

# 10장. 스프링 Security – 로그인, 회원가입



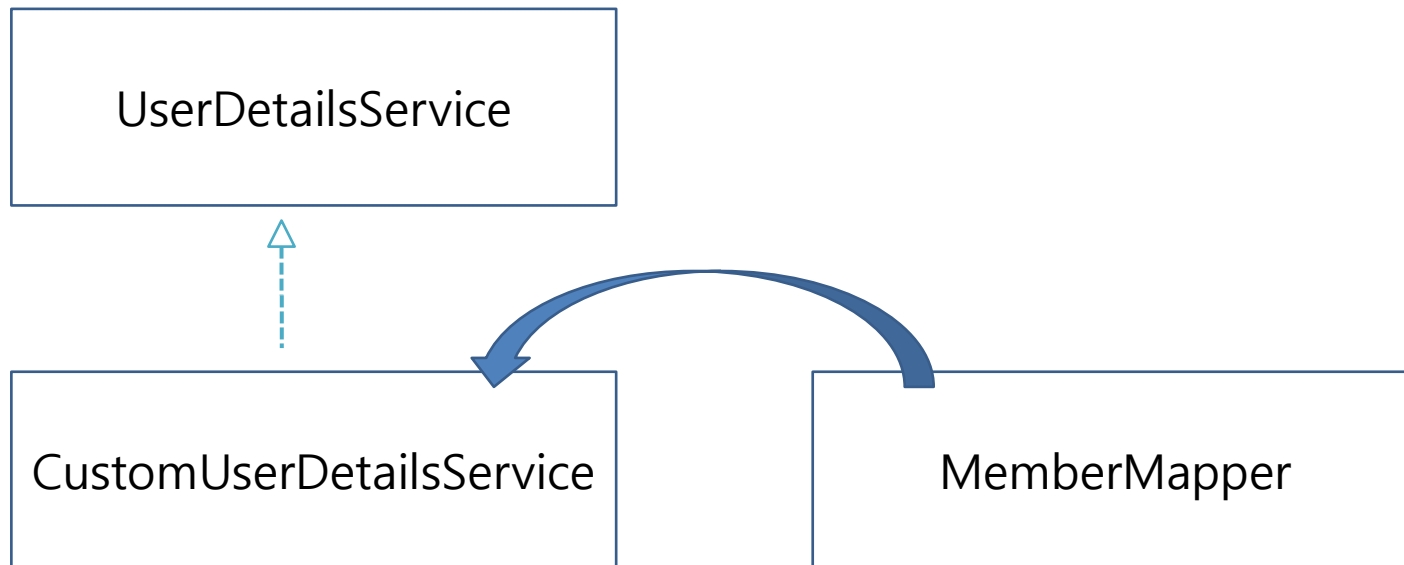
*Security - login*



# Spring Web Security

## ◆ 커스텀 UserDetailsService 활용

스프링 시큐리티에서는 username이라 부르는 사용자의 정보만을 이용하기 때문에 실제 프로젝트에서 사용자의 이름이나 이메일 등의 자세한 정보를 이용하기에 부족함이 있다 CustomUserDetailsService는 스프링 시큐리티의 UserDetails를 구현하고, MemberMapper 타입의 인스턴스를 주입 받아서 실제 기능을 구현한다.



# Spring Web Security

## ◆ 회원 도메인 - 인증 클래스 참조

```
@Data
public class MemberVO implements Serializable{

    private static final long serialVersionUID = 15L;

    private String userid;
    private String userpw;
    private String userName;
    private String enabled;

    private Date regDate;
    private Date updateDate;
    private List<AuthVO> authList;
}
```

▼ com.cloud.domain

- > AuthVO.java
- > BoardVO.java
- > MemberVO.java
- > UserVO.java



# Spring Web Security

## ◆ 회원 도메인, 회원 Mapper 설계

```
@Data
public class AuthVO implements Serializable{

    private static final long serialVersionUID = 20L;

    private String userid;
    private String auth;
}
```



# Spring Web Security

## ◆ 회원 Mapper 설계

```
package com.cloud.mapper;  
  
import com.cloud.domain.MemberVO;  
  
public interface MemberMapper {  
  
    //회원 조회(검색)  
    public MemberVO read(String userid);  
}
```

MemberMapper.java



# Spring Web Security

MemberMapper.xml

```
<mapper namespace="com.cloud.mapper.MemberMapper">
  <resultMap type="com.cloud.domain.MemberVO" id="memberMap">
    <id property="userid" column="userid"/>
    <result property="userid" column="userid"/>
    <result property="userpw" column="userpw"/>
    <result property="userName" column="username"/>
    <result property="regDate" column="regdate"/>
    <result property="updateDate" column="updatedate"/>
    <collection property="authList" resultMap="authMap" />
  </resultMap>

  <resultMap type="com.cloud.domain.AuthVO" id="authMap">
    <result property="userid" column="userid"/>
    <result property="auth" column="auth"/>
  </resultMap>
```



# Spring Web Security

```
<select id="read" resultMap="memberMap">
    SELECT mem.userid, userpw, username, enabled, regdate, updatedate, auth
    FROM tbl_member mem
        LEFT OUTER JOIN tbl_member_auth auth
        on mem.userid = auth.userid
    WHERE mem.userid = #{userid}
</select>
```

MemberMapper.xml

```
</mapper>
```

id가 "read"인 <select> 태그는 resultMap 속성을 지정한다.

memberMap이라는 이름을 가진 <resultMap>은 <result>와 <collection>을 이용해서 바깥쪽 객체( MemberVO의 인스턴스)와 안쪽 객체(AuthVO의 인스턴스)를 구성할 수 있다.

MyBatis에서는 하나의 결과에 여러 개의 데이터를 처리하는 1:N,의 결과를 처리할 수 있는 <resultMap> 태그를 지원한다.



# Spring Web Security

- LEFR OUTER JOIN 테이블 이름 ON 조인 조건

```
-- INNER JOIN : INNER 생략 가능
-- LEFT OUTER JOIN ~ ON ~ : OUTER 생략 가능
-- 회원의 상세 정보 출력
-- 인증 권한이 없는 회원도 출력
SELECT mem.userid, userpw, username, enabled, regdate, updatedate, auth
FROM tbl_member mem
     LEFT OUTER JOIN tbl_member_auth auth
     on mem.userid = auth.userid
WHERE mem.userid = 'member82';
```

USERID	USERPW	USERNAME	ENABLED	REGDATE	UPDATEDATE	AUTH
member82	\$2a\$10\$00tBYjZiHak/K0Y9ZwkAx.DL9g...	회원82	1	22/08/29	22/08/29	ROLE_MEMBER





# Spring Web Security

## ◆ MemberMapper 테스트

```
@Log4j
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration({
    "file:src/main/webapp/WEB-INF/spring/root-context.xml",
    "file:src/main/webapp/WEB-INF/spring/security-context.xml"
})
public class MemberMapperTests {

    @Autowired
    private MemberMapper mapper;

    @Test
    public void testRead() {
        MemberVO vo = mapper.read("admin93");
        Log.info(vo);

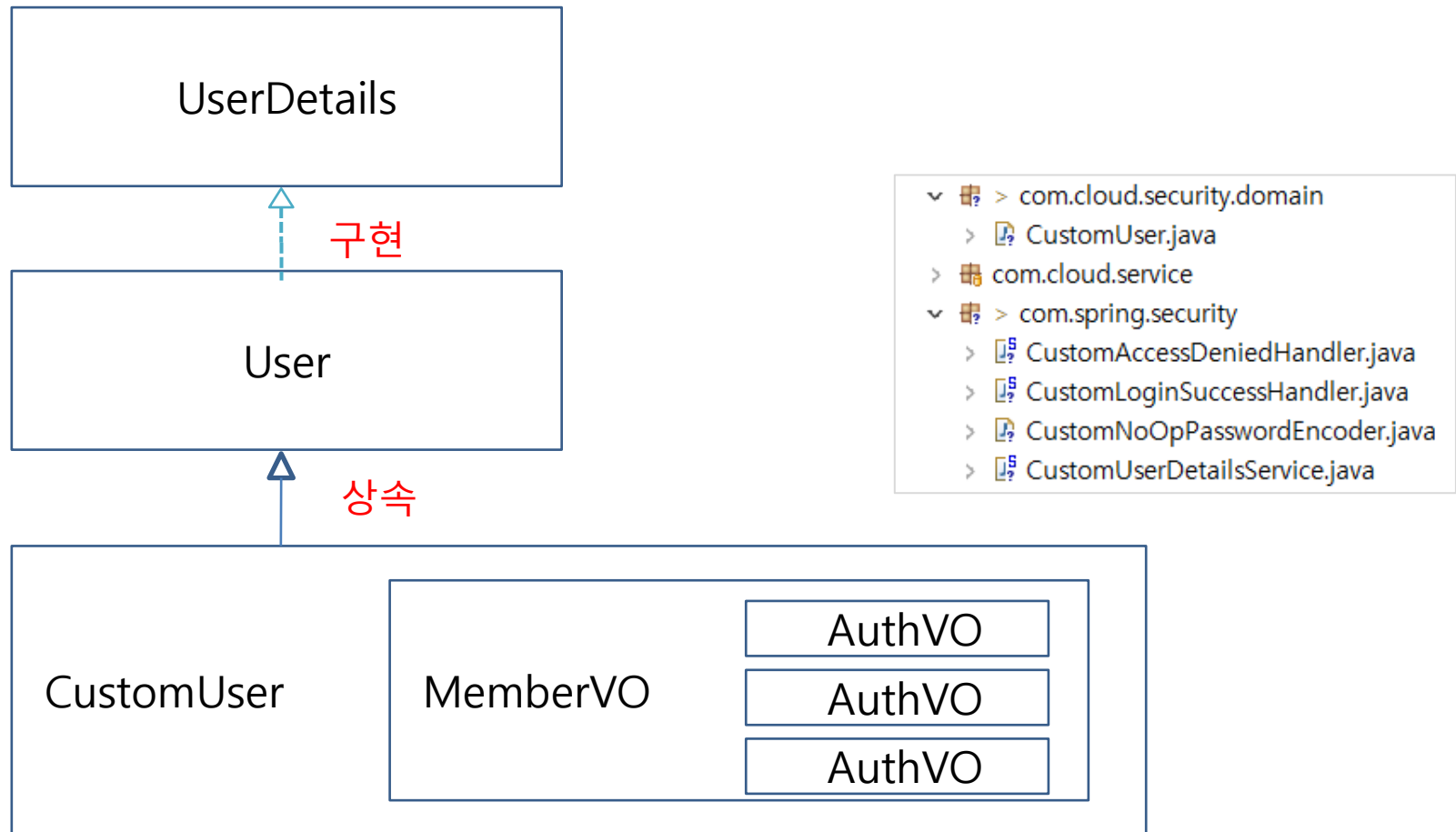
        vo.getAuthList().forEach(authVO -> Log.info(authVO));
    }
}
```

```
INFO : com.cloud.security.MemberMapperTests - MemberVO(userid=admin93, userpw=$2a$10$ir6Pg9DNfa4e7gj/E
INFO : com.cloud.security.MemberMapperTests - AuthVO(userid=admin93, auth=ROLE_ADMIN)
```



# Spring Web Security

## ◆ CustomUserDetailsService 구성



# Spring Web Security

## ◆ CustomUserService 구성

```
@Log4j
public class CustomUserService implements UserDetailsService{

    @Autowired
    private MemberMapper mapper;

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

        Log.warn("Load User By UserName : " + username);

        MemberVO vo = mapper.read(username);

        Log.warn("queried by member mapper: " + vo);

        return vo == null ? null : new CustomUser(vo); //vo가 있으면 CustomerUser
    }
}
```



# Spring Web Security

## ◆ CustomUser 클래스

```
@Getter
public class CustomUser extends User{
    private static final long serialVersionUID = 11L;

    private MemberVO member;

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

    public CustomUser(MemberVO vo) {
        super(vo.getUserid(), vo.getUserpw(),
            vo.getAuthList()
                .stream()
                .map(auth -> new SimpleGrantedAuthority(auth.getAuth()))
                .collect(Collectors.toList()));

        this.member = vo;
    }
}
```



# Spring Web Security

## ◆ CustomUserDetailsService 빈 등록 및 권한 변경

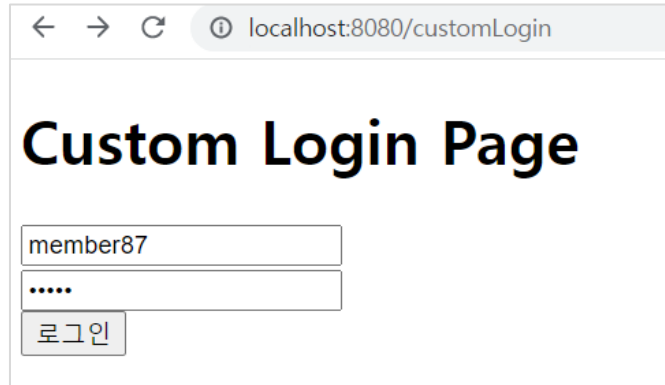
```
<bean id="customUserDetailsService"  
      class="com.spring.security.CustomUserDetailsService" />
```

```
<!-- 인증 - 권한 -->  
<security:authentication-manager>  
  <security:authentication-provider  
    user-service-ref="customUserDetailsService">  
    <!-- <security:jdbc-user-service data-source-ref="dataSource"/> -->  
  
    <!-- sql query를 이용한 인증 -->  
    <!-- <security:jdbc-user-service  
      data-source-ref="dataSource"  
      users-by-username-query="select userid, userpw, enabled from  
        tbl_member where userid=?"  
      authorities-by-username-query="select userid, auth from  
        tbl_member_auth where userid=?" /> -->  
  
    <!-- <security:password-encoder ref="customPasswordEncoder" /> -->  
    <!-- 패스워드 암호화 -->  
    <security:password-encoder ref="bcryptPasswordEncoder" />
```



# Spring Web Security

## ◆ customLogin 페이지 인증 - 최종 완성



← → ↻ ⓘ localhost:8080/customLogin

## Custom Login Page

member87

.....

로그인

```
security.CustomUserDetailsService - Load User By UserName : member81
security.CustomUserDetailsService - queried by member mapper: MemberVO(userid=member81, userpw=$2a$10$
security.CustomLoginSuccessHandler - Login Success
security.CustomLoginSuccessHandler - ROLE NAMES: [ROLE_MEMBER]
security.CustomUserDetailsService - Load User By UserName : admin93
security.CustomUserDetailsService - queried by member mapper: MemberVO(userid=admin93, userpw=$2a$10$
security.CustomLoginSuccessHandler - Login Success
security.CustomLoginSuccessHandler - ROLE NAMES: [ROLE_ADMIN]
```



# Spring Web Security

## ◆ JSP에서 로그인한 사용자 정보 보여주기 – admin.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/security/tags" prefix="security" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>인증</title>
</head>
<body>
    <h1>/sample/admin page</h1>

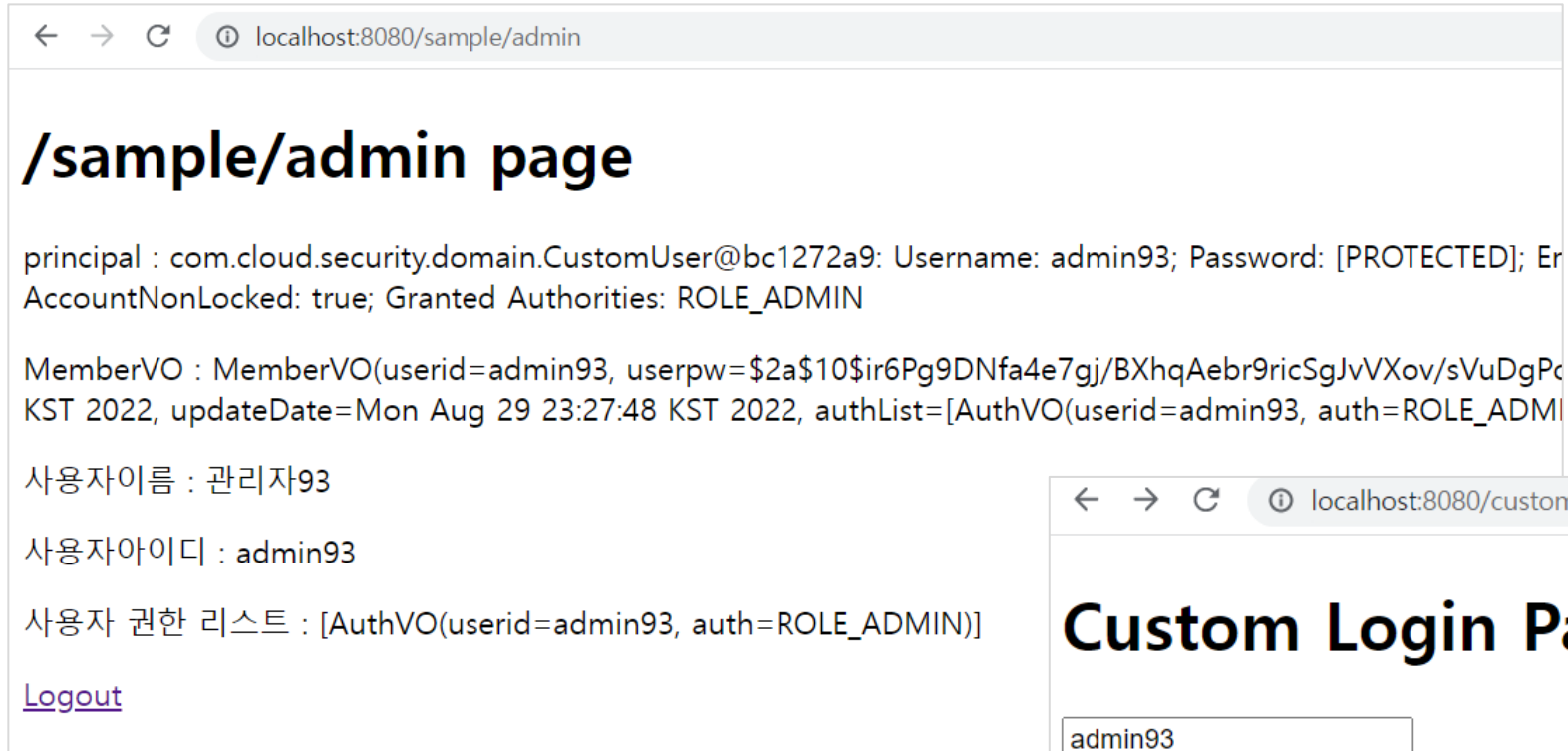
    <p>principal : <security:authentication property="principal"/></p>
    <p>MemberVO : <security:authentication property="principal.member"/></p>
    <p>사용자이름 : <security:authentication property="principal.member.userName"/>
    <p>사용자아이디 : <security:authentication property="principal.username"/></p>
    <p>사용자 권한 리스트 : <security:authentication property="principal.member.authList"/></p>

    <a href="/customLogout">Logout</a>
</body>
</html>
```



# Spring Web Security

## ◆ JSP에서 로그인한 사용자 정보 보여주기



← → ↻ localhost:8080/sample/admin

### /sample/admin page

principal : com.cloud.security.domain.CustomUser@bc1272a9: Username: admin93; Password: [PROTECTED]; Er  
AccountNonLocked: true; Granted Authorities: ROLE\_ADMIN

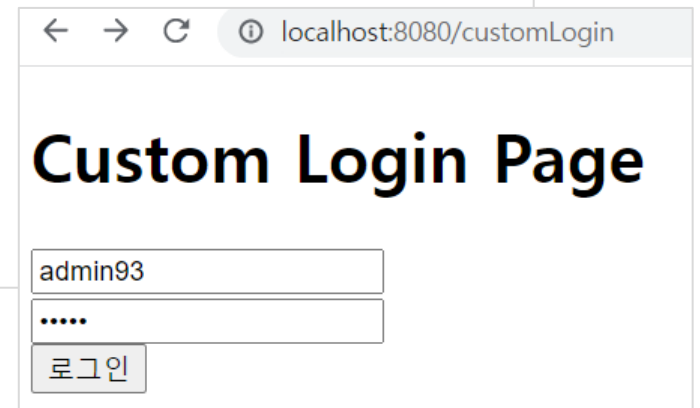
MemberVO : MemberVO(userid=admin93, userpw=\$2a\$10\$ir6Pg9DNfa4e7gj/BXhqAebr9ricSgJvVXov/sVuDgPc  
KST 2022, updateDate=Mon Aug 29 23:27:48 KST 2022, authList=[AuthVO(userid=admin93, auth=ROLE\_ADMIN)

사용자이름 : 관리자93

사용자아이디 : admin93

사용자 권한 리스트 : [AuthVO(userid=admin93, auth=ROLE\_ADMIN)]

[Logout](#)



← → ↻ localhost:8080/customLogin

## Custom Login Page

admin93

.....

로그인





# Spring Web Security

## ◆ 표현식을 이용하는 동적 화면 구성

표현식	설명
hasRole([role]) hasAuthority([authority])	해당 권한이 있으면 true
hasAnyRole([role1, role2]) hasAnyAuthority([authority])	여러 권한들 중에서 하나라도 해당하는 권한이 있으면 true
principal	현재 사용자의 정보를 의미
permitAll	모든 사용자에게 허용
denyAll	모든 사용자에게 거부
isAnonymous()	익명의 사용자의 경우(로그인을 하지 않은 경우도 해당)
isAuthenticated()	인증된 사용자만 true



# Spring Web Security

## ◆ all.jsp

```
<h1>/sample/all page</h1>
<!-- 로그인하지 않은 사용자 -->
<security:authorize access="isAnonymous()">
    <a href="/customLogin">로그인</a>
</security:authorize>

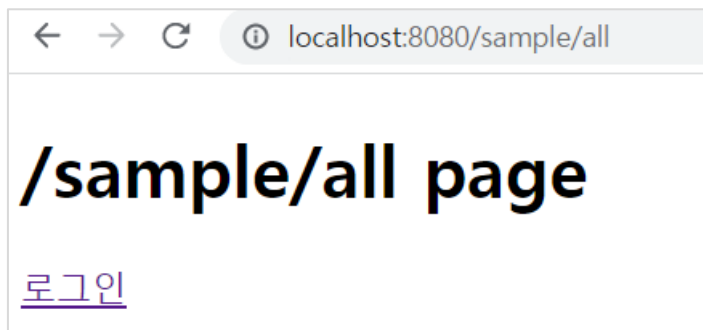
<!-- 로그인한 사용자 -->
<security:authorize access="isAuthenticated()">
    <a href="/customLogout">로그아웃</a>
</security:authorize>
```



# Spring Web Security

## ◆ 권한 접근

로그인 하지 않은 사용자



로그인한 사용자



## ◆ 어노테이션을 이용하는 스프링 시큐리티 설정

**@ Secured:** 스프링 시큐리티 초기부터 사용되었고, ()안에 'ROLE\_ADMIN'과 같은 문자열 혹은 문자열 배열을 이용한다.

**@PreAuthorize, @PostAuthorize:** 스프링 3버전 부터 지원되어 ()안에 표현식을 사용할 수 있으므로 최근에는 더 많이 사용된다.



# Spring Web Security

## ◆ 어노테이션을 이용하는 스프링 시큐리티 설정

servlet-context.xml 설정

```
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /W
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <beans:property name="prefix" value="/WEB-INF/views/" />
    <beans:property name="suffix" value=".jsp" />
</beans:bean>

<context:component-scan base-package="com.cCloud" />

<security:global-method-security
    pre-post-annotations="enabled" secured-annotations="enabled" />
```



# Spring Web Security

## ◆ SpringBoard2에 접속하기

메인 페이지입니다.



[로그인 바로가기](#)

[게시글 목록](#)

# Spring Web Security

## ◆ SpringBoard2에 접목하기

index.jsp

```
<div id="container">
  <section id="main">
    <h2>메인 페이지입니다.</h2>
    <hr>
    
    <h4>
      <a href="/customLogin">로그인 바로가기</a>&nbsp; &nbsp; &nbsp; &nbsp;
      <a href="/board/boardList">게시글 목록</a>
    </h4>
  </section>
</div>
```



# Spring Web Security

## ◆ SpringBoard2에 접속하기

### 로그인

아이디	<input type="text" value="admin93"/>
비밀번호	<input type="password" value="...."/>
<input type="button" value="로그인"/> <input type="button" value="취소"/>	





# Spring Web Security

## ◆ SpringBoard2에 접목하기

customLogin.jsp

```
<div id="container">
  <section id="login">
    <h2>로그인</h2>
    <h3 class="error"><c:out value="${error}" /></h3>
    <form action="/login" method="post">
      <table class="tbl_login">
        <tr>
          <td>아이디</td>
          <td><input type="text" name="username"></td>
        </tr>
        <tr>
          <td>비밀번호</td>
          <td><input type="password" name="password"></td>
        </tr>
        <tr>
          <td colspan="2">
            <input type="submit" value="로그인">
            <input type="reset" value="취소">
          </td>
        </tr>
      </table>
      <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
    </form>
  </section>
```



# Spring Web Security

## ◆ SpringBoard2에 접속하기

### 글 목록

admin93님 환영합니다... [Logout](#)

번호	제목	작성자	등록일	조회수
9	마켓컬리-새벽 배송	member88	2022-08-30 11:06:20	4
8	감기	member81	2022-08-30 10:50:53	2
6	인증	관리자	2022-08-30 07:31:14	3
5	test	test	2022-08-30 06:53:16	0
4	테스트 제목	user00	2022-08-30 06:52:53	0

글쓰기



# Spring Web Security

## ◆ SpringBoard2에 접속하기

boardList.jsp

```
<section id="list">
  <h2>글 목록</h2>
  <h3>
    <security:authentication property="principal.username"/>님 환영합니다...
    &nbsp;&nbsp;&nbsp;&nbsp;<a href="/customLogout" style="color:green">Logout</a>
  </h3>
  <table class="tbl_list">
    <tr>
      <th>번호</th>
      <th>제목</th>
```



# Spring Web Security

## 글쓰기

제목	<input type="text" value="title"/>
작성자	<input type="text" value="admin93"/>
내용	<div><div>content</div><div></div></div>
<div><div>등록</div><div>목록</div></div>	



# Spring Web Security

BoardController.java

```
@GetMapping("/insertBoard")
@PreAuthorize("isAuthenticated()")
public String insertBoard() { //글쓰기 폼 페이지 요청
    return "/board/insertBoard";
}

@PostMapping("/insertBoard")
@PreAuthorize("isAuthenticated()")
public String insertBoard(BoardVO vo){ //글쓰기 처리
    service.insert(vo);
    return "redirect:/board/boardList";
}
```



# Spring Web Security

insertBoard.jsp

```
<tr>
  <td>작성자</td>
  <td><input type="text" name="writer"
    value='<security:authentication property="principal.username"/>' readonly="readonly">
  </td>
</tr>
```



# Spring Web Security

## 글 상세 보기

제목	<input type="text" value="가을"/>
작성자	<input type="text" value="admin93"/>
내용	<div>가을이 성큼 오고 있네요 가을옷을 준비해야 할 듯...</div>
등록일	2022-08-31 08:07:49
조회수	<input type="text" value="1"/>
<div>글 수정 삭제 목록</div>	



# Spring Web Security

boardView.jsp

```
<tr>
  <td colspan="2" align="center">
    <security:authentication property="principal" var="pinfo"/>
    <security:authorize access="isAuthenticated()" >
      <c:if test="${pinfo.username eq board.writer}">
        <input type="submit" value="글 수정">
        <a href="/board/deleteBoard?bno=${board.bno}"
          onclick="return confirm('정말로 삭제하시겠습니까?')">
          <input type="button" value="삭제"></a>
      </c:if>
    </security:authorize>
    <a href="/board/boardList"><input type="button" value="목록"></a>
  </td>
</tr>
```





# Spring Web Security

## 회원 가입

아이디	<input type="text" value="ID"/>
비밀번호	<input type="password" value="PASSWORD"/>
비밀번호 확인	<input type="password" value="PASSWORD_CONFIRM"/>
이름	<input type="text"/>
<input type="button" value="등록"/> <input type="button" value="취소"/>	



# Spring Web Security

```
SpringBoard2 [springwork master]
├── src/main/java
│   ├── com.cloud.controller
│   │   ├── BoardController.java
│   │   ├── CommonController.java
│   │   ├── HomeController.java
│   │   ├── MemberController.java
│   │   └── SampleController.java
│   ├── com.cloud.domain
│   ├── com.cloud.mapper
│   │   ├── BoardMapper.java
│   │   ├── MemberAuthMapper.java
│   │   └── MemberMapper.java
│   ├── com.cloud.security.domain
│   ├── com.cloud.service
│   │   ├── BoardService.java
│   │   ├── BoardServiceImpl.java
│   │   ├── CommonService.java
│   │   ├── CommonServiceImpl.java
│   │   ├── MemberService.java
│   │   └── MemberServiceImpl.java
```

```
WEB-INF
├── classes
├── spring
└── views
    ├── board
    │   ├── boardList.jsp
    │   ├── boardView.jsp
    │   └── insertBoard.jsp
    ├── member
    │   └── signup.jsp
    ├── sample
    │   ├── accessError.jsp
    │   ├── customLogin.jsp
    │   ├── customLogout.jsp
    │   ├── home.jsp
    │   ├── index.jsp
    │   └── login.jsp
    └── web.xml
```



# Spring Web Security

Username 수정

```
@Data
public class MemberVO implements Serializable{

    private static final long serialVersionUID = 15L;

    private String userid;
    private String userpw;
    private String username;
    private String enabled;

    private Date regDate;
    private Date updateDate;
    private List<AuthVO> authList;
}
```



# Spring Web Security

## ◆ 회원 가입

index.jsp

```
<div id="container">
  <section id="main">
    <h2>메인 페이지입니다.</h2>
    <hr>
    
    <h4>
      <a href="/customLogin">로그인</a>&nbsp; &nbsp; &nbsp; &nbsp;
      <a href="/member/signup">회원가입</a>&nbsp; &nbsp; &nbsp; &nbsp;
      <a href="/board/boardList">게시글 목록</a>
    </h4>
  </section>
</div>
```



# Spring Web Security

## ◆ 회원 가입

```
<form action="/member/signup" method="post" name="regForm">
<table class="tbl_reg">
  <tr>
    <td>아이디</td>
    <td><input type="text" name="userid" placeholder="ID"></td>
  </tr>
  <tr>
    <td>비밀번호</td>
    <td><input type="password" name="userpw" placeholder="PASSWORD"></td>
  </tr>
  <tr>
    <td>비밀번호 확인</td>
    <td><input type="password" name="userpw" placeholder="PASSWORD_CONFIRM"></td>
  </tr>
  <tr>
    <td>이름</td>
    <td><input type="text" name="username"></td>
  </tr>
  <tr>
    <td colspan="2">
      <input type="submit" value="등록">
      <input type="reset" value="취소">
    </td>
  </tr>
</table>
<input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
```

signup.jsp



# Spring Web Security

## ◆ 회원 가입

```
public interface MemberMapper {  
  
    public MemberVO read(String userid); //회원 조회(검색)  
  
    public void insertMember(MemberVO member); //회원 가입  
  
}
```

MemberMapper.xml

```
<insert id="insertMember">  
    INSERT INTO tbl_member(userid, userpw, username, enabled)  
    VALUES ({userid}, {userpw}, {username}, '1')  
</insert>
```



# Spring Web Security

## ◆ 회원 가입

```
package com.cloud.mapper;  
  
import com.cloud.domain.AuthVO;  
  
public interface MemberAuthMapper {  
  
    //회원 가입(권한 부여)  
    public void insertMemberAuth(AuthVO auth);  
}
```

MemberAuthMapper.xml

```
<mapper namespace="com.cloud.mapper.MemberAuthMapper">  
    <insert id="insertMemberAuth">  
        INSERT INTO tbl_member_auth(userid, auth)  
        VALUES (#{userid}, #{auth})  
    </insert>  
</mapper>
```



# Spring Web Security

## ◆ 회원 가입

MemberService.java

```
package com.cloud.service;

import com.cloud.domain.MemberVO;

public interface MemberService {

    //회원 가입
    public void signup(MemberVO member);
}
```





# Spring Web Security

```
@Service
public class MemberServiceImpl implements MemberService{

    @Autowired
    private MemberMapper mapper;

    @Autowired
    private MemberAuthMapper authMapper;

    @Autowired
    private PasswordEncoder pwencoder;

    @Override
    public void signup(MemberVO member) {

        String encPw = pwencoder.encode(member.getUserpw());
        member.setUserpw(encPw);

        mapper.insertMember(member);

        //기본 ROLE_USER 부여
        AuthVO auth = new AuthVO();
        auth.setUserid(member.getUserid());
        auth.setAuth("ROLE_USER");
        authMapper.insertMemberAuth(auth);
    }
}
```



# Spring Web Security

```
@AllArgsConstructor
@Log4j
@RequestMapping("/member/*")
@Controller
public class MemberController {

    private MemberService service; //생성자 주입

    //회원 가입 폼 요청
    @GetMapping("/signup")
    public void signUp() {
        Log.info("회원 가입 폼");
    }

    @PostMapping("/signup")
    public String signUp(MemberVO member, RedirectAttributes rttr) {
        service.signUp(member);
        //rttr.addFlashAttribute("result", member.getUserid());
        return "redirect:/board/boardList";
    }
}
```

