

¿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

Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

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

BPM (jBPM o Bonita)
 Generación de informes (JasperReport)
 ESB (Open ESB)

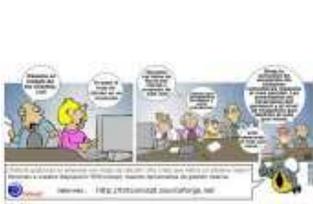


Comic

NUEVO ¿Quieres saber cuánto ganas en relación al mercado? pincha **aquí...**

[Ver cursos que ofrece Autentia](#)

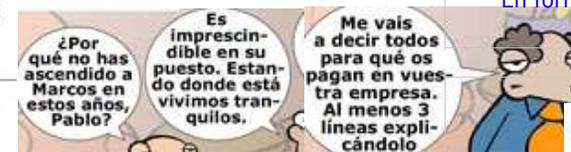
[Descargar comics en PDF y alta resolución](#)



[¡NUEVO!] 2008-09-01



2008-07-31



2008-07-08

2008-06-2

Catálogo de servicios Autentia (PDF 6,2MB)



[En formato comic...](#)

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la sección [comic](#).

Web

www.adictosaltrabajo.com

Tutorial desarrollado por



Carlos García Pérez

Creador del pionero Web [MobileTest](#).

Consultor tecnológico en el desarrollo de proyectos informáticos.

Ingeniero Técnico en Informática *

Puedes encontrarme en [Autentia](#)

Somos expertos en Java/J2EE

Catálogo de servicios de Autentia

[Descargar \(6,2 MB\)](#)

[Descargar en versión comic \(17 MB\)](#)

[AdictosALTrabajo.com](#) es el Web de difusión de conocimiento de [Autentia](#).



[Catálogo de cursos](#)

Últimos tutoriales

2008-10-30

[Autentificación y Autorización mediante JAAS](#)

2008-10-27

[Web Services con Estado](#)

2008-10-24

[Web Services con Axis2. Configuración y ejemplo](#)

2008-10-22

[Migración de JSP a Facelets](#)

2008-10-22

[Rock Band Wii en tu PC](#)

2008-10-18

[Cobertura: Como comprobar cuanto código prueban nuestros test](#)

2008-10-17

[maven-license-plugin: cómo gestionar la licencia de nuestros ficheros fuentes](#)

Descargar este documento en formato PDF: [jaas_example.pdf](#)

Fecha de creación del tutorial: 2008-10-30

Autentificación y Autorización mediante JAAS

Normalmente, casi todas las aplicaciones o sistemas necesitan de los procesos de autentificación y autorización. En este tutorial, vamos a realizar un completo ejemplo de autentificación de usuarios así como realizar un control de las posibles acciones que pueden realizar en función de su identidad. Para ello vamos a basarnos en el estándar de seguridad Java pensado para estas tareas, es decir, vamos a hacer uso de JAAS.

En este tutorial vamos a realizar un completo ejemplo a través del cual dos usuarios podrán logarse y tendrán distintos privilegios en función de quien sea.

Índice de contenido

- [Un poco de teoría: Introducción a JAAS](#)
- [Comprender JAAS a través de un completo ejemplo.](#)
 - [Salidas generadas por la aplicación](#)

- o Autenticación JAAS: Archivo de configuración
- o Autorización JAAS: Archivo de configuración
- o Aplicación que será autenticada y autorizada mediante JAAS
- o Acciones realizadas por la aplicación y autorizadas o no por el AccessController de Java Security
- o Implementación de un LoginModule que autentifica a un usuario a través de su usuario y su contraseña
- o Implementación de javax.security.auth.callback.CallbackHandler: Solicitando al usuario su login y el contraseña.
- o Implementación de java.security.Principal: Representando la identidad de los usuarios.
- Referencias
- Conclusiones

Un poco de teoría: Introducción a JAAS

JAAS son las siglas de Java Authentication and Authorization. Se trata de una especificación integrada en la máquina virtual Java a partir de la versión 1.4 y cuya finalidad es la de **definir un estándar para los procesos de autenticación y autorización**.

Es decir ambos procesos, autenticación y autorización están directamente relacionados con la seguridad de aplicaciones.

¿Qué es la autenticación?

La **autenticación** es el proceso por el cual un usuario o servicio tiene que autenticarse para poder acceder a ciertos servicios que ofrece nuestro sistema.

Existen distintas categorías de autenticación:

- ¿Qué sabes?: Información que el usuario conoce, ejemplos: contraseñas, respuestas a preguntas como: "dni de la abuela"
- ¿Qué tienes?: Elementos físicos que el usuario posee, como por ejemplo una determinada tarjeta.
- ¿Quién eres?: Técnicas biométricas como lectura de retina o huellas dactilares.

¿Qué es la autorización?

La **autorización** es el proceso por el cual se controlan las acciones que tiene un usuario o servicio **normalmente ya autenticado** puede realizar, para ello se le conceden o deniegan permisos.

En Java, aunque muchos libros lo dicen, no es del todo cierto que el proceso de autorización implique que el usuario previamente haya sido autenticado. Digo esto, porque en Java se pueden definir permisos en base a otros parámetros como por ejemplo a nivel de "quién firma el código a ejecutar" (opción SignedBy de la configuración de privilegios) o "En qué JAR está el código a ejecutar" (opción CodeBase de la configuración de privilegios).

Categorías de autorización:

- Autorización declarativa: En este tipo de autorización los privilegios son gestionados un administrador independientemente de manera externa al código de la aplicación
- Autorización programática: En este tipo de autorización las decisiones de autorización se realizan desde el código fuente de la aplicación.

¿Por qué es bueno que use JAAS en mis aplicaciones en vez de realizar yo mismo ese control?

Al tratarse de una especificación estándar, hay muchas **implementaciones disponibles** dándote entre otros los siguientes beneficios:

- Ahorrarás tiempo y dinero, pues podrás usar la implementación que más se asemeje a tus necesidades.
- La especificación está documentada y al ser estándar encontrará más fácilmente personal que tenga conocimientos sobre el tema.
- Cambiar una implementación por otra, puede ser tan sencillo como cambiar el archivo de configuración.
- Aprovecharse de las nuevas versiones que vayan saliendo, sin invertir dinero en su desarrollo.
- Etc.

Comprender JAAS a través de un completo ejemplo.

En este tutorial vamos a realizar un completo ejemplo a través del cual dos usuarios podrán logarse y tendrán distintos privilegios en función de quién sea.

- El usuario1 tendrá privilegios para ver un determinado directorio y para crear un socket que se conecte al puerto 80 del dominio autentia.com.
- El usuario2 tendrá privilegios para ver un determinado directorio, pero **no podrá crear un socket** que se conecte al puerto 80 del dominio autentia.com.

Para ello, realizaremos las siguientes cosas:

- Crearemos los archivos de configuración de autenticación y autorización
- Implementar un `javax.security.auth.spi.LoginModule` que permita autenticar usuarios solicitando para ellos un usuario y un password.
- Implementar un `javax.security.auth.callback.CallbackHandler` que solicite al usuario su login y el contraseña.
- Implementar un `java.security.Principal` que represente la identidad de los usuarios.

Salidas generadas por la aplicación

2008-10-10
Cypal Studio: plugin de GWT para Eclipse

2008-10-10
Planificación de tareas con Spring

2008-10-09
Tutorial de Google Calendar Sync

Últimas ofertas de empleo

2008-10-27
T. Información - Analista / Programador - CIUDAD REAL.

2008-10-03
Marketing - Experto en Marketing - MADRID.

2008-10-01
Atención a cliente - Call Center - MADRID.

2008-09-11
Otras Sin catalogar - BARCELONA.

2008-08-11
Atención a cliente - Call Center - MADRID.

Anuncios Google

Antes de comenzar a ver el código de la aplicación, pienso que es interesante que vea las diferentes salidas que generará la aplicación al ser ejecutada.

Salida generada al ejecutar la aplicación logándonos incorrectamente:

```
java -classpath .; -Djava.security.manager -Djava.security.policy=../conf/policy.jaas
-Djava.security.auth.login.config=../conf/login.conf com.autentia.tutoriales.jaas.JAAS_App

Introduzca su nombre de usuario: carlos
Introduzca su contraseña: 1234
[SimpleLoginModule] El usuario introdujo como nombre de usuario: carlos
[SimpleLoginModule] El usuario introdujo como contraseña: 1234
[SimpleLoginModule] El usuario NO ha sido autenticado satisfactoriamente
javax.security.auth.login.LoginException: javax.security.auth.login.FailedLoginException: Usuario/Contraseña incorrecta.
```

Salida generada al ejecutar la aplicación logándonos correctamente como usuario1:

```
java -classpath .; -Djava.security.manager -Djava.security.policy=../conf/policy.jaas
-Djava.security.auth.login.config=../conf/login.conf com.autentia.tutoriales.jaas.JAAS_App

Introduzca su nombre de usuario: usuario1
Introduzca su contraseña: abcd
[SimpleLoginModule] El usuario introdujo como nombre de usuario: usuario1
[SimpleLoginModule] El usuario introdujo como contraseña: abcd
[SimpleLoginModule] El usuario ha sido autenticado satisfactoriamente
[SimpleLoginModule].commit()
[SimpleLoginModule].commit() Añadimos la identidad (SimplePrincipal) al Subject
file.separator: \
NO se pudo leer la propiedad: java.security.policy
SI se pudo realizar la operación con el Socket
Hay 40 archivos.
```

Salida generada al ejecutar la aplicación logándonos correctamente como usuario2:

```
java -classpath .; -Djava.security.manager -Djava.security.policy=../conf/policy.jaas
-Djava.security.auth.login.config=../conf/login.conf com.autentia.tutoriales.jaas.JAAS_App

Introduzca su nombre de usuario: usuario2
Introduzca su contraseña: 1234
[SimpleLoginModule] El usuario introdujo como nombre de usuario: usuario2
[SimpleLoginModule] El usuario introdujo como contraseña: 1234
[SimpleLoginModule] El usuario ha sido autenticado satisfactoriamente
[SimpleLoginModule].commit()
[SimpleLoginModule].commit() Añadimos la identidad (SimplePrincipal) al Subject
file.separator: \
NO se pudo leer la propiedad: java.security.policy
NO se pudo realizar la operación con el Socket java.security.AccessControlExcept
ion: access denied (java.net.SocketPermission autentia.com resolve)
Hay 40 archivos.
```

Autenticación JAAS: Archivo de configuración del Módulo de Login (login.conf)

Aunque existen varios módulos de Login **ya implementados: Solaris Login Module, NT Login Module, JNDI Login Module...**, nosotros vamos a realizar el nuestro teniendo en mente que luego sustituirlo por otro, desde el punto de vista de nuestra aplicación, es tan fácil como cambiar la configuración (y por supuesto aprender a configurar el otro leyendo su documentación).

```
JAASApplicationDemo {
    com.autentia.tutoriales.jaas.jaas.SimpleLoginModule required debug=true;
};
```

Observe como se integra con el resto de componentes:

- Fijese en la línea 24 del código de la aplicación JAAS_App que se muestra más abajo.
- Fijese en la línea 3 del archivo de configuración de autorización (policy.jaas) que se muestra más abajo.

Autorización JAAS: Archivo de configuración (policy.jaas)

Ahora definimos los permisos que tiene nuestra aplicación a través de un archivo de configuración. Existen otras muchas formas de definir estos permisos (por ejemplo, BD), pero nosotros usaremos esta forma por defecto.

Nota: Si observa el archivo <JRE_HOME>/lib/security/security.policy verá una línea como está:
login.configuration.provider=com.sun.security.auth.login.ConfigFile, es decir, cargar a partir de un archivo.

```
/* Permisos comunes para todos los usuarios */
grant {
    permission javax.security.auth.AuthPermission "modifyPrincipals";
```

```

    permission javax.security.auth.AuthPermission "createLoginContext.JAASApplicationDemo";
};

/* Permisos para el usuario1 */
grant Principal com.autentia.tutoriales.jaas.jaas.SimplePrincipal "usuario1" {
    /* El usuario solo puede leer los archivos y directorios de c:/ */
    permission java.io.FilePermission "c:/", "read";

    /* Concedemos privilegios para conectarnos al puerto 80 del dominio autententia.com */
    permission java.net.SocketPermission "autentia.com:80", "connect,resolve";
};

/* Permisos para el usuario2 */
grant Principal com.autentia.tutoriales.jaas.jaas.SimplePrincipal "usuario2" {
    /* El usuario solo puede leer los archivos y directorios de c:/ */
    permission java.io.FilePermission "c:/", "read";
};

```

Recuerde la sintaxis de la sentencia grant es (todos son parámetros opcionales donde el orden no importa):

```

grant <signer(s) field>, <codeBase URL>
  <Principal field(s)> {
    permission perm_class_name "target_name", "action";
    ....
    permission perm_class_name "target_name", "action";
  };

```

La sintaxis de <Principal field(s)> es:

```
Principal Principal_class "principal_name"
```

Por si no lo sabe, los privilegios describen en base a esos Principals y estos son asociados al usuario (o servicio) que pretende realizar la operación, en el proceso de Autenticación.

Por ejemplo: Si eres un usuario (Subject) que tiene **entre todos sus principals** un GroupPrincipal de nombre "administrador", entonces puedes realizar esta operación, ver esta ventana, etc...

En nuestro ejemplo, estamos diciendo, **si posees un SimplePrincipal de nombre "usuario2"**, entonces tienes permisos para listar el directorio c:/

Es decir, el concepto de Principal y de Permission son conceptos abstractos que podemos implementar en base a nuestras necesidades... Los Frameworks nos abstraen de esto definiendo ellos mismos estos conceptos.

Aplicación que será autenticada y autorizada mediante JAAS

A continuación, mostramos el código fuente de nuestra aplicación (en este caso de escritorio) que realizará la autenticación y la autorización de las operaciones sobre el usuario que se loge.

Vuelvo a repetir, los Framework de seguridad como **Spring Security** nos abstraen de todo esto, pero ellos lo realizan por dentro.

```
01. package com.autentia.tutoriales.jaas;
02.
03. import javax.security.auth.*;
04. import javax.security.auth.login.*;
05. import com.autentia.tutoriales.jaas.actions.TestAction;
06. import com.autentia.tutoriales.jaas.jaas.PromptCallbackHandler;
07.
08. /**
09.  * Aplicación que será Autenticada y Autorizada mediante JAAS
10.  * @author Carlos García. Autentia Real Business Solutions
11.  * @see http://www.mobiletest.es
12.  */
13. public class JAAS_App {
14.     /**
15.      * Punto de entrada a la aplicación.
16.      * java -classpath .; -Djava.security.manager
17.      *      / -Djava.security.policy=../conf/policy.jaas
18.      *      / -Djava.security.auth.login.config=../conf/login.conf
19.      *      / com.autentia.tutoriales.jaas.JAAS_App
20.      */
21.     public static void main(String[] args) {
22.         try {
23.             // Inicializamos el LoginContext bajo las restricciones defini
24.             LoginContext lc = new LoginContext("JAASAplicacionDemo", new PromptCallb
25.
26.             // Logamos al usuario (Lo autenticamos)
27.             lc.login();
28.             // El usuario ha sido correctamente autenticado.
29.
30.             // Intentamos realizar una operación (Autorización) bajo las
31.             Integer count = (Integer) Subject.doAsPrivileged(lc.getSubject(), new Te
32.
33.             // Imprimimos el resultado de la operación y también los Pri
34.             System.out.println("Hay " + count.intValue() + " archivos.");
35.
36.             // Deslogamos al usuario
37.             lc.logout();
38.
39.             Runtime.getRuntime().exit(0);
40.         } catch (Exception ex){
41.             System.out.println(ex.toString());
42.         }
43.     }
44. }
45.
```

Acciones realizadas por la aplicación y autorizadas o no por el AccessController de Java Security

Las siguientes operaciones están vigiladas por al Access Controller

```

01. package com.autentia.tutoriales.jaas.actions;
02.
03. /**
04.  * Esta clase será ejecutada bajo el control del AccessController de J2
05.  * @author Carlos García. http://www.autentia.com
06.  * @see http://www.mobiletest.es
07.  */
08. public class TestAction implements java.security.PrivilegedAction {
09.     private String dirName;
10.
11.     public TestAction(String dirName){
12.         this.dirName = dirName;
13.     }
14.
15.     /**
16.      * @see java.security.PrivilegedAction#run()
17.      */
18.     public Object run() {
19.         java.io.File directory = new java.io.File(dirName);
20.         java.io.File files[] = directory.listFiles( );
21.
22.         this.sameDemoUseOfPermissions();
23.
24.         return Integer.valueOf(files.length);
25.     }
26.
27.     /**
28.      * Realizamos algunas acciones para ver el uso de los Permissions
29.      */
30.     private void sameDemoUseOfPermissions() {
31.         // Tiene permisos por que está permitida la operación en el arch
32.         // Si comentarios la línea veremos que falla la aplicación.
33.         try {
34.             System.out.println("file.separator: " + System.getProperty("file.separator"));
35.         } catch (java.security.AccessControlException ex){
36.             System.out.println("NO se pudo leer la propiedad: file.separator");
37.         }
38.
39.         // Debemos añadir la siguiente línea al archivo <JRE_HOME>/lib/s
40.         // permission java.util.PropertyPermission "java.security.policy
41.         try {
42.             System.out.println("java.security.policy: " + System.getProperty("java.se
43.         } catch (java.security.AccessControlException ex){
44.             System.out.println("NO se pudo leer la propiedad: java.security.policy")
45.         }
46.
47.         // Intentamos establecer una conexión al puerto 80 del servidor
48.         try {
49.             java.net.Socket s = new java.net.Socket("autentia.com", 80);
50.             s.close();
51.             System.out.println("SI se pudo realizar la operación con el Socket");
52.         } catch (Exception ex){
53.             System.out.println("NO se pudo realizar la operación con el Socket " + e
54.         }
55.     }
56. }
57. }
58.

```

Implementación de un LoginModule que autentifica a los usuarios a través de su usuario y su contraseña

El módulo está autocomentario, debería ser suficiente para que comprenda los pasos a realizar.

view plain print ?

```
01. package com.autentia.tutoriales.jaas.jaas;
02.
03. import java.util.*;
04. import javax.security.auth.*;
05. import javax.security.auth.callback.*;
06. import javax.security.auth.login.*;
07. import javax.security.auth.spi.*;
08.
09. /**
10.  * Implementación de LoginModule que reconoce a los usuarios a través c
11.  * Los usuarios válidos son: (usuario1, abcd) o (usuario2, 1234)
12.  * @author Carlos García. http://www.autentia.com
13.  * @see http://www.mobiletest.es
14.  */
15. public class SimpleLoginModule implements LoginModule {
16.
17.     private Subject subject;
18.     private CallbackHandler callbackHandler;
19.     private Map state;
20.     private Map options;
21.
22.
23.     private boolean debug;
24.     private boolean succeeded;
25.     private boolean commitSucceeded;
26.
27.     // El usuario y la contraseña
28.     private String username;
29.     private char[] password;
30.
31.     private SimplePrincipal userPrincipal;
32.
33.     /**
34.      * Inicializa el módulo de login
35.      * @param subject           El Subject a autentificar
36.      * @param callbackHandler   El CallbackHandler para interactuar cor
37.      * @param state             El LoginModule state.
38.      * @param options           Opciones de configuración del módulo de
39.      */
40.     public void initialize(Subject subject, CallbackHandler callbackHandler, Map state) {
41.         this.subject = subject;
42.         this.callbackHandler = callbackHandler;
43.         this.state = state;
44.         this.options = options;
45.
46.
47.         this.succeeded = false;
48.         this.commitSucceeded = false;
49.
50.         this.debug = "true".equalsIgnoreCase((String)options.get("debug"));
51.     }
52.
53.     /**
54.      * Autentifica al usuario preguntandole su nombre de usuario y cont
55.      * @return true in all cases since this <code>LoginModule</code> si
56.      * @exception FailedLoginException if the authentication fails. <p>
57.      * @exception LoginException if this <code>LoginModule</code> is ur
58.      * @see javax.security.auth.spi.LoginModule#login()
59.      */
60.     public boolean login() throws LoginException {
61.         Callback[] callbacks = null;
62.         String      pwd      = null;
63.
64.         // prompt for a user name and password
65.         if (callbackHandler == null){
66.             throw new LoginException("Es necesario que especifique un CallbackHandle
67.         }
68.
69.         try {
70.             callbacks = new Callback[2];
71.             callbacks[0] = new NameCallback("Introduzca su nombre de usuario: ");
72.             callbacks[1] = new PasswordCallback("Introduzca su contraseña: ", false)
73.
74.             // Invocamos al CallbackHandler
75.             callbackHandler.handle(callbacks);
76.
77.             // Obtenemos los datos introducidos por el usuario
78.             username = ((NameCallback) callbacks[0]).getName();
79.             password = ((PasswordCallback) callbacks[1]).getPassword();

```

Implementación de javax.security.auth.callback.CallbackHandler: Solicitando al usuario su login y el contraseña.

A continuación, implementamos un CallbackHandler que pide el nombre de usuario y la contraseña al usuario.

Existen multitud de implementaciones disponibles, como por ejemplo, que nos muestre una ventana Swing GUI para pedir la información requerida en la autenticación.

```
view plain print ?
01. package com.autentia.tutoriales.jaas.jaas;
02.
03. import java.io.*;
04. import javax.security.auth.callback.*;
05.
06. /**
07.  * Solicita al usuario que ejecuta la aplicación que se autentifique por
08.  * @author Carlos García. http://www.autentia.com
09.  * @see http://www.mobiletest.es
10.  */
11. public class PromptCallbackHandler implements javax.security.auth.callback.CallbackHandler {
12.     /**
13.      * @see javax.security.auth.callback.CallbackHandler#handle(javax.security.auth.callback.Callback[])
14.      */
15.     public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
16.         for (int i = 0; i < callbacks.length; i++) {
17.             if (callbacks[i] instanceof TextOutputCallback) {
18.                 // El Login Module envia algún tipo de información al usuario
19.
20.                 TextOutputCallback toc = (TextOutputCallback) callbacks[i];
21.                 System.out.println(toc.getMessageType()); // INFORMATION, ERROR
22.                 System.out.println(toc.getMessage());
23.
24.             } else if (callbacks[i] instanceof NameCallback) {
25.                 // El LoginModule pregunta el login del usuario
26.
27.                 NameCallback nc = (NameCallback) callbacks[i];
28.                 System.out.print(nc.getPrompt());
29.                 nc.setName((new BufferedReader(new InputStreamReader(System.in))).readLine());
30.
31.             } else if (callbacks[i] instanceof PasswordCallback) {
32.                 // El LoginModule pregunta el password del usuario
33.
34.                 PasswordCallback pc = (PasswordCallback) callbacks[i];
35.                 System.out.print(pc.getPrompt());
36.                 String pwd = (new BufferedReader(new InputStreamReader(System.in))).readLine();
37.                 pc.setPassword(pwd.toCharArray());
38.
39.             } else {
40.                 throw new UnsupportedCallbackException(callbacks[i], "Callback desconocido");
41.             }
42.         }
43.     }
44. }
45.
46.
```

Implementación de java.security.Principal: Representando la identidad de los usuarios.

Ahora definimos el java.security.Principal en base a la cual se especificaran los [permisos de los usuarios](#).

```
01. package com.autentia.tutoriales.jaas.jaas;
02.
03. /**
04.  * Representa la identidad de un usuario.
05.  * Las acciones a autorizar se basan en si posee o no cierto Principal
06.  * @author Carlos García. Autentia Real Business Solutions
07.  * @see http://www.mobiletest.es
08.  */
09. public class SimplePrincipal implements java.security.Principal, java.io.Serializable
10.     private String name;
11.
12.     public SimplePrincipal(String name) {
13.         this.name = name;
14.     }
15.
16.     public String getName() {
17.         return name;
18.     }
19.
20.     /**
21.      * @see java.lang.Object#equals(java.lang.Object)
22.      */
23.     public boolean equals(Object o) {
24.         try {
25.             return ((SimplePrincipal) o).name.equalsIgnoreCase(name);
26.         } catch (Exception ex){
27.             return false;
28.         }
29.     }
30. }
31.
```

Referencias

- [Java SE Security](#)

Conclusiones

Bueno, espero que tras la lectura de este tutorial tenga un poco más claro el tema de la seguridad Java, tema que desde mi punto de vista no es facil de comprender debido a las dependencias de conceptos, aquí hemos visto bastantes juntos y puede ser un tutorial bueno para aclarar ideas.

Recordaros, que en [Autentia](#) estamos muy acostumbrados a usar estándares, entre ellos aquellos relacionados con la seguridad.

Un saludo.

Carlos García. Creador de [MobileTest](#), un complemento educativo para los profesores y sus alumnos.

- Puedes opinar sobre este tutorial haciendo clic aquí.
- Puedes firmar en nuestro libro de visitas haciendo clic aquí.
- Puedes asociarte al grupo AdictosAITrabajo en XING haciendo clic aquí.
- Añadir a favoritos Technorati. 



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

Recuerda

[Autentia](#) te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#)). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

¿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?

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com

soluciones reales para su negocio

Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de subscripción a novedades:

E-mail

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	pdf
Como configurar Tomcat + IIS	Os mostramos como activar el filtro ISAPI de IIS para conectarlo con Tomcat. De este modo el servidor Web IIS 5.5 de Microsoft servirá las páginas estáticas y Tomcat los JSPs y Servlets dinámicos	2005-12-11	13789	pdf
XML Signature - Firma Digital sobre XML	En este tutorial aprenderemos a firmar digitalmente y validar un documento utilizando la implementación de Apache.	2008-04-03	2324	pdf
Crap4j, ¿es tu código difícilmente mantenible?	En este tutorial os presentamos Crap4j, un plugin de eclipse que trata de localizar ese código con el que todos nos hemos encontrado alguna vez al auditar, mantener o continuar con un desarrollo y que resulta tan difícil de mantener...	2008-01-20	1477	pdf
XML Encryption, Criptografía sobre XML	Carlos García describe en este tutorial la encriptación de secciones de documentos XML utilizando el lenguaje del W3C: XML Encryption	2008-04-03	1606	pdf
EJB 3.0: Resurrection	Este tutorial nos va a presentar las nuevas funcionalidades que nos aportan los EJB 3.0.	2007-05-07	6378	pdf
Creación de una aplicación web con SpringMVC desde 0	Este tutorial te resultará muy útil para aprender a usar el patrón modelo-vista-controlador (MVC) con Spring a nuestros desarrollos web	2008-05-05	3629	pdf
Control navegación en Servlets	Os mostramos como construir el esqueleto de una aplicación basada en Servlets y JSP, con control de navegación.	2003-07-08	29836	pdf
Protege tu PC	Os mostramos como proteger tu máquina de ataques mientras estés conectado a una red o Internet	2003-07-03	14112	pdf
Certificados en IIS para activación SSL	En este tutorial vamos a habilitar el soporte SSL (Secure Socket Layer, comunicación segura por https) en un servidor IIS (Internet Information Server de Microsoft).	2005-09-09	8495	pdf
Spring: definición dinámica de Beans	Este tutorial habla sobre la modificación dinámica de los beans del contexto para simplificar la configuración de Spring	2007-05-09	4978	pdf

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.