8장. 스프링 Web Security

spring

Spring Web Security

◆ **스프링 시큐리티**

일반적인 웹(JSP)에서는 세션이나 쿠키등을 이용하여 로그인을 하는데, 스프링은 인터셉터(Interceptor)등을 이용한다.

기본 동작 방식은 서블릿의 여러 종류의 필터와 인터셉터를 이용해서 처리된다.
필터는 서블릿의 필터를 의미하고, 인터셉터(Interceptor)는 스프링에서 필터와 유사한 역할을 한다.
인터셉터는 스프링의 빈으로 관리되면서 스프링의 컨텍스트 내에 속한다.

Spring Web Security

◆ 스프링 AOP pom.xml 에 security 관련 라이브러리 추가

```xml
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>5.2.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>5.2.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.2.2.RELEASE</version>
</dependency>
```
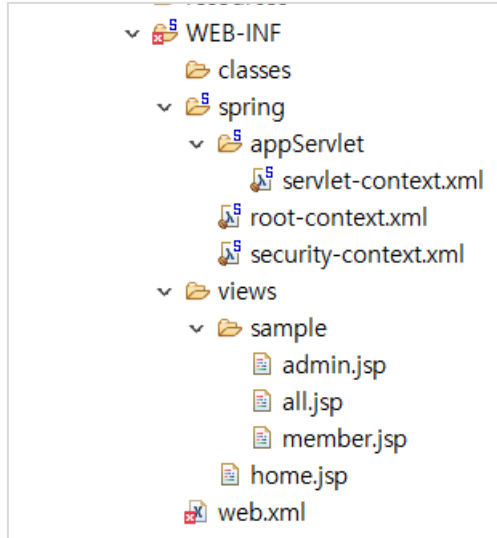
Spring Web Security

◆ **security-context.mxl 생성**

Configuration Bean File로 생성

```
v 🗁 WEB-INF
    🗁 classes
    v 🗁 spring
        v 🗁 appServlet
            🗋 servlet-context.xml
        🗋 root-context.xml
        🗋 security-context.xml
    v 🗁 views
        v 🗁 sample
            📄 admin.jsp
            📄 all.jsp
            📄 member.jsp
        📄 home.jsp
    🗋 web.xml
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:security="http://www.springframework.org/schema/security"
    xsi:schemaLocation="http://www.springframework.org/schema/security http://www.sprin
        http://www.springframework.org/schema/beans http://www.springframework.org/sche



</beans>
```

Spring Web Security

◆ **web.xml에 등록**

```xml
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
    <param-value>/WEB-INF/spring/security-context.xml</param-value>
</context-param>
```

```xml
<!-- Spring Security -->
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</
</filter>

<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Spring Web Security

◆ **security-context.xml 에 등록 및 서버 실행**

```xml
<!-- 기본 서비스 -->
<security:http>

    <security:form-login/>
    <security:logout/>
</security:http>


<!-- 인증 및 권한 설정 -->
<security:authentication-manager>

</security:authentication-manager>
```

```
INFO: Initializing Spring DispatcherServlet 'appServlet'
INFO : org.springframework.web.servlet.DispatcherServlet - Initializing
INFO : org.springframework.web.servlet.DispatcherServlet - Completed ini
8월 27, 2022 8:08:01 오전 org.apache.coyote.AbstractProtocol start
INFO: 프로토콜 핸들러 ["http-nio-8080"]을(를) 시작합니다.
8월 27, 2022 8:08:01 오전 org.apache.catalina.startup.Catalina start
INFO: Server startup in 4023 ms
INFO : com.spring.controller.HomeController - Welcome home! The client l
```

Spring Web Security

◆ **시큐리티 권한에 따른 URI 설계**

- /sample/all   -> 로그인을 하지 않은 사용자도 접근 가능

- /sample/member   -> 로그인한 사용자만 접근 가능

- /sample/admin -> 로그인한 사용자들 중에서 관리자 권한을 가

  진 사용자만 접근 가능

Spring Web Security

◆ 시큐리티 권한에 따른 URI 설계

```java
@Log4j
@RequestMapping("/sample/*")
@Controller
public class SampleController {

    @GetMapping("/all")  //localhost:8080/sample/all
    public void doAll() {
        Log.info("do all can access everybody");
    }


    @GetMapping("/member")
    public void doMember() {
        Log.info("logined member");
    }


    @GetMapping("/admin")
    public void doAdmin() {
        Log.info("admin only");
    }
}
```

```
v 🌱 src/main/java
  > 🔲 com.spring.board
  > 🔳 com.spring.board.impl
  > 🔲 com.spring.common
  v 🔲 com.spring.controller
    > 🗒 HomeController.java
    > 🗒 SampleController.java
  > 🔲 com.spring.user
```

Spring Web Security

◆ 시큐리티 권한에 따른 URI 설계

```html
<title>인증</title>
</head>
<body>
    <h1>/sample/all page</h1>
</body>
</html>
```

```
∨ 📂 views
    ∨ 📂 sample
        📄 admin.jsp
        📄 all.jsp
        📄 member.jsp
    📄 home.jsp
```
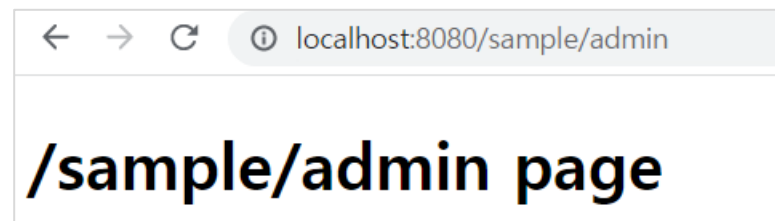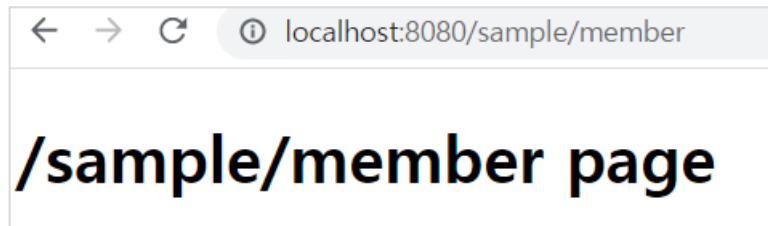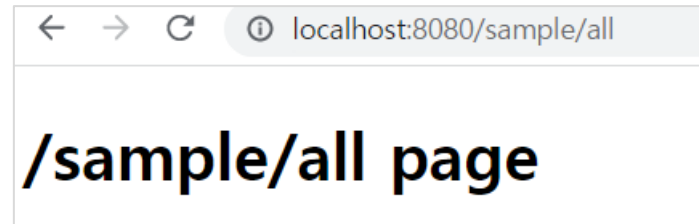
```html
<title>인증</title>
</head>
<body>
    <h1>/sample/member page</h1>
</body>
</html>
```

```html
<body>
    <h1>/sample/admin page</h1>

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

Spring Web Security

◆ 시큐리티 권한에 따른 URI 설계

Spring Web Security

◆ **인증과 권한 부여**

인증이란?.

권한이란?

Spring Web Security

◆ 로그인과 로그아웃

권한 설정 - security-context.mxl

```xml
<!-- 기본 서비스 -->
<security:http>
    <security:intercept-url pattern="/sample/all" access="permitAll" />
    <security:intercept-url pattern="/sample/member" access="hasRole('ROLE_MEMBER')" />
    <security:intercept-url pattern="/sample/admin" access="hasRole('ROLE_ADMIN')" />

    <security:form-login/>
    <security:logout/>
</security:http>

<!-- 인증 및 권한 설정 -->
<security:authentication-manager>
    <security:authentication-provider>
        <security:user-service>
            <security:user name="member" password="{noop}member"
                           authorities="ROLE_MEMBER"/>
            <security:user name="admin" password="{noop}admin"
                           authorities="ROLE_MEMBER, ROLE_ADMIN"/>
        </security:user-service>
    </security:authentication-provider>
</security:authentication-manager>
```
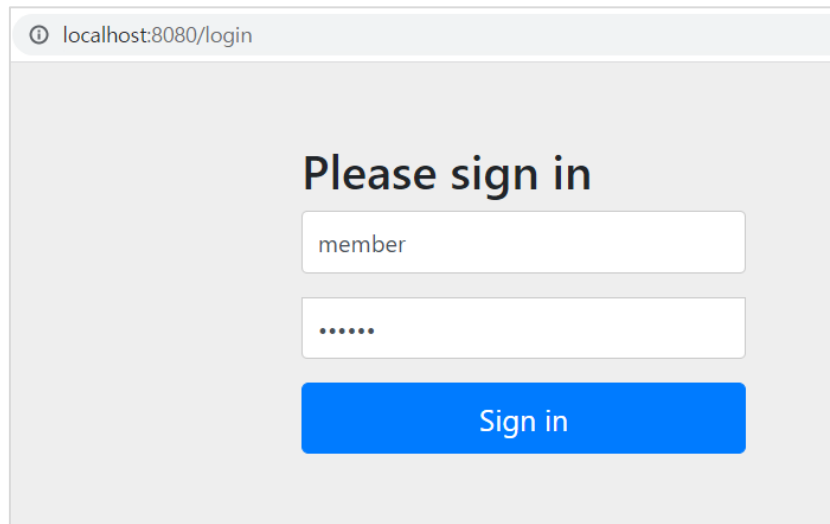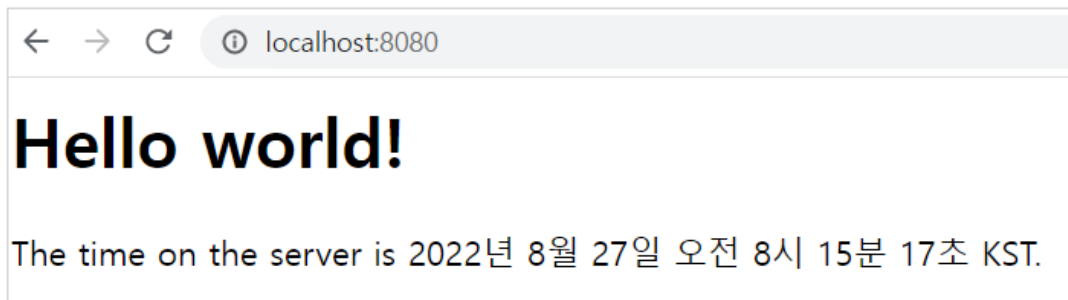
Spring Web Security

◆ **로그인과 로그아웃**

- **/sample/member로 요청(username- member, password-member)**

PasswordEncoder 클래스가 없어서 에러발생

```
SEVERE: 경로가 []인 컨텍스트의 서블릿 [appServlet]을(를) 위한 Servlet.service() 호출이 예외를 발생
java.lang.IllegalArgumentException: There is no PasswordEncoder mapped for the id "nu
        at org.springframework.security.crypto.password.DelegatingPasswordEncoder$Unm
        at org.springframework.security.crypto.password.DelegatingPasswordEncoder.mat
```

Spring Web Security

◆ 로그인과 로그아웃

  - 비밀번호앞에 {noop}를 붙이면 에러 해결됨

```xml
<security:authentication-provider>
    <security:user-service>
        <security:user name="member" password="{noop}member" authorities="ROLE_MEMBER"
    </security:user-service>
</security:authentication-provider>
```
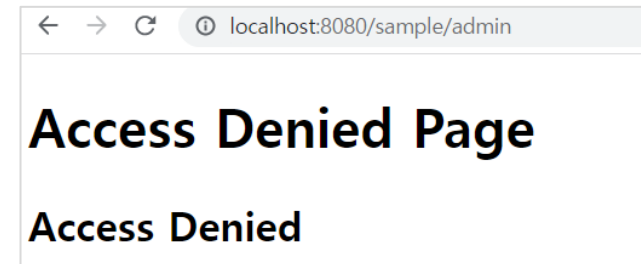
  - 로그인후 '/' 경로로 이동함

localhost:8080

Hello world!

The time on the server is 2022년 8월 27일 오전 8시 15분 17초 KST.
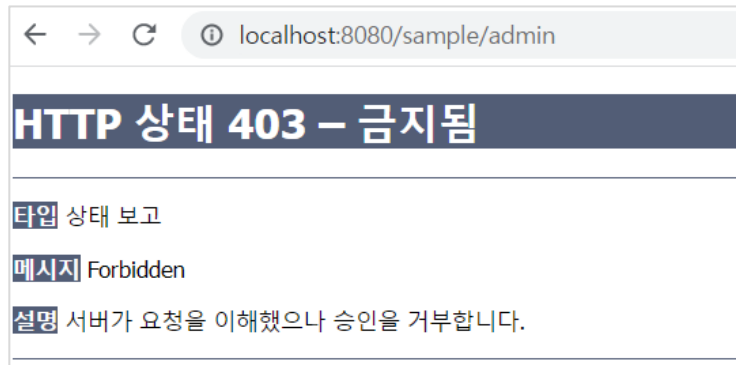
Spring Web Security

◆ 로그인과 로그아웃

- admin 사용자는 /sample/member, /sample/admin 모두 접근 가능

- member 사용자는 /sample/member 에만 접근 가능

✓ member사용자가 admin에 접근하면 에러 발생


```
        <security:form-login/>
        <security:logout/>
        <!-- 에러 처리 -->
        <security:access-denied-handler error-page="/accessError" />
</security:http>
```

Spring Web Security

◆ 접근 에러 처리 1

```java
@Log4j
@Controller
public class CommonController {

    @GetMapping("/accessError")
    public void accessDenied(Authentication auth, Model model) {
        Log.info("access Denied : " + auth);
        model.addAttribute("msg", "Access Denied");
    }
}
```

```html
<body>
    <h1>Access Denied Page</h1>

    <h2><c:out value="${msg}" /> </h2>
</body>
```

accessError.jsp

Spring Web Security

◆ **접근 에러 처리 2**

- **AccessDeniedHandler 인터페이스를 구현하는 경우**

```java
@Log4j
public class CustomAccessDeniedHandler implements AccessDeniedHandler{

    @Override
    public void handle(HttpServletRequest request, HttpServletResponse response,
            AccessDeniedException accessDeniedException) throws IOException, Serv
        Log.error("AccessDeniedHandler");

        Log.error("Redirect.....");

        response.sendRedirect("/accessError");
    }
}
```

```
✓ ⊞ com.spring.controller
  > 📄 CommonController.java
  > 📄 HomeController.java
  > 📄 SampleController.java
✓ ⊞ com.spring.security
  > 📄 CustomAccessDeniedHandler.java
  > 📄 CustomLoginSuccessHandler.java
```

Spring Web Security

◆ 접근 에러 처리 2

- CustomAccessDeniedHandler를 빈으로 등록

```xml
<!-- CustomAccessDeniedHandler를 빈으로 등록 -->
<bean id="customAccessDenied" class="com.spring.security.CustomAccessDeniedHandler"/>

<!-- 기본 서비스 -->
<security:http>
    <security:intercept-url pattern="/sample/all" access="permitAll" />
    <security:intercept-url pattern="/sample/member" access="hasRole('ROLE_MEMBER')" />
    <security:intercept-url pattern="/sample/admin" access="hasRole('ROLE_ADMIN')" />

    <security:form-login/>

    <!-- 에러 처리 -->
    <!-- <security:access-denied-handler error-page="/accessError" /> -->

    <security:access-denied-handler ref="customAccessDenied" />
</security:http>
```
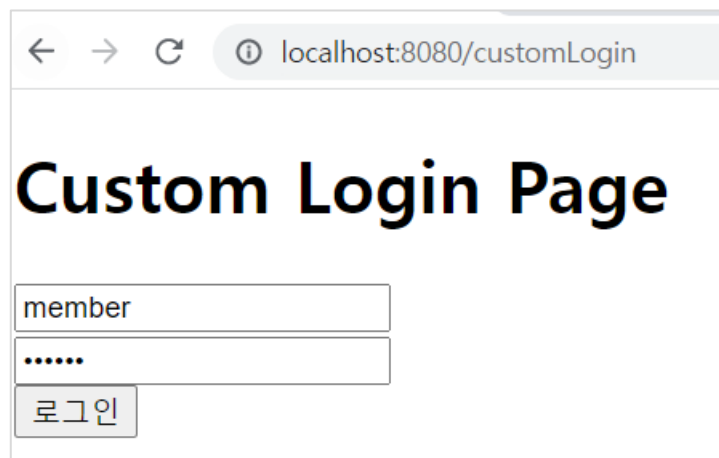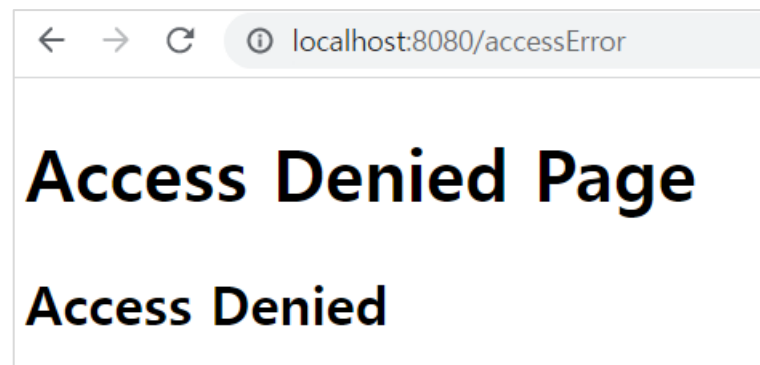
Spring Web Security

◆ 접근 에러 처리 2

Spring Web Security

◆ **커스텀 로그인 페이지**

- 스프링에서 제공하는 로그인 폼을 사용하지 않고 직접 로그인 폼
  을 제작해서 사용하기

```
<security:form-login/>
<!-- 로그인 페이지 직접 제작 -->
<security:form-login login-page="/customLogin" />
<security:logout logout-url="/customLogout" invalidate-session="true" />
```

Spring Web Security

◆ 로그인 페이지 만들기 – **CommonController.java**

```java
//로그인
@GetMapping("/customLogin")
public void login(String error, String logout, Model model) {
    Log.info("error: " + error);
    Log.info("logout:" + logout);

    if(error != null) {
        model.addAttribute("error", "Login Error Check Your Account");
    }

    if(logout != null) {
        model.addAttribute("logout", "LogOut!!");
    }
}
```

Spring Web Security

◆ **로그인 페이지 만들기 – customLogin.jsp**

```html
<h1>Custom Login Page</h1>

<h2><c:out value="${error}" /></h2>
<h2><c:out value="${logout}" /> </h2>

<form method="post" action="/login">
    <div>
        <input type="text" name="username" value="admin">
    </div>
    <div>
        <input type="password" name="password" value="admin">
    </div>
    <div>
        <input type="submit" value="로그인">
    </div>
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
</form>
```
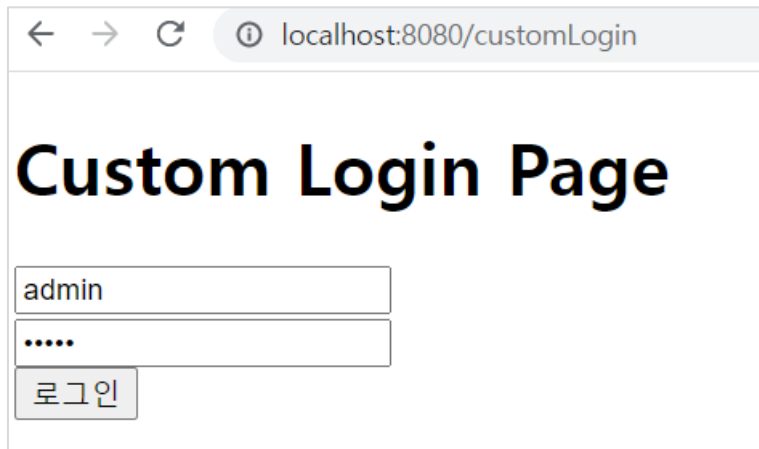
Spring Web Security

◆ 로그인 - **/sample/admin으로 요청**

◆ **CSRF(Cross-site request forgery) 공격과 토큰**

Spring Web Security

◆ 로그인 성공과 **AuthenticationSuccessHandler**

```java
@Log4j
public class CustomLoginSuccessHandler implements AuthenticationSuccessHandler{

    @Override
    public void onAuthenticationSuccess(HttpServletRequest request, HttpServletResp
            Authentication auth) throws IOException, ServletException {

        Log.warn("Login Success");

        List<String> roleNames = new ArrayList<>();

        auth.getAuthorities().forEach(authority -> { //람다식 반복문
            roleNames.add(authority.getAuthority()); //ROLE_ADMIN, ROLE_MEMBER 저장
        });
```

Spring Web Security

◆ 로그인 성공과 **AuthenticationSuccessHandler**

```
Log.warn("ROLE NAMES: " + roleNames);

if(roleNames.contains("ROLE_ADMIN")) {
    response.sendRedirect("/sample/admin");  //admin.jsp로 이동
    return;
}

if(roleNames.contains("ROLE_MEMBER")) {
    response.sendRedirect("/sample/member"); //member.jsp로 이동
    return;
}

response.sendRedirect("/");  //index.jsp로 이동
}
```

Spring Web Security

◆ 로그인 성공과 **AuthenticationSuccessHandler**

```xml
<!-- CustomAccessDeniedHandler를 빈으로 등록 -->
<bean id="customAccessDenied" class="com.spring.security.CustomAccessDeniedHandler" />
<bean id="customLoginSuccess" class="com.spring.security.CustomLoginSuccessHandler" />

<!-- 기본 서비스 -->
<security:http>
    <security:intercept-url pattern="/sample/all" access="permitALL" />
    <security:intercept-url pattern="/sample/member" access="hasRole('ROLE_MEMBER')" />
    <security:intercept-url pattern="/sample/admin" access="hasRole('ROLE_ADMIN')" />

    <!-- <security:form-login/> -->
    <!-- 로그인 페이지 직접 제작 -->
    <security:form-login login-page="/customLogin"
                         authentication-success-handler-ref="customLoginSuccess" />
    <security:logout logout-url="/customLogout" invalidate-session="true" />
```

Spring Web Security

◆ **/customLogin으로 직접 요청**

Spring Web Security

◆ 로그 아웃

```xml
<!-- <security:form-login/> -->
<!-- 로그인 페이지 직접 제작 -->
<security:form-login login-page="/customLogin"
                     authentication-success-handler-ref="customLoginSuccess" />

<security:logout logout-url="/customLogout" invalidate-session="true" />
```

```java
//로그아웃
@GetMapping("/customLogout")
public void logout() {
    Log.info("custom logout");
}
```

Spring Web Security

◆ 로그 아웃

```html
<h1>Logout Page</h1>

<h2><c:out value="${error}" /></h2>
<h2><c:out value="${logout}" /> </h2>

<form action="/customLogout" method="post" >
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
    <button type="submit">로그아웃</button>
</form>
```

logout.jsp

← → C ⓘ localhost:8080/customLogout

Logout Page

로그아웃

→

← → C ⓘ localhost:8080/customLogin?logout

Custom Login Page

LogOut!!

admin

•••••

로그인
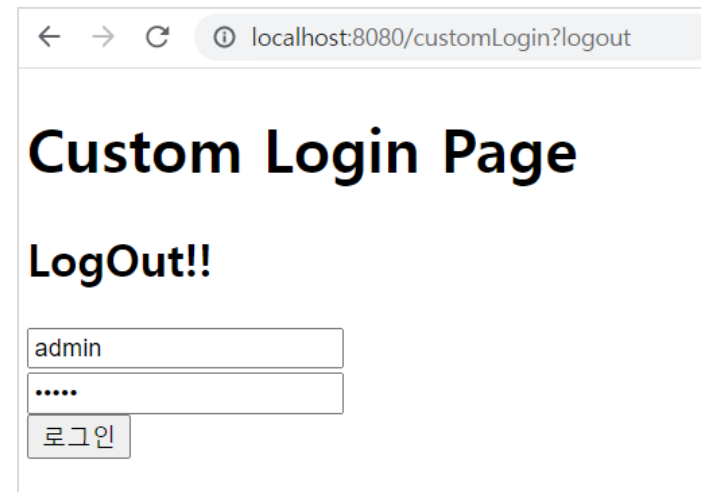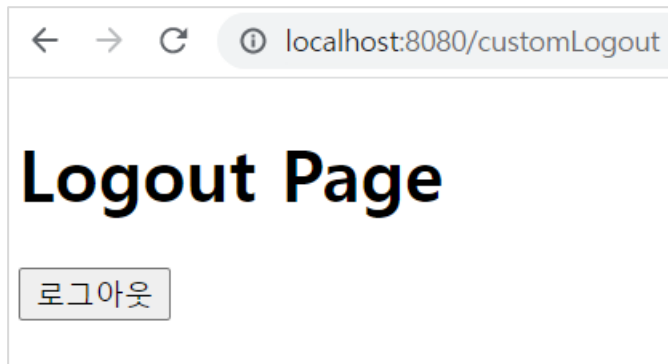
Spring Web Security

◆ **JDBC를 이용하는 간편 인증/권한 처리**

```sql
-- Spring security
CREATE TABLE users(
    username  VARCHAR2(50) NOT NULL PRIMARY KEY,
    password  VARCHAR2(50) NOT NULL,
    enabled   CHAR(1) DEFAULT '1'
);


CREATE TABLE authorities(
    username  VARCHAR2(50) NOT NULL,
    authority VARCHAR2(50) NOT NULL,
    CONSTRAINT fk_authorities_users FOREIGN KEY(username)
    REFERENCES users(username)
);
-- index 생성
CREATE UNIQUE INDEX ix_auth_username ON authorities (username, authority);
```

Spring Web Security

◆ JDBC를 이용하는 간편 인증/권한 처리

```
-- users 추가
INSERT INTO users(username, password) VALUES ('user00', 'pw00');
INSERT INTO users(username, password) VALUES ('member00', 'pw00');
INSERT INTO users(username, password) VALUES ('admin00', 'pw00');

-- 권한 추가
INSERT INTO authorities(username, authority) VALUES ('user00', 'ROLE_USER');
INSERT INTO authorities(username, authority) VALUES ('member00', 'ROLE_MEMBER');
INSERT INTO authorities(username, authority) VALUES ('admin00', 'ROLE_MEMBER');
INSERT INTO authorities(username, authority) VALUES ('admin00', 'ROLE_ADMIN');
```

Spring Web Security

◆ **JDBC를 이용하는 간편 인증/권한 처리**

```xml
<!-- DataSource 설정 -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
    <property name="driverClassName" value="oracle.jdbc.OracleDriver" />
    <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe" />
    <property name="username" value="c##spring" />
    <property name="password" value="spring" />
</bean>

<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource"
      destroy-method="close">
    <constructor-arg ref="hikariConfig" />
```

Spring Web Security

◆ JDBC를 이용하는 간편 인증/권한 처리

```xml
<!-- 인증 및 권한 설정 -->
<security:authentication-manager>
    <security:authentication-provider>
        <security:jdbc-user-service data-source-ref="dataSource" />
        <security:password-encoder ref="customPasswordEncoder" />

        <!-- <security:user-service>
            <security:user name="member" password="{noop}member"
                            authorities="ROLE_MEMBER"/>
            <security:user name="admin" password="{noop}admin"
                            authorities="ROLE_MEMBER, ROLE_ADMIN"/>
        </security:user-service> -->
    </security:authentication-provider>
</security:authentication-manager>
```
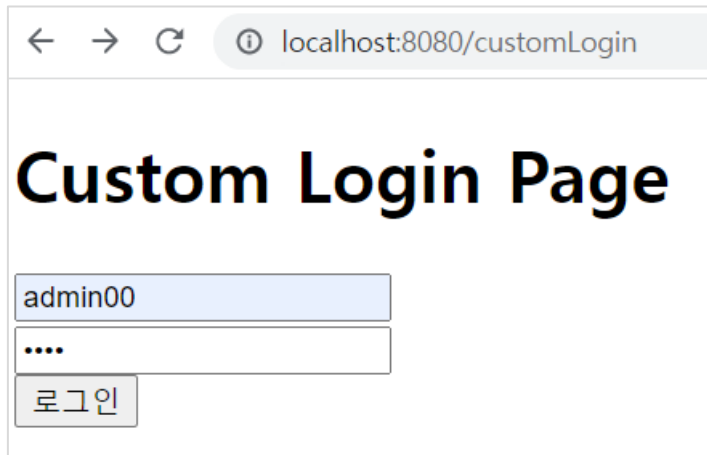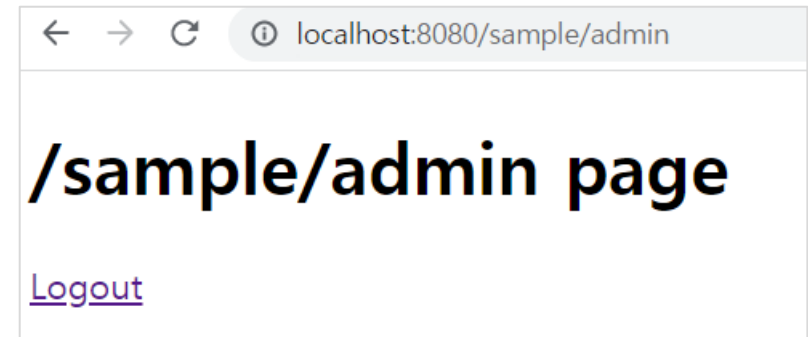
Spring Web Security

◆ JDBC를 이용하는 간편 인증/권한 처리

Spring Web Security

◆ **JDBC를 이용하는 간편 인증/권한 처리**

```java
@Log4j
public class CustomNoOpPasswordEncoder implements PasswordEncoder{

    @Override
    public String encode(CharSequence rawPassword) {
        Log.warn("before encode :" + rawPassword);
        return rawPassword.toString() ;
    }

    @Override
    public boolean matches(CharSequence rawPassword, String encodedPassword) {
        Log.warn("matches: " + rawPassword + ":" + encodedPassword);
        return rawPassword.toString().equals(encodedPassword);
    }
}
```

Spring Web Security

◆ 인증/권한을 위한 테이블 설계

```sql
-- Security 회원 테이블과 권한 테이블
CREATE TABLE tbl_member(
    userid   VARCHAR2(50) PRIMARY KEY,
    userpw   VARCHAR2(100) NOT NULL,
    username VARCHAR2(100) NOT NULL,
    regdate   DATE DEFAULT SYSDATE,
    updatedate   DATE DEFAULT SYSDATE,
    enabled CHAR(1) DEFAULT '1'
);


CREATE TABLE tbl_member_auth(
    auth   VARCHAR2(50) NOT NULL,
    userid   VARCHAR2(50) NOT NULL,
    CONSTRAINT fk_member_auth FOREIGN KEY (userid)
    REFERENCES tbl_member(userid)
);
```

Spring Web Security

◆ **BcryptPasswordEncoder 빈 설정**

```xml
<!-- 암호화 하지 않은 패스워드 -->
<!-- <bean id="customPasswordEncoder" class="com.spring.security.CustomNoOpPassword
<!-- 암호화 한 패스워드 -->
<bean id="bcryptPasswordEncoder"
      class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder" />
```

```xml
<!-- 암호화 -->
<security:password-encoder ref="bcryptPasswordEncoder" />
```

Spring Web Security

◆ 사용자 입력 및 권한 주기

```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/security-context.xml")
public class MemberTests {

    @Autowired
    private PasswordEncoder pwencoder;

    @Autowired
    private DataSource ds;

    //회원 가입 - 100명
    @Test
    public void testInsertMember() {
        String sql = "INSERT INTO tbl_member(userid, userpw, username) VALUES(?,?,?)";

        for(int i=0; i<100; i++) {
            Connection conn = null;
            PreparedStatement pstmt = null;
```

Spring Web Security

```java
try {
    conn = ds.getConnection();
    pstmt = conn.prepareStatement(sql);
    pstmt.setString(2, pwencoder.encode("pw" + i));
    if(i < 80) {
        pstmt.setString(1, "user" + i);
        pstmt.setString(3, "일반사용자" + i);
    }else if(i < 90) {
        pstmt.setString(1, "member" + i);
        pstmt.setString(3, "회원" + i);
    }else {
        pstmt.setString(1, "admin" + i);
        pstmt.setString(3, "관리자" + i);
    }

    pstmt.executeUpdate();    //실행

} catch (Exception e) {
    e.printStackTrace();
} finally {
    if(pstmt != null) {
        try {
            pstmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

Spring Web Security

	USERID	USERPW	USERNAME	REGDATE	UPDATEDATE	ENABLED
1	user0	$2a$10$aWCBpmidaNp99rz3Wkk9EuIouYGt0nGse5hoiGjCo6XCAP6x.LzB6	일반사용자0	22/08/27	22/08/27	1
2	user1	$2a$10$GVYIJfc0r1bvAPPsKJbnQ./pafFX9V6b4f0BlCMKrZu9Zz72WqYZi	일반사용자1	22/08/27	22/08/27	1
3	user2	$2a$10$26DeW8Pp6GelZKmMyCG1kOKgZC0oIYbzxhYdtdCzHVVbaDdXwJ3BG	일반사용자2	22/08/27	22/08/27	1
4	user3	$2a$10$46z/EohunFxjwJLYaybPC.DAKlt7yBsRJ5iBkcRSWs9apfL/dNBdO	일반사용자3	22/08/27	22/08/27	1
5	user4	$2a$10$w7/8YpeF2FluOKxYHWInKOBVk0Qos7cgc3OlfZvjB./ynvON.caOK	일반사용자4	22/08/27	22/08/27	1
6	user5	$2a$10$40QfPWj9Ms2R9sgP3fchR.NK5Uskd5s.iF8x72TkgqOjDObf4RE1S	일반사용자5	22/08/27	22/08/27	1
7	user6	$2a$10$PbWIAcnnC5SMzkfSCZpILu9yBjUPHQPDb7pxkJxK/NMM.gp/GvWAm	일반사용자6	22/08/27	22/08/27	1
8	user7	$2a$10$dWJPulzT/cIHpPXkIQnLPOLQLE15hgWiCHxF/XZ1yiNATDB2JwBpu	일반사용자7	22/08/27	22/08/27	1
9	user8	$2a$10$1tEX8xmlQrA1Rt/HI/TXfejxLY0jmU19m1hINaZyfP0qXNUIibyVK	일반사용자8	22/08/27	22/08/27	1

85	member84	$2a$10$v.RI1D331Q6lo3igRCfNrug1LYTEYwTh/qCfaJWX11y9GWWT/XqRK	회원84	22/08/27	22/08/27	1
86	member85	$2a$10$he9izvrYBHdWoutmv8EFFO6NAsmbE42yvQj02ka2emO.Uq0cSVSNu	회원85	22/08/27	22/08/27	1
87	member86	$2a$10$UmuRA8oChiJVV/Hq9v38feogqw7gGfSvzbysWlTjXwMwyaXzW2L4y	회원86	22/08/27	22/08/27	1
88	member87	$2a$10$CfTa2VOEghHg0AV4uMx5JermwBlPtjaY4dRe.al7.JZlJk04HelDa	회원87	22/08/27	22/08/27	1
89	member88	$2a$10$N7oSoRU6gVY8AYzj41y84uoDtNKmH2.2E.Kbtg9zdPg/o/9YwRqUy	회원88	22/08/27	22/08/27	1
90	member89	$2a$10$uBunowakGrLvxSgcfLpaEehquvdVgi1SWsIhMRPpZnm9gWW0csNpS	회원89	22/08/27	22/08/27	1
91	admin90	$2a$10$SeN/MdY1ISSJHJrENHN.KuwoyqjQHEr7fotn3VH9aeW9fAJ438kaK	관리자90	22/08/27	22/08/27	1
92	admin91	$2a$10$bd8X..RKk4.GNG09wTqPOu2NCPa4/1rnd6D56a6qe4YKDca8S8myq	관리자91	22/08/27	22/08/27	1
93	admin92	$2a$10$N43m5raiMc1IEzCnu9uVLeAb79CQ0NDczOTrpY/wSbRAAJLs94qtK	관리자92	22/08/27	22/08/27	1
94	admin93	$2a$10$m/p1HHAyup8QlWZQe2VTR.XQ95E/DM/L56oti8KvKLW62pEvHhqiG	관리자93	22/08/27	22/08/27	1
95	admin94	$2a$10$BujmpDbEk8QPzNzzUt2OIOQjLnumRK8JhXMJk7IfhcfXrbTEV3xU6	관리자94	22/08/27	22/08/27	1
96	admin95	$2a$10$rG5CaOqrY6AHBVB/ZbX6Zui/mWumCXEAHnrqSzd52rCLPJpkNeq.G	관리자95	22/08/27	22/08/27	1

Spring Web Security

```java
//회원 권한 입력
@Test
public void testInsertAuth() {
    String sql = "INSERT INTO tbl_member_auth(userid, auth) VALUES(?,?)";

    for(int i=0; i<100; i++) {
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            conn = ds.getConnection();
            pstmt = conn.prepareStatement(sql);

            if(i < 80) {
                pstmt.setString(1, "user"+i);
                pstmt.setString(2, "ROLE_USER");
            }else if(i < 90) {
                pstmt.setString(1, "member"+i);
                pstmt.setString(2, "ROLE_MEMBER");
            }else {
                pstmt.setString(1, "admin"+i);
                pstmt.setString(2, "ROLE_ADMIN");
            }

            pstmt.executeUpdate();
```

Spring Web Security

76	ROLE_USER	user75
77	ROLE_USER	user76
78	ROLE_USER	user77
79	ROLE_USER	user78
80	ROLE_USER	user79
81	ROLE_MEMBER	member80
82	ROLE_MEMBER	member81
83	ROLE_MEMBER	member82
84	ROLE_MEMBER	member83
85	ROLE_MEMBER	member84
86	ROLE_MEMBER	member85
87	ROLE_MEMBER	member86
88	ROLE_MEMBER	member87
89	ROLE_MEMBER	member88
90	ROLE_MEMBER	member89
91	ROLE_ADMIN	admin90
92	ROLE_ADMIN	admin91
93	ROLE_ADMIN	admin92
94	ROLE_ADMIN	admin93
95	ROLE_ADMIN	admin94

Spring Web Security

◆ 쿼리를 이용하는 인증

```xml
<security:authentication-provider>
    <!-- jdbc 연동 -->
    <!-- <security:jdbc-user-service data-source-ref="dataSource" /> -->

    <!-- 패스워드 암호화 쿼리 사용 -->
    <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=?" />
```
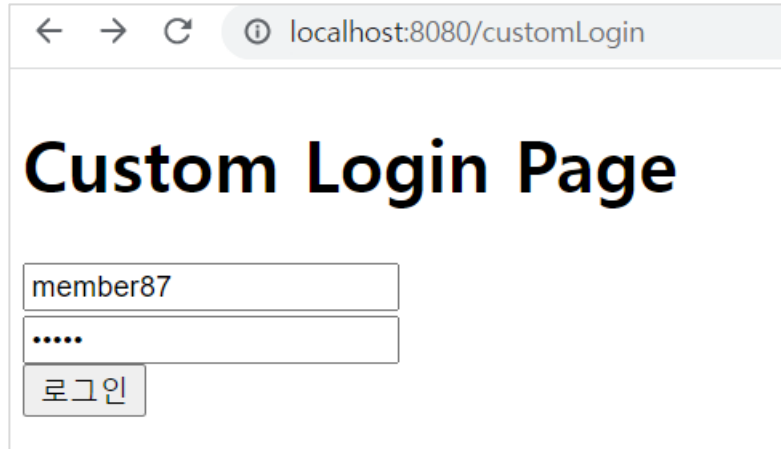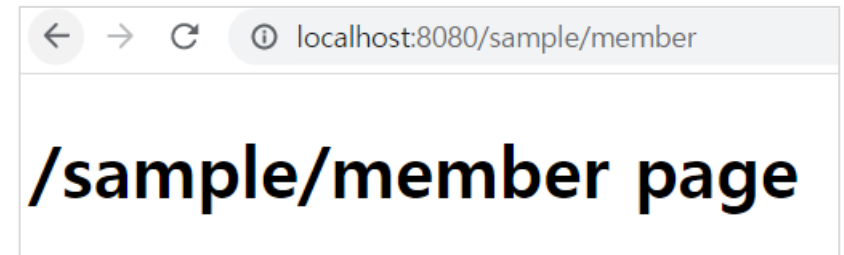
Spring Web Security

◆ **Member87/pw87로 인증**