**로그인 시큐리티**

Pom.xml추가

<dependency>

    <groupId>org.springframework.security</groupId>

    <artifactId>spring-security-web</artifactId>

    <version>4.2.1.RELEASE</version>

</dependency>

<dependency>

    <groupId>org.springframework.security</groupId>

    <artifactId>spring-security-core</artifactId>

    <version>4.2.1.RELEASE</version>

</dependency>

<dependency>

    <groupId>org.springframework.security</groupId>

    <artifactId>spring-security-config</artifactId>

    <version>4.2.1.RELEASE</version>

</dependency>

Web.xml(해당 xml파일을 읽을 수 있게 설정한다.)

<**context-param**>

<**param-name**>contextConfigLocation</**param-name**>

<**param-value**>

/WEB-INF/spring/root-context.xml

/WEB-INF/spring/spring-security.xml

</**param-value**>

</**context-param**>

**spring-security.xml생성 (**WEB-INF/spring/spring-security.xml)

root-context.xml과 같은 위치.

<?xml version="1.0" encoding="UTF-8"?>

<beans:beans xmlns="<http://www.springframework.org/schema/security>"

    xmlns:beans="<http://www.springframework.org/schema/beans>"

    xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

    xsi:schemaLocation="http://www.springframework.org/schema/beans

    http://www.springframework.org/schema/beans/spring-beans.xsd

    http://www.springframework.org/schema/security

    http://www.springframework.org/schema/security/spring-security.xsd">

    <beans:bean id="bcryptPasswordEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder" />

</beans:beans>

Service 단에 암호화 해서 DB로 넘겨준다.( security.xml의 값을 autowired한다.)

*@Autowired*

BCryptPasswordEncoder passwordEncoder;

*@Override*

public void loginCreate(UserVO vo) throws Exception {

//암호화

String encryptPassword = passwordEncoder.encode(vo.getUserpw());

System.***out***.println(encryptPassword);

vo.setUserpw(encryptPassword);

dao.loginCreate(vo);

}

암호화 비교(Controller)

(로그인 mapper 에서 id만의로 데이터를 찾도록 구문을 바꾼다.)

*@Autowired*

BCryptPasswordEncoder passwordEncoder;

// 로그인 처리하는 부분

*@RequestMapping*(value = "/loginProcess", method = *RequestMethod*.***POST***)

public String loginProcess(HttpSession session, UserVO dto) {

System.***out***.println("test");

String returnURL = "";

if (session.getAttribute("login") != null) {

// 기존에 login이란 세션 값이 존재한다면

session.removeAttribute("login"); // 기존값을 제거해 준다.

}

//입력한 페스워드

String inpw = dto.getUserpw();

System.***out***.println("dto : " + inpw);

//입력한 페스워드 암호화

String encryptPassword = passwordEncoder.encode(inpw);

//암호화된 것을 저장.

dto.setUserpw(encryptPassword);

//로그인 정보를 가져옴.

UserVO vo = service.login(dto);

System.***out***.println("vo : " + vo);

// 등어온 pw 와 DB의 가 같으면

if (passwordEncoder.matches(inpw, vo.getUserpw())) {

System.***out***.println("계정정보 일치");

// 권한 별 설정.

//0은 권한이 없는 사용자.

if (vo.getAuth() == 0) {

returnURL = "redirect:/user/loginConfirm?check=0";

//5는 admin

} else if (vo.getAuth() == 5) {

session.setAttribute("login", vo);

returnURL = "redirect:/user/loginConfirm?check=5";

} else {

// 세션에 로그인한 객체 저장

session.setAttribute("login", vo);

returnURL = "redirect:/user/loginConfirm?check=1";

}

} else {

// 로그인에 실패한 경우

System.***out***.println("계정정보 불일치");

returnURL = "redirect:/user/loginConfirm?check=99";

// returnURL ="redirect:/user/login"; // 로그인 폼으로 다시 가도록 함

}

return returnURL; // 위에서 설정한 returnURL 을 반환해서 이동시킴

}