULL 허용	0으로 채움	기본값
1	id	VARCHAR	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음
2	password	VARCHAR	100	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음
3	name	VARCHAR	30	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음
4	role	VARCHAR	12	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음
5	enabled	TINYINT	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

<

도움말 되돌리기 저장



스프링부트 시큐리티

- 데이터베이스 연동하기

Member 테이블 생성

```
-- bootboard의 member 테이블  
-- 데이터 입력  
INSERT INTO member VALUES ('member', 'member123', '회원', 'ROLE_MEMBER', TRUE);  
INSERT INTO member VALUES ('manager', 'manager123', '매니저', 'ROLE_MANAGER', true);  
INSERT INTO member VALUES ('admin', 'admin123', '어드민', 'ROLE_ADMIN', true);
```

member (3r x 5c)					
id	password	name	role	enabled	
admin	admin123	어드민	ROLE_ADMIN	1	
manager	manager123	매니저	ROLE_MANAGER	1	
member	member123	회원	ROLE_MEMBER	1	



스프링부트 시큐리티

- 시큐리티 설정 수정

```
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter{

    @Autowired
    private DataSource dataSource;
```

```
@Autowired
public void authenticate(AuthenticationManagerBuilder auth) throws Exc
    //데이터베이스에 저장된 사용자로 인증 처리
    //사용자가 입력한 아이디로 사용자 정보 조회
    //{noop} - 비밀번호에 암호화를 적용하지 않음
    String query1 = "select id username, concat('{noop}', password) "
        + "password, true enabled from member where id=?";
    //사용자가 입력한 아이디로 권한 정보 조회
    String query2 = "select id, role from member where id=?";

    auth.jdbcAuthentication()
        .dataSource(dataSource)
        .usersByUsernameQuery(query1)
        .authoritiesByUsernameQuery(query2);
```



스프링부트 시큐리티

■ 시큐리티 테스트

localhost:8181/manager로 요청
manager/manager123 -> 로그인

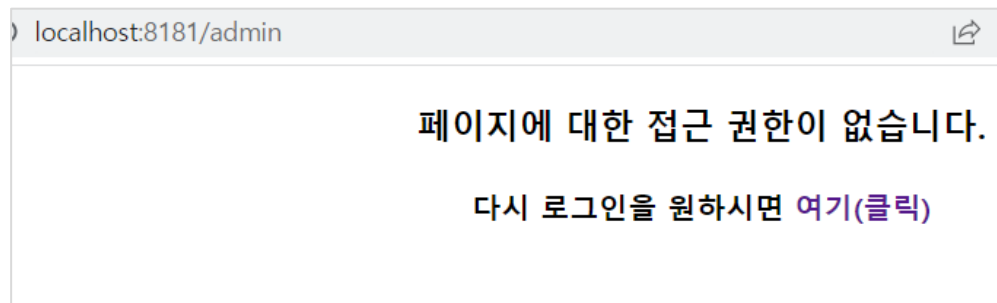
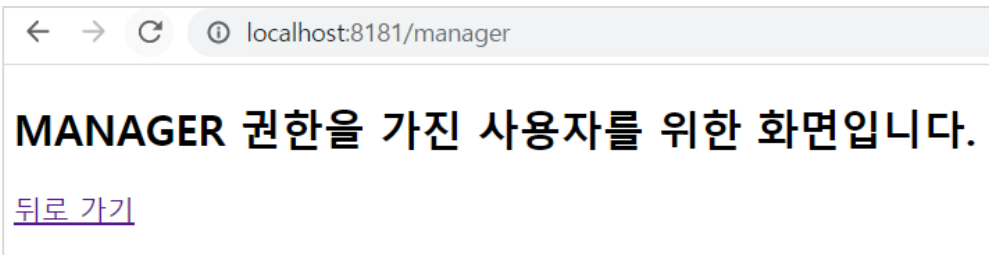
로그인 인증을 성공했습니다.

[INDEX 페이지로 이동](#)

[MEMBER 페이지로 이동](#)

[MANAGER 페이지로 이동](#)

[ADMIN 페이지로 이동](#)



스프링부트 시큐리티

- JPA 연동하기
 - 엔티티 클래스

```
public enum Role {  
    ROLE_ADMIN, ROLE_MANAGER, ROLE_MEMBER  
}
```

```
@Data  
@Entity  
public class Member {  
    @Id  
    private String id;  
    private String password;  
    private String name;  
  
    @Enumerated(EnumType.STRING)  
    private Role role;  
  
    private boolean enabled;  
}
```



스프링부트 시큐리티

- JPA 연동하기
 - 리포지토리

```
package com.boot.persistence;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface MemberRepository extends JpaRepository<Member, String>{  
}
```



스프링부트 시큐리티

▪ 사용자 정의 UserDetailsService 구현하기

```
@Service
public class BoardUserDetailsService implements UserDetailsService{

    @Autowired
    private MemberRepository memberRepo;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        //MemberRepository로 회원 정보를 조회하여
        //UserDetails 타입의 객체로 리턴한다.
        Optional<Member> optional = memberRepo.findById(username);
        if(!optional.isPresent()) {
            throw new UsernameNotFoundException(username + "사용자 없음");
        }else {
            Member member = optional.get();
            return new SecurityUser(member);
        }
    }
}
```



스프링부트 시큐리티

- 사용자 정의 UserDetailsService 구현하기

```
public class SecurityUser extends User{

    private static final long serialVersionUID = 1L;

    public SecurityUser(Member member) {
        super(member.getId(), "{noop}" + member.getPassword(),
            AuthorityUtils.createAuthorityList(member.getRole().toString()));
    }
}
```



스프링부트 시큐리티

- 사용자 정의 UserDetailsService 적용하기

```
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter{

    @Autowired
    private BoardUserDetailsService boardUserDetailsService;

    @Override
    protected void configure(HttpSecurity security) throws Exception {
```

```
        //스프링 시큐리티가 제공하는 UserDetailsService가 아닌 사용자 정의 UserDetailsService 사용
        security.userDetailsService(boardUserDetailsService);
    }
```



스프링부트 시큐리티

- PasswordEncoder 사용하기
 - 비밀번호 암호화

```
package com.boot.config;

import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.User;

import com.boot.domain.Member;

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()));
    }
}
```



스프링부트 시큐리티

- PasswordEncoder 사용하기
 - 비밀번호 암호화 적용하기

SecurityConfig.java

```
@Bean
public PasswordEncoder passwordEncoder() {
    return PasswordEncoderFactories.createDelegatingPasswordEncoder();
}
```



▪ PasswordEncoder 테스트

```
@SpringBootTest
public class PasswordEncoderTest {

    @Autowired
    private MemberRepository memberRepo;

    @Autowired
    private PasswordEncoder encoder;

    @Test
    public void testInsert() {
        Member member = new Member();
        member.setId("manager2");
        member.setPassword(encoder.encode("manager222"));
        member.setName("매니저2");
        member.setRole(Role.ROLE_MANAGER);
        member.setEnabled(true);

        memberRepo.save(member);
    }
}
```



스프링부트 시큐리티

▪ PasswordEncoder 테스트

member (1r x 5c)					
id	enabled	name	password	role	
manager2	1	매니저2	{bcrypt}\$2a\$10\$nGvZLk1Aq9oa/lpNm1vBau9Ha.g...	ROLE_MANAGER	

{noop} 제거

```
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()));
    }
}
```



스프링부트 시큐리티

▪ PasswordEncoder 테스트

localhost:8181/manager로 요청
manager2/manager222 -> 로그인

로그인 인증을 성공했습니다.

[INDEX 페이지로 이동](#)

[MEMBER 페이지로 이동](#)

[MANAGER 페이지로 이동](#)

[ADMIN 페이지로 이동](#)

