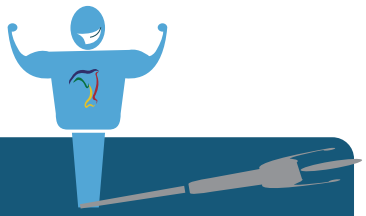
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

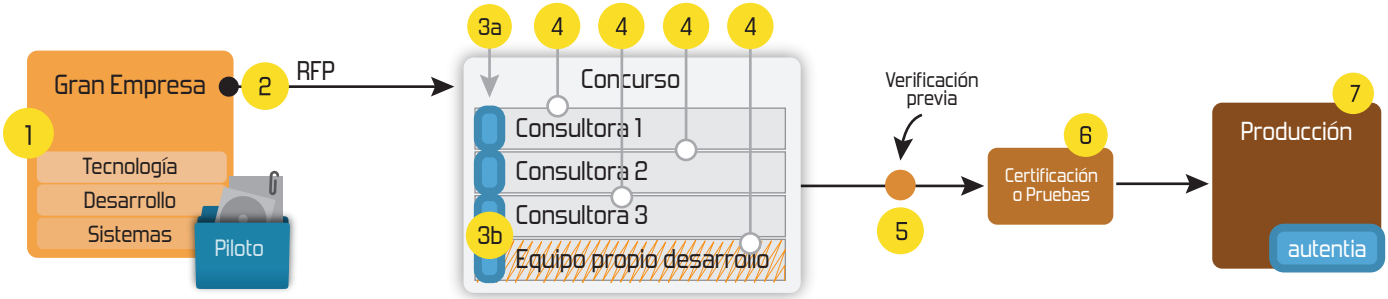
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Lanzado TNTConcept versión 0.8 (10/12/2007)

¿Gestionas tu empresa con hojas de cálculo? ¿No crees que puede haber un modo mejor?

Desde Autentia ponemos a vuestra disposición el software que hemos construido (100% gratuito, con código fuente disponible y sin restricciones funcionales) para nuestra gestión interna, llamado TNTConcept (au**T**e**NT**ia). Construida con las últimas tecnologías de desarrollo Java/J2EE (Spring, JSF, Acegi, Hibernate, Maven, Subversion, etc.) y disponible en licencia GPL, seguro que a muchos profesionales independientes y PYMES os ayudará a organizar mejor vuestra operativa.

Las cosas grandes empiezan siendo algo pequeño ..... Saber más en:
http://tntconcept.sourceforge.net/

Tutorial desarrollado por: Jose Manuel Sánchez Suárez

Puedes encontrarme en Autentia
Somos expertos en Java/J2EE
Contacta en:
jmsanchez@autentia.com

NUEVO CATÁLOGO DE SERVICIOS DE AUTENTIA (PDF 6,2MB)

www.adictosaltrabajo.com es el Web de difusión de conocimiento de www.autentia.com

autentia real business solutions

Catálogo de cursos

Descargar este documento en formato PDF jaas.pdf

Firma en nuestro libro de Visitas **<-----> Asociarme al grupo AdictosAlTrabajo en eConozco**

Fecha de creación del tutorial: 2007-12-10

JAAS (Java Authentication and Authorization Service) y Jboss.

0. Índice de contenidos.

- 1. Introducción
- 2. Entorno.
- 3. Creación de tablas y carga de datos en MySQL.
- 4. Definición del datasource y configuración de la política de seguridad en Jboss.
- 5. Estructura de nuestra aplicación JEE.
- 6. Securizando nuestra aplicación.
- 7. Obtención el usuario y su perfil asociado.
- 8. Integración en la librería de componentes jsf de Apache Tomahawk.
- 9. Conclusiones.

1. Introducción

JAAS (Java Authentication and Authorization Service) es una interfaz que permite a las aplicaciones Java acceder a servicios de control de autenticación y acceso. Apareció como paquete opcional en la versión 1.3 y lleva siendo parte del estándar desde la versión 1.4.

Puede usarse con dos fines:

- la autenticación de usuarios: para conocer quién está ejecutando nuestro código java,

- la autorización de usuarios: para garantizar que quién lo ejecuta tiene los permisos necesarios para hacerlo.

En este tutorial vamos a ver un ejemplo de cómo implementar dicha interfaz desde una aplicación JEE, en la que la información sobre los usuarios y sus perfiles la mantenemos en una base de datos.

Comprobaremos que la tarea se limita a la parametrización de ficheros de configuración (*.xml): del lado del servidor de aplicaciones y del lado de nuestra aplicación web.

El objetivo es delegar el mecanismo de autenticación y autorización en el servidor de aplicaciones. En la práctica significa, que ya no será necesario implementar un filtro de autorización en todas nuestras aplicaciones web y no tendremos que copiar el LoginAction.java o el LoginBean.java de una aplicación a otra arrastrando posibles errores.

Al implementar una interfaz que forma parte del estándar, nos beneficiamos no solo del acceso a la información del usuario y su perfil a través de métodos que forman parte del mismo (del estándar), si no que, además, nos podemos beneficiar de otras implementaciones que encapsulan dichos métodos dentro de su librería de tags, como el proyecto tomahawk de apache, lo veremos en el último punto.

Por último, al tratarse de una interfaz podríamos decidir modificar la política de seguridad pasado mañana, para que vez de autenticarnos contra base de datos lo hiciésemos contra un servidor LDAP... y lo único que tendríamos que tocar es un fichero de configuración en el servidor.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Sobremesa Dell Dimension 6400, 2.13 Ghz, 2 Gb RAM
- Sistema operativo: Windows XP Media center Edition
- JDK 1.6.0_2
- Eclipse 3.3.
- Jboss 4.2.1.GA
- MySQL 5.0

3. Creación de tablas y carga de datos en MySQL.

Si la autenticación la vamos a realizar contra base de datos, lo primero que necesitamos es crear las tablas en MySQL.

Ni el nombre de las tablas es fijo, ni el nombre de los campos de cada tabla, con lo que si ya tenemos un modelo de datos que responde a esta estructura no será necesario migrarlo. Veremos que, tanto la consulta de usuario, como la consulta de perfil se realizan a través de una sentencia sql parametrizable.

El siguiente script crea dos tablas:

- role: con una clave única y un nombre. El campo nombre será importante a la hora de definir la autorización.
- user: con una clave única, un login, una contraseña y una clave foránea que vincula un usuario a un rol.

view plain print ?

```
DROP TABLE IF EXISTS `user`;
DROP TABLE IF EXISTS `role`;
CREATE TABLE `role` (
`id` int(10) unsigned NOT NULL,
`name` varchar(64) collate utf8_spanish_ci NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
CREATE TABLE `user` (
`id` int(10) unsigned NOT NULL auto_increment,
`login` varchar(128) collate utf8_spanish_ci NOT NULL,
`password` varchar(128) collate utf8_spanish_ci NOT NULL,
`roleId` int(10) unsigned NOT NULL,
PRIMARY KEY (`id`),
KEY `ndx_user_roleId` (`roleId`),
CONSTRAINT `fk_user_roleId` FOREIGN KEY (`roleId`) REFERENCES `role`
(`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
```

El siguiente script inserta dos usuarios y dos roles:

- usuario:"admin", con contraseña:"admin", asociado al rol "ADMINISTRATOR"
- usuario:"user", con contraseña:"password", asociado al rol "USER"

view plain print ?

```
INSERT INTO `role` (`id`,`name`) VALUES
(1,'ADMINISTRADOR'),
(2,'USER');
INSERT INTO `user` (`id`,`login`,`password`,`roleId`) VALUES
(1,'admin','ISMvKXpXpadDiUoOSoAfww==',1),
(2,'user','X03MO1qnZdYdgyfeuILPmQ==',2);
```

Las contraseñas están encriptadas usando el algoritmo MD5 en Base64.

4. Definición del datasource y configuración de la política de seguridad en Jboss.

La definición de una fuente de datos en Jboss se realiza añadiendo un fichero con extensión -ds.xml en el directorio **jboss-4.2.1.GA\server\default\deploy**, con el siguiente contenido:

```
view plain  print  ?
01. <?xml version="1.0"
02. encoding="UTF-8"?>
03. <datasources>
04. <local-tx-datasource>
05. <jndi-name>simpleDS</jndi-name>
06. <connection-url>jdbc:mysql://localhost:3306/test</connection-url>
07. <driver-class>com.mysql.jdbc.Driver</driver-class>
08. <user-name>test</user-name>
09. <password>test</password>
10. </local-tx-datasource>
11. </datasources>
```

Tenemos que configurar:

- jndi_name: con un id único para nuestro datasource, que usaremos para invocarlo desde nuestra política de seguridad (el siguiente punto),
- connection-url: con la dirección y el nombre de nuestra base de datos MySQL
- user-name: con el login de un usuario para acceder a la base de datos. Claro está que no se refiere a uno de los usuarios de las tablas antes creadas, sino a un usuario de MySQL con permisos de lectura para acceder a la base de datos.
- password: con la contraseña de acceso del usuario anterior.

En nuestro caso tenemos un fichero **jboss-4.2.1.GA\server\default\deploy\test-ds.xml** para el acceso a la base de datos MySQL de test, con un usuario de MySQL test con contraseña test.

Jboss no viene con el driver para MySQL preinstalado, con lo que, si aún no lo tenemos, deberemos obtenerlo de la siguiente ubicación Download Connector/J 5.0: y copiar el jar, mysql-connector-java-5.0.*-bin.jar, en **jboss-4.2.1.GA\server\default\lib**

En el fichero **jboss-4.2.1.GA\server\default\conf\login-config.xml** debemos introducir la siguiente política de seguridad:

```
view plain  print  ?
01. <!--#simpleWeb_config -->
02. <application-policy
03. name="simpleWebAuthenticationPolicy">
04. <authentication>
05. <login-module
06. code="org.jboss.security.auth.spi.DatabaseServerLoginModule"
07. flag="required">
08. <module-option
09. name="dsJndiName">java:/simpleDS</module-option>
10. <module-option name="principalsQuery">select password
11. from User where login=?</module-option>
12. <module-option name="rolesQuery">select r.name, 'Roles'
13. from User u, Role r where u.roleId=r.id and
14. login=?</module-option>
15. <module-option
16. name="hashAlgorithm">MD5</module-option>
17. <module-option
18. name="hashEncoding">base64</module-option>
19. </login-module>
20. </authentication>
21. </application-policy>
```

Tenemos que configurar:

- <application-policy name="**simpleWebAuthenticationPolicy**">: con un id único que identifica nuestra política de seguridad, que usaremos desde la aplicación web para enlazarla,
- <module-option name="dsJndiName">java:**simpleDS**<module-option>: con el nombre del datasource que hemos definido en el punto anterior,
- <module-option name="principalsQuery">**select password from User where login=?**</module-option>: con la sentencia sql para la obtención de la contraseña del usuario que recibe como parámetro. Como comentaba, si vuestro modelo de datos no se corresponde con el de este ejemplo, aquí asignaríamos la sentencia correspondiente.
- <module-option name="rolesQuery">**select r.name, 'Roles' from User u,Role r where u.roleId=r.id and login=?**</module-option>: con la sentencia sql para la obtención del nombre del perfil asociado al usuario, con las mismas consideraciones que en el punto anterior,
- <module-option name="hashAlgorithm">**MD5**</module-option>:indica el nombre de algoritmo que hemos usado para guardar nuestras contraseñas en la tabla de usuarios,
- <module-option name="hashEncoding">**base64**</module-option>: indica el tipo de encoding que usará el algoritmo para comprobar la contraseña.

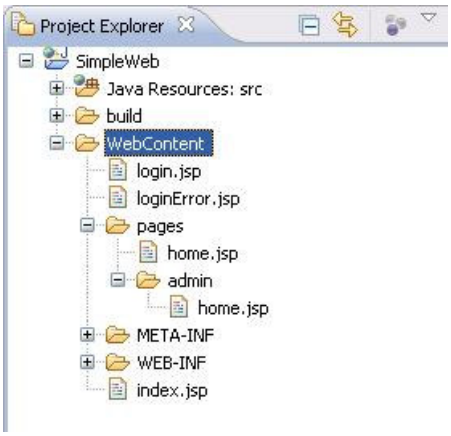
En este punto hemos configurado el módulo JAAS de Jboss para que acceda a nuestra base de datos a través de un

datasource, el mismo que debería usar nuestra aplicación web.

Cambiar de base de datos, para pasar a un entorno de producción, por ejemplo, es sencillo, basta con modificar el vínculo (el nombre del datasource) entre la política de seguridad y el datasource, o lo que es más común, al tener varios datasources (uno por entorno), tendríamos varias políticas de seguridad (una por cada uno de ellos).

5. Estructura de nuestra aplicación JEE.

Crearemos un proyecto web dinámico en Eclipse, en nuestro caso con nombre **SimpleWeb**, con la siguiente estructura de directorios:



La idea es que:

- el contenido del directorio **/pages** no sea visible si no se ha accedido a la aplicación a través de la página de login, y
- el contenido del directorio **/pages/admin** no sea visible a todos los usuarios, sólo a aquellos con perfil de administración.

Veamos primero el código de nuestras páginas:

La página **index.jsp**, tendría el siguiente código:

```
view plain  print  ?
01. <%@page language="java" contentType="text/html;
02. charset=UTF-8"%>
03. <%request.getSession().invalidate();%>
04. <%response.sendRedirect("pages/home.jsp");%>
```

03. Limpia la sesión para eliminar el login de usuario si lo hubiese.
04. Redireccióna la petición a una página que estará protegida, la que se mostrará después de un login satisfactorio.

El filtro de autorización de JAAS funciona de modo que, si se accede a una página protegida y no hay usuario en la sesión, te remite a la página de login. Teniendo en cuenta esto, lo que debe hacer nuestra página de inicio es redirigir la petición a una página protegida, para que salte la página de login y tras un login satisfactorio nos redirija a la página protegida.

Un error común es no llevar a cabo ésta primera redicción, y marcar como página de entrada el login.jsp, si hacemos esto (podemos probarlo), tras el login, nos intentará redirijir al login, apareciendo el siguiente error:

```
Estado HTTP 400 – Referencia directa al formulario de conexión (página de formulario de login) inválida
```

La página **login.jsp**, tendría el siguiente código:

```
view plain  print  ?
01. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
02. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><%@page
03. language="java" contentType="text/html; charset=UTF-8"%>
04. <html xmlns="http://www.w3.org/1999/xhtml">
05. <head>
06. <title>login</title>
07. </head>
08. <body>
09. <div id="layout" >
10. <form method="post" action="j_security_check"
11. name="j_security_check">
12. <table id="loginTable">
13. <tbody>
14. <tr>
15. <td class="loginLabel">
16. <span>Usuario</span>
17. </td>
18. <td>
19. <input type="text" name="j_username" />
20. </td>
21. </tr>
22. <tr>
23. <td class="loginLabel">
24. <span>Contraseña</span>
25. </td>
26. <td>
27. <input type="password" name="j_password" />
28. </td>
29. </tr>
30. <tr>
31. <td colspan="2">
32. <input type="submit" value="Entrada al sistema" />
33. </td>
34. </tr>
35. </tbody>
36. </table>
37. </form>
38. </div>
39. </body>
40. </html>
```

Contiene el formulario de entrada de usuario y contraseña:

- es obligatorio que el formulario tenga un name concreto (**j_security_check**) y un action concreto (**j_security_check**), el action invoca al filtro de autorización de JAAS,
- el nombre del campo de entrada para el login de usuario debe llamarse **j_username** y el campo password **j_password**.

La página **loginError.jsp**, contiene un mensaje de error informando que el usuario y/o la contraseña introducidos no son correctos:

```
view plain  print  ?
01. <%@ page language="java" contentType="text/html;
02. charset=ISO-8859-1"
03. pageEncoding="ISO-8859-1"%>
04. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
05. "http://www.w3.org/TR/html4/loose.dtd">
06. <html>
07. <head>
08. <meta http-equiv="Content-Type" content="text/html;
09. charset=ISO-8859-1">
10. <title>Login Error</title>
11. </head>
12. <body>
13. <p>Usuario y/o contraseña
14. erróneos</p>
15. <p><a href="#"
16. onclick="history.back();">Volver</a></p>
17. </body>
18. </html>
```

La página **/pages/home.jsp**, contiene un título y un enlace para acceder a la administración del web:

```
view plain  print  ?
01. <%@ page language="java" contentType="text/html;
02. charset=ISO-8859-1"
03. pageEncoding="ISO-8859-1"%>
04. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
05. "http://www.w3.org/TR/html4/loose.dtd">
06. <html>
07. <head>
08. <meta http-equiv="Content-Type" content="text/html;
09. charset=ISO-8859-1">
10. <title>Home de SimpleWeb</title>
11. </head>
12. <body>
13. <h1>Home de SimpleWeb</h1>
14. <p><a href="admin/home.jsp">Acceso a
15. administración</a></p>
16. </body>
17. </html>
```

La página **/pages/admin/home.jsp**, contiene un título informando que estamos en la administración del web:

```
view plain  print  ?
01. <%@ page language="java" contentType="text/html;
02. charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
03. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04. "http://www.w3.org/TR/html4/loose.dtd">
05. <html>
06. <head>
07. <meta http-equiv="Content-Type" content="text/html;
08. charset=ISO-8859-1">
09. <title>Home de la administración de
10. SimpleWeb</title>
11. </head>
12. <body>
13. <h1>Home de la administración de
14. SimpleWeb</h1>
15. </body>
16. </html>
```

Si arrancamos el servidor y accedemos a la aplicación en éste punto del tutorial accediendo a al siguiente url: http://localhost:8080/SimpleWeb/ veremos que:

- tenemos acceso al home (pages/home.jsp) de nuestro web a través de la página de inicio, sin necesidad de introducir el usuario y contraseña, y
- desde la home tenemos acceso a la administración pulsando sobre el enlace.

6. Securizando nuestra aplicación.

En el fichero **WebContent/WEB-INF/web.xml** añadiremos las siguientes restricciones de seguridad:

```
view plain  print  ?
01. <!--#Security constraints -->
02. <security-constraint>
03. <web-resource-collection>
04. <web-resource-name>User Authentication
05. Policy</web-resource-name>
06. <url-pattern>/pages/*</url-pattern>
07. </web-resource-collection>
08. <auth-constraint>
09. <role-name>*</role-name>
10. </auth-constraint>
11. </security-constraint>
12. <security-constraint>
13. <web-resource-collection>
14. <web-resource-name>Administrator Authentication
15. Policy</web-resource-name>
16. <url-pattern>/pages/admin/*</url-pattern>
17. </web-resource-collection>
18. <auth-constraint>
19. <role-name>ADMINISTRATOR</role-name>
20. </auth-constraint>
21. </security-constraint>
```

El contenido del directorio /pages/* (**06. <url-pattern>/pages/*</url-pattern>**) será visible a todos los usuarios,

independientemente de su perfil (**09. <role-name>*</role-name>**).

El contenido del directorio /pages/admin/* (**16. <url-pattern>/pages/admin/*</url-pattern>**) será visible solo para los usuarios con un perfil de administrador (**19. <role-name>ADMINISTRATOR</role-name>**), donde ADMINISTRATOR es el nombre del perfil en base de datos. Que debe corresponderse con el resultado de la sentencia sql definida en el parámetro rolesQuery del fichero login-config.xml.

En el mismo fichero **WebContent/WEB-INF/web.xml** añadiremos la definición de los 2 roles que tenemos:

```
view plain  print  ?
01. <!--#Roles -->
02. <security-role>
03. <description>Administrator
04. User</description>
05. <role-name>ADMINISTRATOR</role-name>
06. </security-role>
07. <security-role>
08. <description>Final User</description>
09. <role-name>USER</role-name>
10. </security-role>
```

El contenido de los campos <role-name> se corresponde con el nombre de los perfiles en base de datos.

Por último, en el mismo fichero **WebContent/WEB-INF/web.xml** añadiremos la definición del formulario de entrada de login y contraseña:

```
view plain  print  ?
01. <!--#Form-Based Authentication -->
02. <login-config>
03. <auth-method>FORM</auth-method>
04. <realm-name>Form-Based Authentication
05. Area</realm-name>
06. <form-login-config>
07. <form-login-page>/login.jsp</form-login-page>
08. <form-error-page>/loginError.jsp</form-error-page>
09. </form-login-config>
10. </login-config>
```

Definimos el método de autenticación (a través de formulario), la página de login y la página que aparecerá tras un error en el login.

El m&eacutetodo de autenticación puede ser también un diálogo de sistema con <auth-method>BASIC</auth-method>.

Para enlazar el filtro JAAS con el módulo de autenticación de jboss, crearemos un fichero **WebContent/WEB-INF/jboss-web.xml** con el siguiente contenido:

```
view plain  print  ?
01. <jboss-web>
02. <security-domain>java:/jaas/simpleWebAuthenticationPolicy</security-domain>
03. </jboss-web>
```

El nombre de la política de seguridad (simpleWebAuthenticationPolicy) es el definido en el campo <application-policy name=""> del fichero login-config.xml.

Si probamos de nuevo la aplicación veremos como "salta" la página de login al acceder al index:

- si accedemos con el usuario admin:admin, veremos como tenemos acceso a la página de login y pulsando sobre el enlace accedemos a la página de administración
- si accedemos con el usuario user:password, veremos como tenemos acceso a la página de login y pulsando sobre el enlace aparecerá un error de autorización, porque no forma parte del perfil de administradores.

```
Estado HTTP 403 - El acceso al recurso pedido ha sido denegado
```

7. Obtención el usuario y su perfil asociado.

La interfaz HttpServletRequest nos proporciona dos métodos para acceder a información sobre el usuario autenticado:

- **getUserPrincipal**: devuelve el usuario conectado,
- **isUserInRole**: recibe como parámetro un string y devuelve un boolean, true si el usuario pertenece al rol que se pasa como parámetro,

Vamos a modificar la página **pages/home.jsp** para ocultar el enlace hacía la administración si el usuario no tiene un perfil de administrador, y para mostrar el name del usuario.

```
view plain  print  ?
01. <%@ page language="java" contentType="text/html;
02. charset=ISO-8859-1"
03. pageEncoding="ISO-8859-1"%>
04. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
05. "http://www.w3.org/TR/html4/loose.dtd">
06. <html>
07. <head>
08. <meta http-equiv="Content-Type" content="text/html;
09. charset=ISO-8859-1">
10. <title>Home de SimpleWeb</title>
11. </head>
12. <body>
13. <h1>Home de SimpleWeb</h1>
14. <p>Usuario: <%=request.getUserPrincipal().getName()%></p>
15. <% if ( request.isUserInRole("ADMINISTRATOR"))
16.   {
17. %>
18. <p><a href="admin/home.jsp">Acceso a administración</a></p>
19. <% } %>
20.
21. </body>
22. </html>
```

8. Integración en la librería de componentes jsf de Apache Tomahawk.

Si estamos trabajando con la implementación de jsf de myfaces y añadimos a nuestra librería de tags la extensión de Apache Tomahawk, podremos comprobar como uno de los atributos de los tags es **visibleOnUseRole**.

Vamos a modificar la página **pages/home.jsp**:.

```
view plain  print  ?
01. <%@ page language="java" contentType="text/html;
02. charset=ISO-8859-1"
03. pageEncoding="ISO-8859-1"%>
04. <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
05. <%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>
06. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
07. "http://www.w3.org/TR/html4/loose.dtd">
08. <f:view>
09. <html>
10. <head>
11. <meta http-equiv="Content-Type" content="text/html;
12. charset=ISO-8859-1">
13. <title>Home de SimpleWeb</title>
14. </head>
15. <body>
16. <h1>Home de SimpleWeb</h1>
17. <p>Usuario: <%=request.getUserPrincipal().getName()%></p>
18. <h:commandLink onclick="admin/home.jsf" value="Acceso a administracion" visibleOnUserRole="ADMINISTR
19. </body>
20. </html>
21. </f:view>
```

El atributo visibleOnUserRole puede recibir varios perfiles separador por comas.

9. Conclusiones.

¿Para qué vamos a implementar un mecanismo de autenticación y/o autorización propio, si tenemos uno dentro del estándar JEE?. Si además, es relativamente fácil de configurar...

Para ampliar información sobre éste tutorial consultad el Capítulo 8 (Seguridad) del manual de Jboss:

- buscar org.jboss.security.auth.spi.LdapLoginModule: para ver cómo configurar la seguridad contra un servidor LDAP,
- buscar org.jboss.security.auth.spi.UsersRolesLoginModule: para ver cómo configurar la seguridad a través de ficheros de propiedades.

También podéis revisar éste tutorial de Roberto Canales sobre Seguridad en Tomcat.

Una reflexión para terminar: *"Para entrar en casa ajena puedes, echar la puerta abajo a patadas (con más o menos perseverancia) o, simplemente mirar debajo del felpudo para ver si el dueño deja ahí una copia de sus llaves"*.

Un saludo.

Jose

This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 2.5 License.

Puedes opinar sobre este tutorial aquí

Recuerda

que el personal de Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales)

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?

¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

info@autentia.com

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos .....
Autentia = Soporte a Desarrollo & Formación

J2EE, EJBs, Struts...

Autentia S.L. Somos expertos en:
J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..
y muchas otras cosas

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	**Descripción**

Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores.

En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo.

Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.

Patrocinados por enredados.com .... Hosting en Castellano con soporte Java/J2EE

www.AdictosAlTrabajo.com Opimizado 800X600