

Spring Security

INDICE

INDICE	2
1.- INTRODUCCIÓN A SPRING SECURITY	3
2.- CONFIGURACION DEL PROYECTO.....	5
DelegatingFilterProxy	6
Spring-security.xml.....	7
3.- AUTENTICACION CON FORMULARIO PERSONALIZADO	9
4.- REMEMBERME	11
5.- LOGOUT	13
6.- OTRAS CONFIGURACIONES	14
HTTPS	14
CONTROL UNICA SESION	14
CIFRADO DE PASSWORD	15
7.- EXPRESSIONES SPRING SECURITY.....	16
isAuthenticated	16
isAnonymous.....	16
isRememberMe	16
hasRole.....	16
hasAnyRole	17
hasIpAddress	17
sFullyAuthenticated.....	17
denyAll	17
permitAll.....	18
principal	18
authentication	18
isAuthenticated	18
8.- ETIQUETAS SPRING SECURITY	20
9.- AUTORIZACION EN LOS METODOS.....	21
anotacion @Secured	21
anotacion @RolesAllowed.....	21
10.- AUTORIZACION CON LDAP	23
Concepto LDAP	23
LDAP en memoria.....	24
Acceso a un LDAP remoto	26
11.- AUTORIZACION CONTRA BBDD	27
12.- SPRING SECURITY CON AOP	29
INDICE DE GRÁFICOS	30

1.- INTRODUCCIÓN A SPRING SECURITY

Spring Security proporciona servicios de seguridad para aplicaciones de software empresariales basados en JEE, enfocado particularmente sobre proyectos contruidos usando SpringFramework.

Como sabemos la seguridad comprende dos operaciones: La primera operación es conocida como "autenticación", por el cual se establece si un usuario(que quiere realizar una acción en nuestra aplicación) es quien dice ser, y la segunda operación es llamada "autorización" que se refiere al proceso de decidir si a un usuario le es permitido realizar una determinada acción en nuestra aplicación.

Para llegar al punto donde una acción de autorización es necesaria, la identidad del usuario ya ha sido establecida por el proceso de "autenticación", estos conceptos son comunes y no todos son especificos a Spring Security.

En el nivel de "autenticación" Spring Security soporta muchos modelos de autenticación, muchos de estos modelos de autenticación son proporcionados por terceros o son desarrollados por estandares importantes como el IETF(Internet Engineering tTask Force), adicionalmente, Spring Security proporciona su propio conjunto de características de autenticación.

Especificamente, Spring Security actualmente soporta integración de autenticación con todas las siguientes tecnologías:

1. HTTP BASIC authentication headers (an IEFT RFC-based standard).
2. HTTP Digest authentication headers (an IEFT RFC-based standard).
3. HTTP X.509 client certificate exchange (an IEFT RFC-based standard).
4. LDAP (un enfoque muy comun para necesidades de autenticación multiplataforma, especificamente en entornos extensos).
5. Form-based authentication (necesario para interfaces de usuario simples).
6. OpenID authentication.
7. Computer Associates Siteminder.
8. JA-SIG Central Authentication Service.
9. Transparent authentication context propagation for Remote Method Invocation (RMI) and HttpInvoker.
10. Automatic "remember-me" authentication.
11. Anonymous authentication.
12. Run-as authentication.
13. Java Authentication and Authorization Service (JAAS)

- 14.Container integration with JBoss, Jetty, Resin and Tomcat (tambien podemos usar autenticación gestionada por el contenedor)
- 15.Java Open Source Single Sign On (JOSSO) *
- 16.OpenNMS Network Management Platform *
- 17.AppFuse *
- 18.AndroMDA *
- 19.Mule ESB *
- 20.Direct Web Request (DWR) *
- 21.Grails *
- 22.Tapestry *
- 23.JTrac *
- 24.Jasypt *
- 25.Roller *
- 26.Elastic Plath *
- 27.Atlassian Crowd *
- 28.Nuestros propios sistemas de autenticación.

(* Indica proporcionado por un tercero)

2.- CONFIGURACION DEL PROYECTO

Si ya tenemos la aplicación construida deberemos configurarla para que quede protegida por Spring Security .

Para ello el primer paso que debemos realizar es dar de alta en el fichero web.xml la ruta en donde tenemos ubicado el fichero de configuración de Spring (usando un context param) .

El siguiente paso es declarar un listener que nos inicialice el framework y por último un filtro que protega toda la aplicación de accesos no permitidos y delegue en Spring Framework todas las operativas de seguridad. Así pues el web.xml tendra el siguiente contenido.

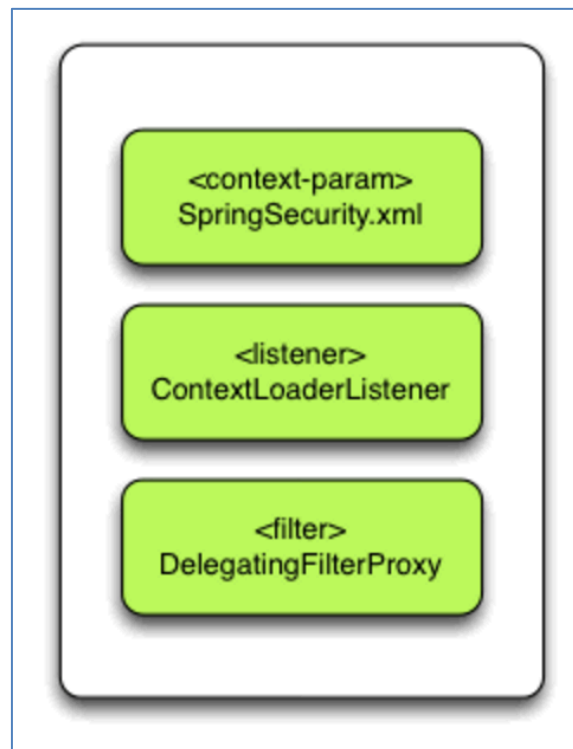


Gráfico 1. Componentes de web.xml

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

<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

Gráfico 2. Cargar el contexto spring-security.xml

DELEGATINGFILTERPROXY

Una vez tenemos configurado el fichero web.xml ,el filtro de SpringSecurity se encargará de bloquear el acceso a toda la aplicación.

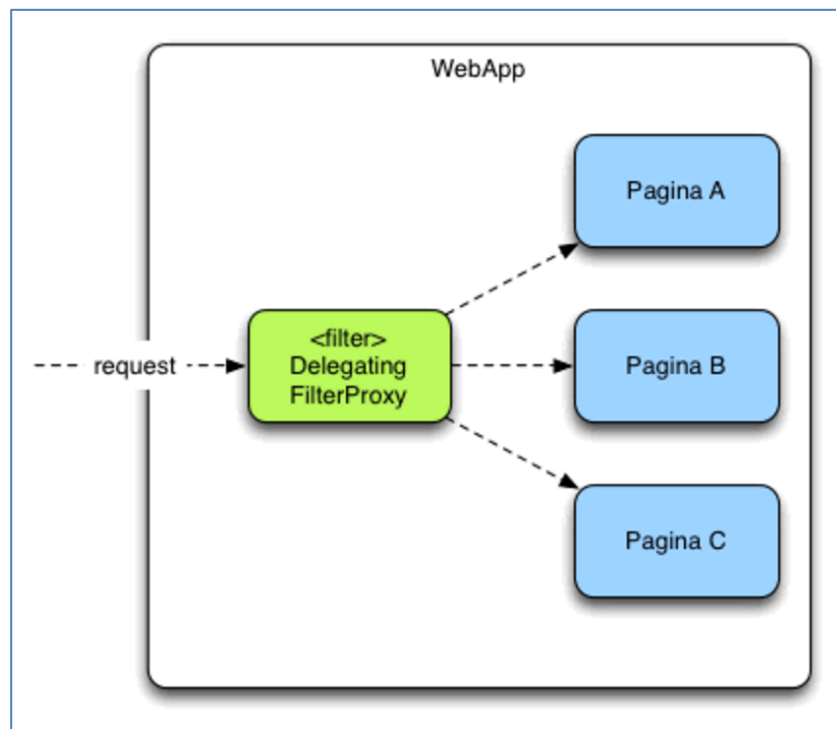


Gráfico 3. Bloqueo con filtro DelegatingFilterProxy

```

<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>

```

Gráfico 4. Declaración de filtro DelegatingFilterProxy en el web.xml

SPRING-SECURITY.XML

Acabamos de configurar el framework Spring ,es momento de ver el contenido del fichero springsecurity.xml.

Este fichero es al cual el filtro de Spring delega para gestionar la seguridad.

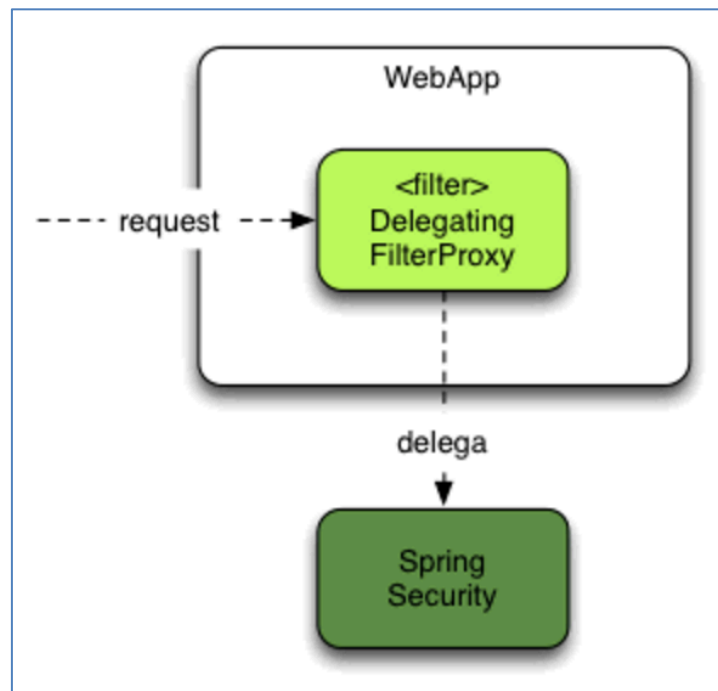


Gráfico 5. El filtro delega en spring-security.xml

```

<http auto-config="true">
  <!-- Todas ls peticiones requieren de autenticacion
        Solo son los usuarios con role ROLE_USER tienen acceso -->
  <intercept-url pattern="/**" />

  <intercept-url pattern="/todos" access="ROLE_USER"/>
  <intercept-url pattern="/buscar" access="ROLE_ADMIN"/>
</http>

<authentication-manager>
  <authentication-provider>
    <user-service>
      <user name="juan" password="pw123" authorities="ROLE_USER"/>
      <user name="maria" password="pw456" authorities="ROLE_ADMIN"/>
    </user-service>
  </authentication-provider>
</authentication-manager>

```

Gráfico 6. Contenido de spring-security.xml

Con las etiquetas <intercept-url> estamos indicando que todas las peticiones requieren de autenticación.


Los usuarios con el ROLE_USER tienen autorización para la petición /todos y los usuarios con ROLE_ADMIN tienen autorización para la petición /buscar.

Mediante <authentication-manager> hemos definido los usuarios en memoria. No es la mejor practica pero a lo largo de este capitulo veremos otras posibilidades para almacenar los usuarios declarados.

Todo el código de este ejemplo lo encontrareis en **Ejemplo1_Spring_Security.zip**

3.- AUTENTICACION CON FORMULARIO PERSONALIZADO

En el ejemplo anterior, nos aparecía el formulario que vemos en la siguiente imagen para logarnos.

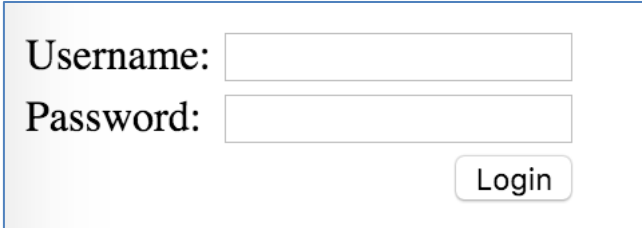


The image shows a default login form with a light gray background and a blue border. At the top, it has the title "Login with Username and Password" in bold black text. Below the title, there are two input fields: "User:" followed by a text box, and "Password:" followed by a text box. At the bottom left, there is a "Login" button with rounded corners and a light gray background.

Gráfico 7. Formulario login por defecto

Pero también tenemos la posibilidad de crear nuestro propio formulario y que Spring Security lo utilice para autenticar a los usuarios.

En la siguiente imagen vemos el formulario que nos vamos a construir.



The image shows a custom login form with a light gray background and a blue border. It has two input fields: "Username:" followed by a text box, and "Password:" followed by a text box. At the bottom right, there is a "Login" button with rounded corners and a light gray background.

Gráfico 8. Formulario login personalizado

Es muy importante que en el action conservemos el valor `j_spring_security_check` para que todo siga funcionando correctamente.

Los parámetros deben ser `j_username` y `j_password` como nombres por defecto pero esta vez existe la posibilidad de cambiarlos.

La pagina `login.jsp` recoge el formulario que vemos en la imagen.

```

<form action="j_spring_security_check" method="post">
  <table>
    <tr>
      <td>Username:</td>
      <td><input type="text" name="user" /></td>
    </tr>
    <tr>
      <td>Password:</td>
      <td><input type="password" name="password" /></td>
    </tr>
    <tr>
      <td colspan="2" align="right" >
        <input type="submit" value="Login" />
      </td>
    </tr>
  </table>
</form>

<font color="red">
  <span> ${sessionScope["SPRING_SECURITY_LAST_EXCEPTION"].message} </span>
</font>

```

Gráfico 9. login.jsp

Ahora debemos configurar en el archivo spring-security.xml cual es la pagina que contiene el formulario de login.

```

<!-- Desactivamos la seguridad en /login porque sino entramos en un bucle -->
<http pattern="/login" security="none" />

<http use-expressions="true">
  <!-- Todas ls peticiones requieren e autenticacion
       Solo son los usuarios con role ROLE_USER tienen acceso -->
  <intercept-url pattern="/**" access="isAuthenticated()"/>

  <!-- Los parametros por defecto son j_username y j_password pero
       si los queremos cambiar hay que informar a Spring Security -->
  <form-login login-page="/login"
              default-target-url="/index"
              authentication-failure-url="/login"
              username-parameter="user"
              password-parameter="password" />

</http>

```

Gráfico 10. Configuración form-login

Todo el código de este ejemplo lo encontrareis en **Ejemplo2_Spring_Security_Login.zip**

4.- REMEMBERME

En este capítulo mostraremos cómo activar y configurar la funcionalidad en una aplicación web remember-me con Spring Security.

Este mecanismo será capaz de identificar al usuario a través de múltiples sesiones - por lo que la primera cosa a tener en cuenta es que se Remember me sólo se activa después del timeout de la sesión . De forma predeterminada, esto sucede después de 30 minutos de inactividad, pero el tiempo de espera se puede configurar en el *web.xml*.

Lo que haremos será incluir el checkbox Remember me en el formulario.

```
<form action="j_spring_security_check" method="post">
  <table>
    <tr>
      <td>Username:</td>
      <td><input type="text" name="user" /></td>
    </tr>
    <tr>
      <td>Password:</td>
      <td><input type="password" name="password" /></td>
    </tr>
    <tr>
      <td></td>
      <td>
        Remember me:
        <input type="checkbox" name="_spring_security_remember_me" />
      </td>
    </tr>
    <tr>
      <td colspan="2" align="right">
        <input type="submit" value="Login" />
      </td>
    </tr>
  </table>
</form>
```

Gráfico 11. checkbox Remember me

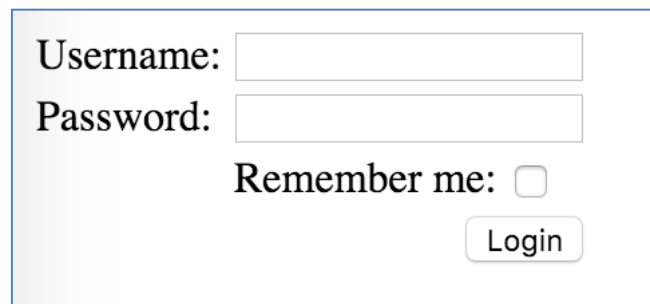
La clave aquí es importante - es un secreto privado valor para toda la aplicación y se va a utilizar cuando se genera el token.

Luego declararemos la nueva funcionalidad en el archivo spring-security.xml

```
<!-- El token almacena el usuario, password, tiempo de expiracion y  
la clave privada, el atributo key es la clave privada.  
La expiracion se establece en segundos -->  
<remember-me  
    token-validity-seconds="1209600"  
    key="userKey"/>
```

Gráfico 12. Elemento remember-me

El resultado será el siguiente:



A login form with a light gray background. It contains the following elements: a 'Username:' label followed by a text input field; a 'Password:' label followed by a text input field; a 'Remember me:' label followed by an unchecked checkbox; and a 'Login' button located below the checkbox.

Gráfico 13. Formulario con Remember me

Todo el código de este ejemplo lo encontrareis en **Ejemplo3_Spring_Security_RememberMe.zip**

5.- LOGOUT

Si queremos cerrar la sesión podemos hacerlo de forma programática.

En la pagina jsp vamos a incluir un enlace con el path logout.

```
<table align="right">
  <tr>
    <td>
      <a href="logout">Cerrar sesion</a>
    </td>
  </tr>
</table>
```

Gráfico 14. Link para logout

En el archivo spring-security.xml configuramos el logout para que al detectar la petición redirige a la pagina index y por supuesto cierra la sesión.

```
<!-- Con la petición /logout cerramos la sesión y redirigimos
a la página index -->
<logout logout-url="/logout"
logout-success-url="/index" />
```

Gráfico 15. Configuración logout

Todo el código de este ejemplo lo encontrareis en **Ejemplo4_Spring_Security_Logout.zip**

6.- OTRAS CONFIGURACIONES

HTTPS

Podemos interceptar una url para que se ejecute bajo el protocolo https.

```
<!-- Interceptamos la petición de alta para que se redirija como https -->  
<intercept-url pattern="/alta" requires-channel="https"/>
```

Gráfico 16. Redirección a https

CONTROL UNICA SESION

También podemos controlar que haya una sola sesión por usuario y si hay mas de uno lanzamos un error.

```
<!-- Control de sesiones, detectar que haya una sola sesion  
por usuario -->  
<session-management>  
  <concurrency-control max-sessions="1"  
    error-if-maximum-exceeded="true"/>  
</session-management>
```

Gráfico 17. Control de única sesión

Necesitamos un listener para poder escuchar el número de sesiones abiertas.

```
<listener>  
  <listener-class>  
    org.springframework.security.web.session.HttpSessionEventPublisher  
  </listener-class>  
</listener>
```

Gráfico 18. Listener para detectar nuevas sesiones

CIFRADO DE PASSWORD

Podemos encriptar las password utilizando diferentes algoritmos de cifrado como md5, sha, ...etc.

```
<authentication-manager>
  <authentication-provider>
    <user-service>
      <user name="juan" password="8fc83302c44fcb68b793ceca1d376996" authorities="ROLE_USER" />
      <user name="maria" password="pw456" authorities="ROLE_ADMIN" />
    </user-service>

    <!-- Configurar para que las contraseñas se encripten con md5,
         otros valores pueden ser:
         sha, ssh, md4, md5, plaintext, sha-256 -->
    <password-encoder hash="md5" />

  </authentication-provider>
</authentication-manager>
```

Gráfico 19. Encriptar password

Todo el código de este ejemplo lo encontrareis en **Ejemplo5_Spring_Security_Https.zip**

7.- EXPRESIONES SPRING SECURITY

Tenemos a nuestra disposición una serie de expresiones que nos pueden ayudar para definir accesos.

ISAUTHENTICATED

Devuelve true si el usuario esta autenticado

```
<intercept-url pattern="/"**" access="isAuthenticated()"/>
```

Gráfico 20. isAuthenticated

ISANONYMOUS

Devuelve true si el usuario no esta autenticado, es anónimo

```
<intercept-url pattern="/"**" access="isAnonymous()"/>
```

Gráfico 21. isAnonymous

ISREMEMBERME

Devuelve true si el usuario es recordado

```
<intercept-url pattern="/"**" access="isRememberMe()"/>
```

Gráfico 22. isRememberMe

HASROLE

Devuelve true si el usuario tiene el role de ROLE_USER


```
<intercept-url pattern="/"**" access="hasRole('ROLE_USER')"/>
```

Gráfico 23. hasRole

HASANYROLE

Devuelve true si el usuario tiene el role de ROLE_USER o ROLE_ADMIN

```
<intercept-url pattern="/"**" access="hasAnyRole(['ROLE_USER', 'ROLE_ADMIN'])"/>
```

Gráfico 24. hasAnyRole

HASIPADDRESS

Devuelve true si el usuario se ha logado desde esa ip

```
<intercept-url pattern="/"**" access="hasIpAddress('192.168.1.2')"/>
```

Gráfico 25. hasIpAddress

SFULLYAUTHENTICATED

Devuelve true cuando el usuario ni es anonimo ni recordado

```
<intercept-url pattern="/"**" access="isFullyAuthenticated()" />
```

Gráfico 26. isFullyAuthenticated

DENYALL

Siempre devuelve false, se utiliza para prohibir una ruta

```
<intercept-url pattern="/"**" access="denyAll()" />
```

Gráfico 27. denyAll

PERMITALL

Siempre devuelve true, permitido a todos

```
<intercept-url pattern="/" access="permitAll()" />
```

Gráfico 28. permitAll

PRINCIPAL

Acceso al objeto principal

Ejemplo: principal.username == 'juan'

```
<intercept-url pattern="/" access="principal"/>
```

Gráfico 29. principal

AUTHENTICATION

Acceso al objeto de autentificación generado por SecurityContext

```
<intercept-url pattern="/" access="authentication"/>
```

Gráfico 30. authentication

ISAUTHENTICATED

Todas las peticiones requieren de autenticación

```
<intercept-url pattern="/" access="isAuthenticated()" />
```

Gráfico 31. isAuthenticated

Todo el código de este ejemplo lo encontrareis en
Ejemplo6_Spring_Security_EtiquetasVista.zip

8.- ETIQUETAS SPRING SECURITY

Si queremos utilizar las etiquetas de Spring Security en nuestras paginas debemos agregar la librería tal como se muestra en la siguiente imagen.

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

Gráfico 32. Librería de etiquetas

A través de la etiqueta sec:authorize podemos marcar secciones que solo se mostraran a un perfil de usuarios.

En la siguiente imagen vemos un ejemplo: Esta seccion solo la veran los usuarios logados co role ROLE_USER

```
<sec:authorize access="hasRole('ROLE_USER')">
```

Gráfico 33. Marcar secciones

Con la etiqueta sec:authentication podemos recuperar datos almacenados en el proceso de autenticación.

```
Hola <sec:authentication property="principal.username"/>
```

Gráfico 34. Obtener el username

Aparte del principal se puede utilizar:

- authorities; privilegios otorgados al usuario
- credentials; suele ser el password
- details; info adicional como por ejemplo el id de la session
- principal; el objeto creado al logarse, el usuario

Todo el código de este ejemplo lo encontrareis en **Ejemplo6_Spring_Security_EtiquetasVista.zip**

9.- AUTORIZACION EN LOS METODOS

Una vez que ya sabemos como autenticar usuarios en nuestra aplicación vamos a ver como podemos configurar los métodos seguros para que solo determinados roles puedan ejecutarlos.

ANOTACION @SECURED

Habilitamos la autorizacion con la anotacion @Secured

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

Gráfico 35. Habilitar anotación @Secured

```
@Secured({"ROLE_USER"})
public void metodo1() {
    System.out.println("Ejecutando metodo 1 los usuarios con ROLE_USER");
}

@Secured({"ROLE_USER", "ROLE_ADMIN"})
public void metodo2() {
    System.out.println("Ejecutando metodo 2 los usuarios "
        + "con ROLE_USER o ROLE_ADMIN");
}
```

Gráfico 36. Métodos anotados con @Secured

ANOTACION @ROLESALLOWED

Habilitamos la autorizacion con la anotacion @RolesAllowed

```
<global-method-security jsr250-annotations="enabled" />
```

Gráfico 37. Habilitar anotación @RolesAllowed

```
@RolesAllowed("ROLE_USER")
public void metodo3() {
    System.out.println("Ejecutando metodo 3 los usuarios con ROLE_USER");
}

@RolesAllowed({"ROLE_USER", "ROLE_ADMIN"})
public void metodo4() {
    System.out.println("Ejecutando metodo 4 los usuarios "
        + "con ROLE_USER o ROLE_ADMIN");
}
```

Gráfico 38. Métodos anotados con @RolesAllowed

Todo el código de este ejemplo lo encontrareis en **Ejemplo7_Spring_Security_Autorizacion_Metodos.zip**

10.- AUTORIZACION CON LDAP

CONCEPTO LDAP

LDAP es un protocolo ligero de acceso a directorios:

L LIGHTWEIGHT
D DIRECTORY
A ACCESS
P PROTOCOL

Que es la versión ligera del protocolo, del que deriva su nombre, pero mas robusto, que se llama DAP.

Sirve para dar y compartir datos de individuos, usuarios de sistemas, dispositivos de redes y sistemas, sobre redes ya existentes para programas (clientes) de correo electrónico y aplicaciones que requieren autenticación (autorización de acceso para usar de manera común la información que exista en el servidor de la red).

Sirve para acceder a varias bases de datos, de manera concurrente o no, en donde se guardan datos sensibles, y se usa LDAP para evitar muchos nombres de usuario y contraseñas de un mismo usuario, que necesite acceder a muchas bases de datos a la vez.

Ejemplo: Un usuario, tiene su cuenta de correo electrónico, dentro del servidor de correos de la empresa xyz, además tiene un directorio o carpeta destinado a el, dentro del servidor de archivos de la empresa xyz en donde guarda documentos creados por el, además, tiene una cuenta de ftp, para acceder de manera remota a sus documentos, además le ha creado el administrador de la red de la empresa xyz, una cuenta para acceder al servidor de proyectos de su empresa, y el administrador a usado LDAP para que use una sola cuenta de usuario y una sola contraseña universal para acceder a estos 4 servicios, en 4 diferentes servidores, en donde todos los servidores tienen su base de datos independiente, unidos todos por LDAP.

LDAP EN MEMORIA

Podemos declarar los usuarios en memoria a través del archivo `users.ldif` y luego configurar el root para efectuar las búsquedas.

Es importante establecer un id en la configuración para luego poder hacer referencia en el authentication manager.

```
<ldap-server id="serverLDAP"
            root="dc=habuma,dc=com" ldif="classpath:users.ldif" />
```

Gráfico 39. Configuración del servidor LDAP en memoria

Posteriormente, como ya indicábamos, creamos el authentication provider haciendo referencia al servidor local.

```
<authentication-manager>
  <authentication-provider user-service-ref="serverLDAP" />
</authentication-manager>
```

Gráfico 40. Authentication provider para el servidor local

El archivo `users.ldif` es el que se muestra a continuación:


```
dn: ou=groups,dc=habuma,dc=com
objectclass: top
objectclass: organizationalUnit
ou: groups

dn: ou=people,dc=habuma,dc=com
objectclass: top
objectclass: organizationalUnit
ou: people

dn: uid=habuma,ou=people,dc=habuma,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Craig Walls
sn: Walls
uid: habuma
userPassword: password

dn: uid=jsmith,ou=people,dc=habuma,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: John Smith
sn: Smith
uid: jsmith
userPassword: password

dn: cn=spitter,ou=groups,dc=habuma,dc=com
objectclass: top
objectclass: groupOfNames
cn: spitter
member: uid=habuma,ou=people,dc=habuma,dc=com
```

Gráfico 41. Archivo users.ldif

ACCESO A UN LDAP REMOTO

Otra forma de trabajar es a través de un servidor LDAP remoto que configuraremos el acceso a través de la etiqueta `<ldap-user-service>`

```
<ldap-user-service id="ldapService"
    user-search-base="ou=people"
    user-search-filter="(uid={0})"
    group-search-base="ou=groups"
    group-search-filter="member={0}"
    server-ref="serverLDAP"/>
```

Gráfico 42. Servicio para LDAP remoto

Y como proveedor de autenticación hacemos referencia a dicho servicio:

```
<authentication-manager>
    <authentication-provider user-service-ref="ldapService" />
</authentication-manager>
```

Gráfico 43. Authentication provider para el servidor local

En esta caso Spring Security asume que el puerto LDAP por defecto es el 33389.

También tenemos la opción de configurar la url del servidor remoto y como vemos en la siguiente imagen estamos accediendo a través de otro puerto.

```
<ldap-server url="ldap://habuma.com:389/dc=habuma,dc=com" />
```

Gráfico 44. Acceso al servidor remoto a través de URL

Todo el código de este ejemplo lo encontrareis en **Ejemplo8_Spring_Security_LDAP.zip**

11.- AUTORIZACION CONTRA BBDD

Otra forma de validar usuarios es utilizando una base de datos común.

En este caso debemos declarar las queries que se lanzarán para comprobar si un usuario existe y si tiene autorización.

En la siguiente imagen definimos las siguientes consultas:

- **Users-by-username-query**; obtiene las credenciales del usuario introducido en el formulario para verificar que este existe en la base de datos.
- **Authorities-by-username-query**; Obtiene las autorizaciones de ese usuario.
- **Group-authorities-by-username-query**; Devuelve las autorizaciones del grupo al que pertenece el usuario.

```
<jdbc-user-service id="jdbcService"
                  data-source-ref="dataSource"
    users-by-username-query="select username,password,enabled
                             from users where username = ?"
    authorities-by-username-query="select username, authority
                                   from authorities where username = ?"
    group-authorities-by-username-query="select g.id,
g.group_name, ga.authority from groups g,
group_members gm, group_authorities ga where
gm.username = ? and g.id = ga.group_id and g.id =
gm.group_id"
/>
```

Gráfico 45. Definición de las Queries

Indicamos al authentication-manager quien es ahora el proveedor de autenticación.

```
<authentication-manager>
  <authentication-provider user-service-ref="jdbcService" />
</authentication-manager>
```

Gráfico 46. Proveedor de autenticacion

Debe existir un bean data-source con los datos de conexión.

```
<!-- Crear un bean con los datos de conexion e la BBDD -->
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name="driverClassName" value="com.mysql.jdbc.Driver" />
  <property name="url" value="http:mysql://localhost:3306/bbdd" />
  <property name="username" value="root" />
  <property name="password" value="root" />
</bean>
```

Gráfico 47. Data source

Todo el código de este ejemplo lo encontrareis en **Ejemplo9_Spring_Security_JDBC.zip**

12.- SPRING SECURITY CON AOP

Podemos utilizar AOP para declarar que perfil de usuarios tienen autorización para ejecutar determinados métodos.

En la siguiente imagen vemos como hacerlo a través del punto de corte.

```
<global-method-security>
  <protect-pointcut access="ROLE_USER"
    expression="execution(* app.persistencia.ProductosDAO.consultarId(..))"/>
</global-method-security>
```

Gráfico 48. Configuración con AOP

Es importante recordar que en el proyecto hay que incluir las librerías de AOP



Gráfico 49. Librerías AOP y Spring Security

Todo el código de este ejemplo lo encontrareis en **Ejemplo10_Spring_Security_AOP.zip**

INDICE DE GRÁFICOS

Gráfico 1. Componentes de web.xml	5
Gráfico 2. Cargar el contexto spring-security.xml	6
Gráfico 3. Bloqueo con filtro DelegatingFilterProxy	6
Gráfico 4. Declaración de filtro DelegatingFilterProxy en el web.xml.....	7
Gráfico 5. El filtro delega en spring-security.xml.....	7
Gráfico 6. Contenido de spring-security.xml	8
Gráfico 7. Formulario login por defecto	9
Gráfico 8. Formulario login personalizado	9
Gráfico 9. login.jsp	10
Gráfico 10. Configuración form-login.....	10
Gráfico 11. checkbox Remember me.....	11
Gráfico 12. Elemento remember-me	12
Gráfico 13. Formulario con Remember me	12
Gráfico 14. Link para logout.....	13
Gráfico 15. Configuración logout	13
Gráfico 16. Redirección a https	14
Gráfico 17. Control de única sesión	14
Gráfico 18. Listener para detectar nuevas sesiones	14
Gráfico 19. Encriptar password	15
Gráfico 20. isAuthenticated	16
Gráfico 21. isAnonymous.....	16
Gráfico 22. isRememberMe	16
Gráfico 23. hasRole.....	17
Gráfico 24. hasAnyRole	17
Gráfico 25. hasIpAddress	17
Gráfico 26. isFullyAuthenticated	17
Gráfico 27. denyAll	17
Gráfico 28. permitAll	18
Gráfico 29. principal	18
Gráfico 30. authentication	18
Gráfico 31. isAuthenticated.....	18
Gráfico 32. Librería de etiquetas	20

Gráfico 33. Marcar secciones	20
Gráfico 34. Obtener el username.....	20
Gráfico 35. Habilitar anotación @Secured	21
Gráfico 36. Métodos anotados con @Secured	21
Gráfico 37. Habilitar anotación @RolesAllowed	21
Gráfico 38. Métodos anotados con @RolesAllowed	22
Gráfico 39. Configuración del servidor LDAP en memoria	24
Gráfico 40. Authentication provider para el servidor local	24
Gráfico 41. Archivo users.ldif.....	25
Gráfico 42. Servicio para LDAP remoto	26
Gráfico 43. Authentication provider para el servidor local	26
Gráfico 44. Acceso al servidor remoto a través de URL	26
Gráfico 45. Definición de las Queries	27
Gráfico 46. Proveedor de autenticación.....	27
Gráfico 47. Data source	28
Gráfico 48. Configuración con AOP	29
Gráfico 49. Librerías AOP y Spring Security	29