1. 소개
  1)웹 보안 개요
    -All-in-One Java 애플리케이션 개발, 전병선, 2014 참조
    -웹 어플리케이션을 작성하면서 보안과 관련해서 고려할 사항은 다음과 같다.
    -인증(authentication) : 사용자가 자신임을 증명하는 속성
    -권한(authorization) : 사용자가 어떤 리소스에 접근하게 할지를 결정하는 속성
    -기밀성(privacy or confidentiality) : 권한이 없는 사용자에게 접근을 허용하지 않게 하는 속성
    -무결성(integrity) : 권한이 없는 사용자가 데이터를 변경하지 못하게 하는 속성
    -부인방지(repudiation) : 사용자가 자신이 했다는 것을 부인하지 못하게 하는 속성

  2)Spring Security는 이들 보안 요소 중에서 인증과 권한 service를 제공한다.
  3)인증은 사용자가 자신임을 증명하는 것이고, 권한은 사용자가 어떤 resource에 접근할 수 있는지를 결정하는 것이다.
  4)우리가 Web Application의 resource에 접근할 수 있는지 여부는 권한과 관련되어 있다.
  5)사용자가 권한이 있는지를 알기 위해서는 권한을 가진 사용자인지 인증되어야 한다.
  6)그러니까 먼저 사용자가 인증되어야 하고, 인증된 사용자에 부여된 권한으로 resource에 접근할 수 있는지를 check한다.
  7)사용자를 인증하는 다양한 방법이 있지만 가장 보편적으로 사용하는 방법은 form 인증 방식이다.
  8)사용자가 인증된 후에는 resource에 접근할 수 있는 권한을 가졌는지를 check해야 한다.
  9)사용자의 권한을 수립하는 첫 번째 단계는 principle 형식으로 사용자를 표현하는 것이다.
  10)principle은 resource에 접근하기 위한 보안 식별자가 할당된 계정 보유자로서, 사용자, group, service, computer 등이 principle이 될 수 있다.
  11)개별 사용자에 대해 일일이 권한을 가졌는지를 검사하는 것을 비효율적일 뿐만 아니라 경우에 따라서는 불가능하기도 하다.
  12)따라서, 사용자에게 권한을 부여할 때 가장 많이 사용하는 일반적인 방법은 사용자를 역할(role)에 할당하는 역할 기반 방식이다.
  13)ROLE_ADMIN, ROLE_USER, ROLE_MNAGER 등의 역할을 정의하고, 개별 사용자에게 역할을 할당한다.

2. 환경설정
  1)Spring Legacy Project 생성
    -In Package Explorer > right-click > New > Spring Legacy Project
    -Project name : SpringSecurityDemo
    -Select Spring MVC Project > Next
    -com.example.biz > Finish

  2)server.xml에 project 추가
    -Tomcat v9.0 Server at localhost > right-click > Add and Remove
    -SpringSecurityDemo > Add > Finish > OK

  3)Spring security library download and install
    -In pom.xml 아래와 같이 수정
      <java-version>1.8</java-version>
      <org.springframework-version>4.3.24.RELEASE</org.springframework-version>
      <org.aspectj-version>1.9.4</org.aspectj-version>
      <org.slf4j-version>1.7.26</org.slf4j-version>
      ...
      <!-- Servlet -->
      <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <scope>provided</scope>
      </dependency>
      <dependency>

```
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>javax.servlet.jsp-api</artifactId>
        <version>2.3.3</version>
        <scope>provided</scope>
      </dependency>
      ...
      <!-- Test -->
      <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
      </dependency>

      <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
      </dependency>

  -In mvnrepository, search 'spring security'
  -Select 'Spring Security Core' 4.2.6
  -Select 'Spring Security Web' 4.2.6
  -Select 'Spring Security Config' 4.2.6
  -Select 'Spring Security Taglibs' 4.2.6
  -pom.xml > right-click > Run As > Maven clean and Maven install

  -spring-security-core
    --Spring security framework의 핵심 class와 interface를 정의한다.
    --이 module은 Spring security를 사용하는 모든 application에 필수 항목이다.
  -spring-security-web
    --Web application의 보안을 위한 지원을 제공한다.
  -spring-security-config
    --Spring security는 Spring tx 및 mvc schema와 비슷하게 Spring security의 기능을 간편하게 구성하기 위한 security schema를 제공한다.
    --security namespace의 요소를 구문분석한다.
  -spring-security-taglibs
    --보안 정보에 접근하고 JSP page에 표시된 내용을 보호하는 tag를 정의한다.
  -spring-security-acl
    --ACL(접근 제어 목록)을 사용해 application에서 domain 객체의 instance를 보호할 수 있게 한다.

4)Web 요청 보안 구성
  -Application에서 web 요청을 보호하는 방법은 다음과 같다.
    --web.xml에서 Spring의 DelegatingFilterProxy filter를 구성한다.
    --Spring Security Framework에서 제공하는 요청 보안을 활성화한다.

5)DelegatingFilterProxy filter 구성
  -spring-web-4.x.x.RELEASE.jar file에 들어있는 Spring Framework의 web module servlet API의 Filter interface를 구현하는 DelegatingFilterProxy를 정의한다.
  -In web.xml 에 code 추가
    --반드시 filter의 이름은 springSecurityFilterChain 이어야 한다.

    <filter>
```

```
103         <filter-name>springSecurityFilterChain</filter-name>
104         <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
105       </filter>
106
107       <filter-mapping>
108         <filter-name>springSecurityFilterChain</filter-name>
109         <url-pattern>/*</url-pattern>
110       </filter-mapping>
111
112     -위의 code에서 <filter-mapping>요소에는 DelegatingFilterProxy filter를 들어오는 모든 web 요청과 연결
        하도록 지정했다.
113     -<filter-name> 요소에는 DelegatingFilterProxy filter가 요청을 처리하도록 위임할 Spring bean의 이름을
        지정한다.
114     -이 code에서는 DelegatingFilterProxy filter가 수신하는 web 요청은 root application context의
        springSecurityFilterChain이라는 Spring bean에 위임된다.
115
116   6)보안 관련 설정 file 만들기
117     -/src/main/webapp/WEB-INF/conig folder 생성
118     -project > right-click > Build Path > Config Build Path
119     -Source tab > Add Folder > select src/main/webapp/WEB-INF/
120     -Click [Create New Folder] > Folder name : config > OK > Apply and Close
121     -src/main/webapp/WEB-INF/config > right-click > New > Spring Bean Configuration File
122     -File name : securityContext.xml > Next
123     -Check beans, security > Finish
124     -In /WEB-INF/web.xml에 보안 관련 설정 file 등록
125
126       <context-param>
127         <param-name>contextConfigLocation</param-name>
128         <param-value>classpath:securityContext.xml</param-value>
129       </context-param>
130
131   7)Web 요청에 대한 보안 구성
132     -DelegatingFilterProxy filter를 구성했으니 이제 web 요청에 대한 보안을  구성하도록 하겠다.
133     -In securityContext.xml code 추가
134
135       <security:http auto-config="true">
136         <security:intercept-url pattern="/login.html" access="hasRole('ROLE_USER')" />
137         <security:intercept-url pattern="/welcome.html" access="hasRole('ROLE_ADMIN')" />
138       </security:http>
139
140     -auto-config='true'를 설정한 것만으로 기본 login page/HTTP 기본인증/logout 기능등을 제공한다.
141     -만일 여기서 use-expressions='true'라고 하면 SpEL을 사용한다는 의미이다.
142     -use-expressions은 기본값이 false이다.
143     -이럴때에는 SpEL을 사용하지 않는다는 의미이다.
144
145     -<intercept-url pattern="..." access="ROLE_ANONYMOUS" />
146     -<intercept-url pattern="..." access="IS_AUTHENTICATED_ANONYMOUSLY" />
147     -<intercept-url pattern="..." access="ROLE_USER" />
148     -<intercept-url pattern="..." access="ROLE_ADMIN" />
149     -<intercept-url> 요소의 access 속성은 bool 값으로 평가되는 SpEL 식이다.
150     -해당 URL 에 접근하기위한 권한을 설정하여준다.
151     -접근가능한 IP 등을 설정할 수도 있다.
152     -그리고 권한은 위쪽이 우선시된다.
153     -만일 SpEL식이 true를 반환하는 경우 사용자는 pattern과 일치하는 URL에 접근할 수 있으며 false를 반환하는 경
```

우 pattern 속성과 일치하는 URL에 대한 접근이 거부된다.
-Spring Security Framework에는 hasRole, hasAnyRole, isAnonymous 등의 몇 가지 기본식을 제공한다.
-만일 hasAnyRole('ROLE_CUSTOMER', 'ROLE_ADMIN')식은 사용자에게 ROLE_CUSTOMER 또는 ROLE_ADMIN 역할이 있는 경우 true를 반환한다.
-또한 pattern이 만일 /**이면 모든 URL과 일치한다는 뜻이다.

-<intercept-url pattern="/login" access="permitAll" />
  --/login 으로는 모두 허용해준다. /login 을 막아놓으면 안되니까.

-<intercept-url pattern="/resources/**" access="permitAll" />
  --Resource도 허용

-<intercept-url pattern="/**" access="hasRole('ROLE_USER')" />
  --나머지는 모두 ROLE_USER 권한을 가진사람만 허용해준다.

```
<security:authentication-manager>
  <security:authentication-provider>
    <security:user-service>
      <security:user name="javaexpert" password="12345678"
      authorities="ROLE_USER"/>
      <security:user name="admin" password="12345678"
      authorities="ROLE_ADMIN,ROLE_USER"/>
    </security:user-service>
  </security:authentication-provider>
</security:authentication-manager>
```

-<authentication-manager> 요소는 AuthenticationManager instance를 구성하고 <authentication-provider>는 AuthenticationProvider instance를 구성한다.
-기본적으로 <authentication-provider>는 Spring UserDetailsService를 DAO로 사용해서 지정된 사용자 이름을 참고해 사용자 저장소에서 사용자 세부 정보를 load한다.
-DaoAuthenticationProvider는 사용자가 제공한 자격 증명과 구성된 UserDetailsService가 load한 사용자 세부 정보를 비교해 인증을 수행한다.
-UserDetailsService는 데이터 원본(Data Source), 일반 file 또는 다른 사용자 저장소에서 사용자 세부 정보를 load할 수 있다.
-<authentication-provider> 하위의 <user-service>는 <user>가 정의하는 사용자를 load하는 memory 내 UserDetailsService를 구성한다.
-위의 예제에서는 ROLE_USER와 ROLE_ADMIN의 사용자를 정의했다.
-name 속성에는 사용자에게 할당된 사용자 이름을 지정하며 password 속성에는 사용자에게 할당된 암호를 지정한다.
-authorities 속성에는 사용자에게 할당된 role을 지정한다.

-<authentication-provider user-service-ref="memberService"/>
  --사용자이름/비밀번호를 제공해줄 서비스 등록
-위의 예제는 Inmemory로 처리하고 있음.

8)web.xml 에서 코드 수정

```
<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>*.html</url-pattern>  <--여기 수정할 것
</servlet-mapping>
```

9) Test
-/WEB-INF/views/security folder 생성
-/WEB-INF/views/security/login.jsp

```
199
200         <body>
201           <h1>login.jsp</h1>
202         </body>
203
204       -/WEB-INF/views/security/welcome.jsp
205
206         <body>
207           <h1>welcome.jsp</h1>
208         </body>
209
210       -/WEB-INF/views/home.jsp
211
212         <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
213         <%@ page session="false" %>
214         <html>
215         <head>
216           <title>Home</title>
217         </head>
218         <body>
219         <h1>
220           ${greeting}
221         </h1>
222
223         </body>
224         </html>
225
226       -HomeController.java 코드 수정
227
228         @RequestMapping(value = "/index.html", method = RequestMethod.GET)
229         public String home(Model model) {
230           model.addAttribute("greeting", "Hello! Spring Security");
231           return "home";
232         }
233
234         @RequestMapping("/login.html")
235         public String login(Model model) {
236           return "security/login";
237         }
238
239         @RequestMapping("/welcome.html")
240         public String welcome(Model model) {
241           return "security/welcome";
242         }
243
244       -project > right-click > Run As > Run on Server
245
246       -/login.html로 들어가면 정상적으로 login.jsp 로 이동한다.
247       -username과 password를 미리 저장된 값을 넣지 않으면 Bad credentials로 로그인에러 발생한다.
248       -/welcome.html로 들어가면 User와 Password를 입력할 수 있는 창으로 이동한다.
249       -여기서 user와 password를 잘못 입력하면 'Bad credentials' 에러 메시지가 나온다.
250       -User에 미리 설정한 javaexpert, Password에 12345678을 입력하면 403 Forbidden 오류가 나온다.
251       -즉 권한이 없다는 뜻이다.
252       -만일 미리 설정한 admin/12345678을 입력하면 정상적으로 welcome.jsp로 이동한다.
```

-왜냐하면 이 계정은 ROLE_ADMIN 권한을 갖고 있기 때문이다.

3. Login page 만들기

1)securityContext.xml code 변경

```
<security:http auto-config="true">
  <security:form-login login-page="/loginForm.html"
    username-parameter="j_username"
    password-parameter="j_password"
    login-processing-url="/j_spring_security_check" />
  <security:intercept-url pattern="/login.html"
    access="hasRole('ROLE_USER')" />
  <security:intercept-url
    pattern="/welcome.html" access="hasRole('ROLE_ADMIN')" />
  <security:csrf disabled="true"/>
</security:http>
```

-form 기반 인증을 사용하고자 할 때는 아래와 같이 설정한다.

```
<form-login login-page="/login"
  default-target-url="/monitering"
  username-parameter="username"
  password-parameter="password"
  authentication-failure-url="/login?error"
  always-use-default-target='true'
/>
```

-사용자가 만든 login page를 login-page 속성을 통해 Spring에게 알려준다.
-이거 설정을 안하면 Spring이 기본적으로 내장된 것을 사용.
-default-target-url="/monitering" 은 login 성공하면 이동할 page를 설정한다.
-authentication-failure-url="/login?error" 은 login 실패시 호출해줄 URL(login page에 error parameter를 보내준다)
-always-use-default-target='true' 이거 안하면 login 성공해도 /monitering 로 제대로 안갈 수 있다.

```
<logout invalidate-session="true" logout-url="/logout"   logout-success-url="/login?logout"
/>
```

-Logout되면 session을 초기화한다.
-logout-success-url="/login?logout" 은 logout되면 이동할 page
-logout-url="/logout" 은 logout을 위한 URL설정.
-이거 안해주면 default로 j_spring_security_logout 해주면됨.
-<csrf/> 요소는 간단한 설정으로 csrf 를 통한 해킹을 막을 수 있다.
(CSRF 설명: http://tm-csc.tistory.com/entry/7-WebHacking%EC%9B%B9%ED%95%B4%ED%82%B9-CSRF)

2)/views/security/loginForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
```

```
303     <head>
304     <meta charset="UTF-8">
305     <title>로그인 창</title>
306     </head>
307     <body>
308       <h1>loginForm.jsp</h1>
309
310       <form action="<c:url value="j_spring_security_check" />" method="post">
311         ID : <input type="text" name="j_username"> <br />
312         PW : <input type="text" name="j_password"> <br />
313         <input type="submit" value="LOGIN"> <br />
314       </form>
315     </body>
316     </html>
317
318   3)HomeController.java code 추가
319     @RequestMapping("/loginForm.html")
320     public String loginForm(Model model) {
321       return "security/loginForm";
322     }
323
324   4)Test
325     -project > right-click > Run As > Run on Server
326     -http://localhost:8080/biz/login.html 을 요청하면 http://localhost:8080/biz/loginForm.html 로
      redirect
327     -ID와 Password를 미리 설정한 javaexpert/12345678을 넣으면 정상적으로 login.jsp로 이동한다.
328     -http://localhost:8080/biz/welcome.html 을 요청하면 http://localhost:8080/biz/loginForm.html
      로 redirect
329     -만일 ID와 Password를 미리 설정한 javaexpert/12345678을 넣으면 403 forbidden error가 발생한다.
330     -만일 ID와 Password를 미리 설정한 admin/12345678을 넣으면 정상적으로 welcome.jsp로 이동한다.
331     -만일 잘못된 ID와 Password를 입력하면 계속 loginForm.html?error가 나타난다.
332
333     -만일 Could not verify the provided CSRF token because your session was not found. 오류가 발생
      하면 아래의 조치를 취한다.
334       --securityContext.xml 에서 코드 추가한다.
335         <security:http auto-config="true">
336           <security:form-login login-page="/loginForm.html" />
337           <security:intercept-url pattern="/login.html"
338             access="hasRole('ROLE_USER')" />
339           <security:intercept-url
340             pattern="/welcome.html" access="hasRole('ROLE_ADMIN')" />
341           <security:csrf disabled="true"/>          <---cpde 추가
342         </security:http>
343
344
345 4. login 실패시 실패 message 출력하기
346   1)securityContext.xml code 추가
347
348     <security:form-login login-page="/loginForm.html"
349       username-parameter="j_username"
350       password-parameter="j_password"
351       authentication-failure-url="/loginForm.html?ng"    <---code 추가
352       login-processing-url="/j_spring_security_check" />
353
```

2)loginForm.jsp 코드 추가

```
<h1>loginForm.jsp</h1>
<c:url value="j_spring_security_check" var="loginUrl" />  <--추가
<form action="${loginUrl}" method="post">                 <-- 수정
  <c:if test="${param.ng != null}">
    <p>
      Login Failure<br />
      <c:if test="${SPRING_SECURITY_LAST_EXCEPTION != null}">
        Message : <c:out value="${SPRING_SECURITY_LAST_EXCEPTION.message}" />
      </c:if>
    </p>
  </c:if>
  ID : <input type="text" name="j_username"> <br />
  PW : <input type="text" name="j_password"> <br />
  <input type="submit" value="LOGIN"> <br />
</form>
```

3)Test

-http://localhost:8080/biz/login.html 을 요청하면 http://localhost:8080/biz/loginForm.html 로 redirect

-ID와 Password를 미리 설정한 javaexpert/12345678을 넣으면 정상적으로 login.jsp로 이동한다.

-만일 잘못된 ID나 Password를 입력하면 아래와 같은 오류 message가 나온다.

```
Login Failure
Message : Bad credentials
```

-http://localhost:8080/biz/welcome.html 을 요청하면 http://localhost:8080/biz/loginForm.html 로 redirect

-만일 ID와 Password를 미리 설정한 javaexpert/12345678을 넣으면 403 forbidden error가 발생한다.

-만일 ID와 Password를 미리 설정한 admin/12345678을 넣으면 정상적으로 welcome.jsp로 이동한다.

-만일 잘못된 ID와 Password를 입력하면 계속 loginForm.html?ng가 나오면서 아래의 message가 나타난다.

```
Login Failure
Message : Bad credentials
```

5. login, logout 상태 표시하기

1)security/login.jsp page code 추가

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<body>
  <h1>login.jsp</h1>

  <c:if test="${not empty pageContext.request.userPrincipal}">
    <p>현재 로그인 상태</p>
  </c:if>
  <c:if test="${empty pageContext.request.userPrincipal}">
    <p>현재 로그아웃 상태</p>
  </c:if>

  User ID : ${pageContext.request.userPrincipal}<br />
  <a href="${pageContext.request.contextPath}/">Log Out</a>
</body>
```

2)Test
-http://localhost:8080/biz/login.html 을 요청하면 http://localhost:8080/biz/loginForm.html 로 redirect
-아이디와 패스워드를 미리 설정한 javaexpert/12345678을 넣으면 정상적으로 login.jsp로 이동한다.
-이 때 화면에는 아래의 message가 나온다.

현재 로그인 상태
User ID : org.springframework.security.authentication.UsernamePasswordAuthenticationToken@b09a9234:
Principal: org.springframework.security.core.userdetails.User@f486f6cc:
Username: javaexpert;
Password: [PROTECTED];
Enabled: true;
AccountNonExpired: true;
credentialsNonExpired: true;
AccountNonLocked: true;
Granted Authorities: ROLE_USER;
Credentials: [PROTECTED];
Authenticated: true;
Details: org.springframework.security.web.authentication.WebAuthenticationDetails@166c8:
RemoteIpAddress: 0:0:0:0:0:0:0:1;
SessionId: 949A2731FFC91E3C350818E0772C8E45;
Granted Authorities: ROLE_USER

Log Out

3)여기서 username만 출력하려면

User ID : ${pageContext.request.userPrincipal.name} 으로 변경한다.

4)securityContext.xml code 추가

```
<security:form-login login-page="/loginForm.html"
  username-parameter="j_username"
  password-parameter="j_password"
  authentication-failure-url="/loginForm.html?ng"
  login-processing-url="/j_spring_security_check" />
<security:logout logout-url="/j_spring_security_logout"/>    <--code 추가
```

5)Test
-로그인 창에서 Log Out link를 클릭하면 index.html로 이동한다.

6. Lab

1)Spring Legacy Project 생성
-In Package Explorer > right-click > New > Spring Legacy Project
-Project name : SpringSecurityDemo1
-Select Spring MVC Project > Next
-com.example.biz > Finish

2)server.xml에 project 추가

```
   -Tomcat v9.0 Server at localhost > right-click > Add and Remove
   -SpringSecurityDemo1 > Add > Finish > OK

3)Spring security library download and install
   -In pom.xml 아래와 같이 수정
      <java-version>1.8</java-version>
      <org.springframework-version>4.3.24.RELEASE</org.springframework-version>
      <org.aspectj-version>1.9.4</org.aspectj-version>
      <org.slf4j-version>1.7.26</org.slf4j-version>

   -version 수정
      <dependency>
         <groupId>javax.servlet</groupId>
         <artifactId>javax.servlet-api</artifactId>
         <version>4.0.1</version>
         <scope>provided</scope>
      </dependency>
      <dependency>
         <groupId>javax.servlet.jsp</groupId>
         <artifactId>javax.servlet.jsp-api</artifactId>
         <version>2.3.3</version>
         <scope>provided</scope>
      </dependency>
      <dependency>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <version>4.12</version>
         <scope>test</scope>
      </dependency>

   -In mvnrepository, search 'spring security'
      -Select 'Spring Security Core' 4.2.12
      -Select 'Spring Security Web' 4.2.12
      -Select 'Spring Security Config' 4.2.12
      -Select 'Spring Security Taglibs' 4.2.12

      <dependency>
         <groupId>org.springframework.security</groupId>
         <artifactId>spring-security-core</artifactId>
         <version>4.2.12.RELEASE</version>
      </dependency>
      <dependency>
         <groupId>org.springframework.security</groupId>
         <artifactId>spring-security-web</artifactId>
         <version>4.2.12.RELEASE</version>
      </dependency>
      <dependency>
         <groupId>org.springframework.security</groupId>
         <artifactId>spring-security-config</artifactId>
         <version>4.2.12.RELEASE</version>
      </dependency>
      <dependency>
         <groupId>org.springframework.security</groupId>
         <artifactId>spring-security-taglibs</artifactId>
```

```
510         <version>4.2.12.RELEASE</version>
511       </dependency>
512
513     -pom.xml > right-click > Run As > Maven clean and Maven install
514     -project > right-click > Properties > Project Facets > Java > Select 1.8
515     -Runtimes Tab > Check Apache Tomcat v9.0
516     -Click Apply and Close
517
518   4)HomeController.java 코드 수정
519
520     package com.example.biz;
521
522     import java.util.Date;
523
524     import org.springframework.stereotype.Controller;
525     import org.springframework.ui.Model;
526     import org.springframework.web.bind.annotation.RequestMapping;
527     import org.springframework.web.bind.annotation.RequestMethod;
528
529     @Controller
530     public class HomeController {
531
532       @RequestMapping(value = "/", method = RequestMethod.GET)
533       public String home(Model model) {
534         model.addAttribute("serverTime", new Date());
535         return "home";
536       }
537     }
538
539   5)project > right-click > Run As > Run on Server
540
541   6)src/main/webapp/WEB-INF/spring/Spring Bean Configuration File > spring-security.xml 파일
      생성
542     -check 'security'
543
544     <security:http auto-config="true" use-expressions="true">
545     </security:http>
546
547     <security:authentication-manager>
548     </security:authentication-manager>
549
550   7)web.xml 코드 수정
551
552     <context-param>
553       <param-name>contextConfigLocation</param-name>
554       <param-value>/WEB-INF/spring/spring-security.xml</param-value>
555     </context-param>
556     ...
557     ...
558     <filter>
559       <filter-name>springSecurityFilterChain</filter-name>
560       <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
561     </filter>
562
```

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

8)WEB-INF/spring/appServlet/servlet-context.xml

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

9)project > right-click > Run As > Run on Server

10)spring-security.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.springframework.org/schema/security/spring-security-4.2.xsd
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">

    <security:http auto-config="true" use-expressions="true">
      <security:intercept-url pattern="/login"
        access="permitAll" />
      <security:intercept-url pattern="resources/**"
        access="permitAll" />
      <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')" />
      <security:form-login login-page="/login"
        default-target-url="/welcome" username-parameter="j_username"
        password-parameter="j_password"
        authentication-failure-url="/login?error"
        login-processing-url="/j_spring_security_check"
        always-use-default-target="true" />
      <security:logout invalidate-session="true"
        logout-url="/logout" logout-success-url="/login?logout" />
      <security:csrf />
    </security:http>

    <security:authentication-manager>
      <!-- <security:authentication-provider>
        <security:user-service>
          <security:user name="javaexpert" password="12345678"
            authorities="ROLE_USER" />
          <security:user name="admin" password="12345678"
            authorities="ROLE_ADMIN,ROLE_USER" />
        </security:user-service>
      </security:authentication-provider> -->
      <security:authentication-provider user-service-ref="memberService" />
    </security:authentication-manager>

    <bean id="memberService"
      class="com.example.service.MemberService">
```

```
        <property name="userService" ref="userService" />
      </bean>
      <bean id="userService"
        class="com.example.service.UserService">
        <property name="userDao" ref="userDao" />
      </bean>
      <bean id="userDao" class="com.example.dao.UserDao">
        <property name="username" value="javaexpert" />
        <property name="password" value="12345678" />
      </bean>
    </beans>

9)HomeController.java

    package com.example.biz;

    import java.util.Date;

    import org.springframework.stereotype.Controller;
    import org.springframework.ui.Model;
    import org.springframework.web.bind.annotation.RequestMapping;
    import org.springframework.web.bind.annotation.RequestMethod;
    import org.springframework.web.bind.annotation.RequestParam;
    import org.springframework.web.servlet.ModelAndView;

    @Controller
    public class HomeController {

      @RequestMapping(value = "/", method = RequestMethod.GET)
      public String home(Model model) {
        model.addAttribute("serverTime", new Date());
        return "home";
      }

      @RequestMapping(value = "/login", method = RequestMethod.GET)
      public ModelAndView login(
        @RequestParam(value = "error", required = false) String error,
        @RequestParam(value = "logout", required = false) String logout) {
        ModelAndView model = new ModelAndView();
        if (error != null) {
          model.addObject("error", "Invalid username and password!");
        }
        if (logout != null) {
          model.addObject("msg", "You've been logged out successfully.");
        }
        model.setViewName("login");
        return model;
      }

      @RequestMapping("/welcome")
      public String welcome(Model model) {
        return "welcome";
      }

```

```
    @RequestMapping(value = "/logout", method = RequestMethod.GET)
    public String logout(Model model) {
      return "logout";
    }
  }


10)com.example.vo.UserVO.java

  package com.example.vo;

  public class UserVO {
    private String username;
    private String password;

    public UserVO() {}

    public UserVO(String username, String password) {
      this.username = username;
      this.password = password;
    }

    public String getUsername() {
      return username;
    }

    public void setUsername(String username) {
      this.username = username;
    }

    public String getPassword() {
      return password;
    }

    public void setPassword(String password) {
      this.password = password;
    }

    @Override
    public String toString() {
      return "UserVO [username=" + username + ", password=" + password + "]";
    }
  }

11)com.example.dao.UserDao.java

  package com.example.dao;

  import org.springframework.stereotype.Repository;

  import com.example.vo.UserVO;

  @Repository("userDao")
  public class UserDao {
```

```
723       private String username;
724       private String password;
725
726       public void setUsername(String username) {
727         this.username = username;
728       }
729
730       public void setPassword(String password) {
731         this.password = password;
732       }
733
734       public UserVO getUsersByID(String username) {
735         return new UserVO(this.username, this.password);
736       }
737     }
738
739   12)com.example.service.UserService.java
740
741     package com.example.service;
742
743     import org.springframework.beans.factory.annotation.Autowired;
744     import org.springframework.stereotype.Service;
745
746     import com.example.dao.UserDao;
747     import com.example.vo.UserVO;
748
749     @Service("userService")
750     public class UserService {
751       private UserDao userDao;
752
753       public void setUserDao(UserDao userDao) {
754         this.userDao = userDao;
755       }
756
757       public UserVO getUsersByID(String username) {
758         return this.userDao.getUsersByID(username);
759       }
760     }
761
762   13)com.example.service.MemberService.java
763
764     package com.example.service;
765
766     import java.util.ArrayList;
767     import java.util.Collection;
768
769     import org.springframework.beans.factory.annotation.Autowired;
770     import org.springframework.security.core.authority.SimpleGrantedAuthority;
771     import org.springframework.security.core.userdetails.User;
772     import org.springframework.security.core.userdetails.UserDetails;
773     import org.springframework.security.core.userdetails.UserDetailsService;
774     import org.springframework.security.core.userdetails.UsernameNotFoundException;
775     import org.springframework.stereotype.Service;
776
```

```
777     import com.example.vo.UserVO;
778
779     @Service("memberService")
780     public class MemberService implements UserDetailsService{
781       private UserService userService;
782
783
784       /*
785        * 사용자가 입력한 ID/PWD 를 검증하기위해 저장소에 그 ID/PWD 가 있는지 확인하는 source이다. 저장소는
          in-memory가 될 수도있고,
786       * DB가 될 수도있고 어떠한 것(Facebook,Naver등등)도 될 수있다. userService는 DB 에 접근하기 위한
          service이다.
787        */
788
789       public void setUserService(UserService userService) {
790         this.userService = userService;
791       }
792
793       @Override
794       public UserDetails loadUserByUsername(String username) throws
          UsernameNotFoundException {
795         UserVO userVO = this.userService.getUsersByID(username);
796         if(userVO == null)
797           throw new UsernameNotFoundException("No user found with username " +
              userVO.getUsername());
798         Collection<SimpleGrantedAuthority> roles = new ArrayList<SimpleGrantedAuthority>();
799         roles.add(new SimpleGrantedAuthority("ROLE_USER"));
800         UserDetails user = new User(username, userVO.getPassword(), roles);
801         return user;
802       }
803     }
804
805   14)views/login.jsp
806
807     <%@ page language="java" contentType="text/html; charset=UTF-8"
808       pageEncoding="UTF-8"%>
809     <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
810     <!DOCTYPE html>
811     <html>
812     <head>
813       <title>로그인 페이지</title>
814       <meta charset="UTF-8">
815       <script>
816         function doLogin() {
817           if(frm.j_username.value == "") {
818             alert("아이디를 입력해주세요.");
819             return;
820           }
821           if(frm.j_password.value == "") {
822             alert("패스워드를 입력해주세요.");
823             return;
824           }
825           frm.submit();
826         }
```

```
827         </script>
828       </head>
829       <body>
830         <form name="frm" action="j_spring_security_check" method="post">
831           <ul>
832             <li>
833               <label for="userID">ID</label>
834               <input id = "userID" type="userID" name="j_username" placeholder="ID"
                  required>
835             </li>
836             <li>
837               <label for="password">Password</label>
838               <input id = "password" type="password" name="j_password"
                  placeholder="PASSWORD" required></li>
839             <li>
840               <input type="button" value="로그인" onclick="doLogin()"/>
841             </li>
842           </ul>
843           <input type="hidden" name="${_csrf.parameterName}"
844             value="${_csrf.token}" />
845         </form>
846
847         <c:if test="${not empty error}">
848           <div class="error">${error}</div>
849         </c:if>
850         <c:if test="${not empty msg}">
851           <div class="msg">${msg}</div>
852         </c:if>
853       </body>
854       </html>
855
856     15)views/welcome.jsp
857
858       <%@ page language="java" contentType="text/html; charset=UTF-8"
859         pageEncoding="UTF-8"%>
860       <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
861       <!DOCTYPE html>
862       <html>
863       <head>
864       <meta charset="UTF-8">
865       <title>Member Page</title>
866       </head>
867       <body>
868         <c:if test="${pageContext.request.userPrincipal.name != null}">
869           <h2>
870             Welcome : ${pageContext.request.userPrincipal.name} |
871             <a href="logout"> Logout</a>
872           </h2>
873         </c:if>
874       </body>
875       </html>
876
877     16)logout.jsp
878
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="username" value="${pageContext.request.userPrincipal.name}" />
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Logout page</title>
</head>
<body>
  <c:url value="/logout" var="logoutUrl" />
  <!-- csrt for log out-->
  <form action="${logoutUrl}" method="post" id="logoutForm">
    <input type="hidden" name="${_csrf.parameterName}"
     value="${_csrf.token}" />
  </form>
  <script>
    function formSubmit() {
      document.getElementById("logoutForm").submit();
    }
  </script>
  <form>
    <span style="color: gray;" ><h5> ${username} 님 반갑습니다.
    <a href = "javascript:formSubmit()"> 로그아웃 </a> </h5></span>
  </form>
</body>
</html>
```

17)Test
-SpringSecurityDemo1 > right-click > Run As > Run on Server
-http://localhost:8080/biz/login 으로 이동되며
-이때 ID와 Password를 잘 못 입력하면 Invalid username and password! 를 보며
-javaexpert/12345678을 입력하면 welcome으로 이동한다.
-welcome에서 Logout link click하면
-logout으로 이동하고 [로그아웃] link를 click하면 다시 첫 page로 이동하지만, You've been logged out successfully. message를 만난다.

7. 보안관련 taglibs 사용하기

1)SpringSecurityDemo project에서
2) pom.xml 에서 spring-security-taglibs : 4.2.12.RELEASE 확인
3)security/login.jsp에서
-아래 taglib 추가

```
<%@ taglib uri="http://www.springframework.org/security/tags" prefix="security" %>
```

-코드 변경

```
<c:if test="${not empty pageContext.request.userPrincipal}">
  <p>현재 로그인 상태</p>
</c:if>
<c:if test="${empty pageContext.request.userPrincipal}">
  <p>현재 로그아웃 상태</p>
</c:if>
```

-위의 코드를 아래의 코드로 변경

```
<security:authorize access="hasRole('ROLE_USER')">
  <p>현재 로그인 상태</p>
</security:authorize>
<security:authorize access="hasRole('ROLE_ANONYMOUS')">
  <p>현재 로그아웃 상태</p>
</security:authorize>
```

-코드 변경

```
User ID : ${pageContext.request.userPrincipal.name}<br />
```

-아래의 코드로 변경

```
User ID : <security:authentication property="name" /><br />
```

4)Test
-http://localhost:8080/biz/login.html 을 요청하면 http://localhost:8080/biz/loginForm.html 로 redirect
-아이디와 패스워드를 미리 설정한 javaexpert/12345678을 넣으면 정상적으로 login.jsp로 이동한다.
-이 때 화면에는 아래의 message가 나온다.

```
login.jsp
현재 로그인 상태
User ID : javaexpert
Log Out
```

-Logout link를 click하면 다시 첫 page로 이동한다.

8. Database-based Login Authentication
1)Spring Legacy Project 생성
-In Package Explorer > right-click > New > Spring Legacy Project
-Project name : SpringSecurityDatabaseDemo
-Select Spring MVC Project > Next
-com.example.biz > Finish

2)server.xml에 project 추가
-Tomcat v9.0 Server at localhost > right-click > Add and Remove
-SpringSecurityDatabaseDemo > Add > Finish > OK

3)Spring security library download and install
-In pom.xml 아래와 같이 수정

```
<java-version>1.8</java-version>
<org.springframework-version>4.3.24.RELEASE</org.springframework-version>
<org.aspectj-version>1.9.4</org.aspectj-version>
<org.slf4j-version>1.7.26</org.slf4j-version>
```

-version 수정

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
```

```
        <version>4.0.1</version>
        <scope>provided</scope>
      </dependency>
      <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>javax.servlet.jsp-api</artifactId>
        <version>2.3.3</version>
        <scope>provided</scope>
      </dependency>
      <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
      </dependency>

  -In mvnrepository, search 'spring security'
    -Select 'Spring Security Core' 4.2.12
    -Select 'Spring Security Web' 4.2.12
    -Select 'Spring Security Config' 4.2.12
    -Select 'Spring Security Taglibs' 4.2.12

    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-core</artifactId>
      <version>4.2.12.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-web</artifactId>
      <version>4.2.12.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-config</artifactId>
      <version>4.2.12.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-taglibs</artifactId>
      <version>4.2.12.RELEASE</version>
    </dependency>

  -pom.xml > right-click > Run As > Maven clean and Maven install
  -project > right-click > Properties > Project Facets > Java > Select 1.8
  -Runtimes Tab > Check Apache Tomcat v9.0
  -Click Apply and Close

4)HomeController.java

  package com.example.biz;

  import org.springframework.stereotype.Controller;
  import org.springframework.ui.Model;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class HomeController {
  @RequestMapping(value = "/", method = RequestMethod.GET)
  public String home(Model model) {
    model.addAttribute("greeting", "Hello Spring Security");
    return "home";
  }
}
```

5)home.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
  <title>Home</title>
</head>
<body>
<h1>${greeting} </h1>

</body>
</html>
```

6)project > right-click > Run As > Run on Server

7)Database table & sequence creation

```
CREATE SEQUENCE seq_user_role_id
MAXVALUE 9999;
/

CREATE  TABLE Users (
  username VARCHAR2(20),
  password VARCHAR2(20) NOT NULL,
  enabled NUMBER(1) DEFAULT 1 ,
  CONSTRAINT users_username_pk PRIMARY KEY (username)
);

CREATE TABLE User_roles (
  user_role_id  NUMBER(4),
  username VARCHAR2(45),
  role VARCHAR2(45) NOT NULL,
  CONSTRAINT user_roles_id PRIMARY KEY (user_role_id),
  CONSTRAINT user_roles_uk UNIQUE (role,username),
  CONSTRAINT user_roles_fk FOREIGN KEY (username) REFERENCES users (username)
);
```

8)Inserts some records for testing.

```
INSERT INTO Users(username,password,enabled) VALUES ('javaexpert','12345678', 1);
INSERT INTO users(username,password,enabled) VALUES ('alex','12345678', 1);
```

```
1093
1094      INSERT INTO user_roles (user_role_id, username, role)
1095      VALUES (seq_user_role_id.NEXTVAL, 'javaexpert', 'ROLE_USER');
1096
1097      INSERT INTO user_roles (user_role_id, username, role)
1098      VALUES (seq_user_role_id.NEXTVAL, 'javaexpert', 'ROLE_ADMIN');
1099
1100      COMMIT;
1101
1102    9)pom.xml에 oracle driver와 MyBatis 관련 dependency 추가
1103
1104      <dependency>
1105        <groupId>com.oracle</groupId>
1106        <artifactId>ojdbc8</artifactId>
1107        <version>12.2</version>
1108      </dependency>
1109      <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
1110      <dependency>
1111        <groupId>org.mybatis</groupId>
1112        <artifactId>mybatis</artifactId>
1113        <version>3.5.1</version>
1114      </dependency>
1115      <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
1116      <dependency>
1117        <groupId>org.mybatis</groupId>
1118        <artifactId>mybatis-spring</artifactId>
1119        <version>2.0.1</version>
1120      </dependency>
1121
1122    10)Database 연결을 위한 환경설정
1123      -Build path에 config foler 추가
1124        --project right-click > Build Path > Configure Build Path > Select [Source] tab
1125        --Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
1126        --Folder name : config > Finish > OK > Apply and Close
1127        --Java Resources > config folder 확인
1128
1129      -config/dbinfo.properties file 생성
1130
1131        db.driver=oracle.jdbc.driver.OracleDriver
1132        db.url=jdbc:oracle:thin:@localhost:1521:XE
1133        db.username=hr
1134        db.password=hr
1135
1136      -pom.xml에 spring-jdbc 추가
1137
1138        <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
1139        <dependency>
1140          <groupId>org.springframework</groupId>
1141          <artifactId>spring-jdbc</artifactId>
1142          <version>4.3.24.RELEASE</version>
1143        </dependency>
1144
1145      -web.xml 코드 추가
1146
```

```
    <filter>
      <filter-name>springSecurityFilterChain</filter-name>
      <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    </filter>

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

-web.xml 수정

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

-src/main/webapp/WEB-INF/spring/spring-security.xml 생성
  --src/main/webapp/WEB-INF/spring > right-click > New > Spring Bean Configuration File
  --File name : spring-security.xml > Next
  --Check beans, security, context > Finish

    <?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"
      xmlns:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/security
      http://www.springframework.org/schema/security/spring-security-4.2.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

      <context:property-placeholder location="classpath:dbinfo.properties"/>
      <bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName"
          value="${db.driverClass}" />
        <property name="url"
          value="${db.url}" />
        <property name="username" value="${db.username}" />
        <property name="password" value="${db.password}" />
      </bean>

      <!-- enable use-expressions -->
      <security:http>

        <security:intercept-url pattern="/admin**" access="hasRole('ROLE_ADMIN')" />

        <!-- access denied page -->
        <security:access-denied-handler error-page="/403" />

        <security:form-login
          login-page="/login"
          default-target-url="/welcome"
```

```
                authentication-failure-url="/login?error"
                login-processing-url="/j_spring_security_check"
                username-parameter="username"
                password-parameter="password" />
              <security:logout invalidate-session="true"
                logout-url="/j_spring_security_logout" logout-success-url="/login?logout" />
              <!-- enable csrf protection -->
              <security:csrf   disabled="true"/>
            </security:http>

            <!-- Select users and user_roles from database -->
            <security:authentication-manager>
              <security:authentication-provider>
               <security:jdbc-user-service data-source-ref="dataSource"
                 users-by-username-query=
                  "select username,password, enabled from users where username=?"
                 authorities-by-username-query=
                  "select username, role from user_roles where username =?  " />
              </security:authentication-provider>
            </security:authentication-manager>
          </beans>

    11)views/hello.jsp

      <%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
      <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
      <%@taglib prefix="sec"
        uri="http://www.springframework.org/security/tags"%>
      <html>
      <body>
        <h1>Title : ${title}</h1>
        <h1>Message : ${message}</h1>

        <sec:authorize access="hasRole('ROLE_USER')">
          <!-- For login user -->
          <c:url value="/j_spring_security_logout" var="logoutUrl" />
          <form action="${logoutUrl}" method="post" id="logoutForm">
            <input type="hidden" name="${_csrf.parameterName}"
              value="${_csrf.token}" />
          </form>
          <script>
            function formSubmit() {
              document.getElementById("logoutForm").submit();
            }
          </script>

          <c:if test="${pageContext.request.userPrincipal.name != null}">
            <h2>
              User : <sec:authentication property="name" /> | <a
                href="javascript:formSubmit()"> Logout</a>
            </h2>
          </c:if>

```

```
1253
1254        </sec:authorize>
1255      </body>
1256      </html>
1257
1258    12)views/login.jsp
1259
1260      <%@ page language="java" contentType="text/html; charset=UTF-8"
1261        pageEncoding="UF-8"%>
1262        <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
1263      <html>
1264      <head>
1265      <title>Login Page</title>
1266      <style>
1267        .error {
1268          padding: 15px;
1269          margin-bottom: 20px;
1270          border: 1px solid transparent;
1271          border-radius: 4px;
1272          color: #a94442;
1273          background-color: #f2dede;
1274          border-color: #ebccd1;
1275        }
1276
1277        .msg {
1278          padding: 15px;
1279          margin-bottom: 20px;
1280          border: 1px solid transparent;
1281          border-radius: 4px;
1282          color: #31708f;
1283          background-color: #d9edf7;
1284          border-color: #bce8f1;
1285        }
1286
1287        #login-box {
1288          width: 300px;
1289          padding: 20px;
1290          margin: 100px auto;
1291          background: #fff;
1292          -webkit-border-radius: 2px;
1293          -moz-border-radius: 2px;
1294          border: 1px solid #000;
1295        }
1296      </style>
1297      </head>
1298      <body onload='document.loginForm.username.focus();'>
1299
1300        <h1>Spring Security Login Form (Database Authentication)</h1>
1301
1302        <div id="login-box">
1303
1304          <h2>Login with Username and Password</h2>
1305
1306          <c:if test="${not empty error}">
```

```
          <div class="error">${error}</div>
        </c:if>
        <c:if test="${not empty msg}">
          <div class="msg">${msg}</div>
        </c:if>

        <form name='loginForm'
         action="<c:url value='/j_spring_security_check' />" method='POST'>

        <table>
          <tr>
            <td>User:</td>
            <td><input type='text' name='username'></td>
          </tr>
          <tr>
            <td>Password:</td>
            <td><input type='password' name='password' /></td>
          </tr>
          <tr>
            <td colspan='2'><input name="submit" type="submit"
              value="submit" /></td>
          </tr>
         </table>

         <input type="hidden" name="${_csrf.parameterName}"
          value="${_csrf.token}" />

        </form>
      </div>
    </body>
    </html>

13)views/admin.jsp

    <%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
    <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
    <html>
    <body>
      <h1>Title : ${title}</h1>
      <h1>Message : ${message}</h1>

      <c:url value="/j_spring_security_logout" var="logoutUrl" />
      <form action="${logoutUrl}" method="post" id="logoutForm">
        <input type="hidden" name="${_csrf.parameterName}"
          value="${_csrf.token}" />
      </form>
      <script>
        function formSubmit() {
          document.getElementById("logoutForm").submit();
        }
      </script>

      <c:if test="${pageContext.request.userPrincipal.name != null}">
```

```
        <h2>
          Welcome : ${pageContext.request.userPrincipal.name} | <a
            href="javascript:formSubmit()"> Logout</a>
        </h2>
      </c:if>

  </body>
  </html>

14)views/403.jsp

  <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
  <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
  <html>
  <body>
    <h1>HTTP Status 403 - Access is denied</h1>

    <c:choose>
      <c:when test="${empty username}">
        <h2>You do not have permission to access this page!</h2>
      </c:when>
      <c:otherwise>
        <h2>Username : ${username} <br/>
            You do not have permission to access this page!</h2>
      </c:otherwise>
    </c:choose>

  </body>
  </html>

15)HomeController.java

  @RequestMapping(value = { "/", "/welcome**" }, method = RequestMethod.GET)
  public ModelAndView defaultPage() {

    ModelAndView model = new ModelAndView();
    model.addObject("title", "Spring Security Login Form - Database Authentication");
    model.addObject("message", "This is default page!");
    model.setViewName("hello");
    return model;

  }

  @RequestMapping(value = "/admin**", method = RequestMethod.GET)
  public ModelAndView adminPage() {

    ModelAndView model = new ModelAndView();
    model.addObject("title", "Spring Security Login Form - Database Authentication");
    model.addObject("message", "This page is for ROLE_ADMIN only!");
    model.setViewName("admin");
    return model;

  }
```

```
1415
1416      @RequestMapping(value = "/login", method = RequestMethod.GET)
1417      public ModelAndView login(@RequestParam(value = "error", required = false) String error,
1418        @RequestParam(value = "logout", required = false) String logout) {
1419
1420        ModelAndView model = new ModelAndView();
1421        if (error != null) {
1422         model.addObject("error", "Invalid username and password!");
1423        }
1424
1425        if (logout != null) {
1426         model.addObject("msg", "You've been logged out successfully.");
1427        }
1428        model.setViewName("login");
1429
1430        return model;
1431
1432      }
1433
1434      //for 403 access denied page
1435      @RequestMapping(value = "/403", method = RequestMethod.GET)
1436      public ModelAndView accesssDenied() {
1437
1438        ModelAndView model = new ModelAndView();
1439
1440        //check if user is login
1441        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
1442        if (!(auth instanceof AnonymousAuthenticationToken)) {
1443         UserDetails userDetail = (UserDetails) auth.getPrincipal();
1444         model.addObject("username", userDetail.getUsername());
1445        }
1446
1447        model.setViewName("403");
1448        return model;
1449
1450      }
1451      @RequestMapping(value = "/logout", method = RequestMethod.GET)
1452      public String logout(Model model) {
1453        return "logout";
1454      }
1455
1456    16)Test
1457      -javaexpert/12345678은 admin과 일반 login모두 성공
1458      -하지만 alex/12345678은 login 실패, 왜냐하면 role이 없기 때문에
1459      -login하려면 아래 query처럼 user_roles에 등록해야 ROLE_USER 권한을 부여 받음.
1460        INSERT INTO user_roles (user_role_id, username, role)
1461        VALUES (seq_user_role_id.NEXTVAL, 'alex', 'ROLE_USER');
1462      -그렇다면 alex가 admin page에 접근할 수 있을까?
1463      -현재는 403 error page로 이동한다. 왜냐하면 그는 ROLE_USER 권한만 있기 때문이다.
1464        HTTP Status 403 - Access is denied
1465        Username : alex
1466        You do not have permission to access this page!
```