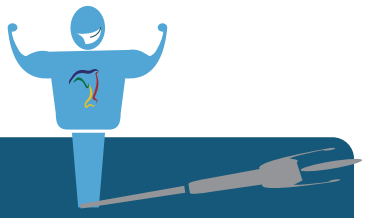
autentia

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...

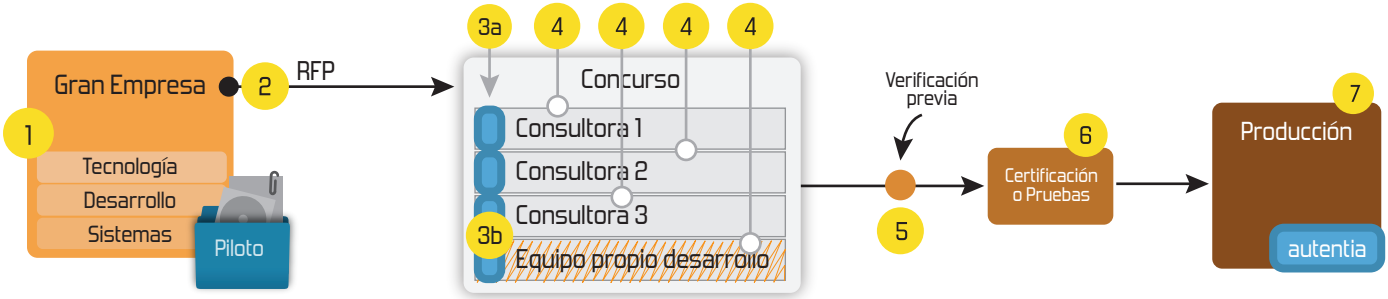
1. Desarrollo de componentes y proyectos a medida



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

DESARROLLADO POR:
Rubén Aguilera Díaz-Heredero

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Fecha de publicación del tutorial: 2011-06-21

Implementando SSO con CAS: ejemplo práctico

0. Índice de contenidos.

1. Entorno

Este tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Mac Book Pro 17" (2,6 Ghz Intel Core i7, 8 GB DDR3)
- Sistema Operativo: Mac OS X Snow Leopard 10.6.4
- Spring MVC Portlet 3.0.4
- Maven 2.2.1
- Eclipse 3.6 (Helios) con M2Eclipse
- CAS Server 3.4.2.1
- CAS Client 3.2.0

2. Introducción

Ya vimos hace tiempo las bondades de CAS en el tutorial Introducción a CAS donde veíamos también como se instalaba y su funcionamiento más básico.

En esta oportunidad vamos a ver cómo podemos securizar dos aplicaciones web distintas con CAS, permitiendo que al logarnos en una el acceso a la otra no solicite nuevamente las credenciales.

3. Creación de las aplicaciones web a securizar

Lo primero que vamos a hacer es crear dos aplicaciones web muy sencillitas gracias a la ayuda de Maven. Para ello como de costumbre escribimos en un terminal mvn archetype:generate, seleccionamos el arquetipo web (TIP: suele estar 3 más abajo del de por defecto, es decir, si por defecto sale el 109 pues suele ser el 112, pero mejor comprobadlo) e introducimos los siguientes datos para la primera aplicación:

- Versión del arquetipo: 1.0
- groupId: com.autentia
- artifactId: cas-example-1
- versión: 1.0-SNAPSHOT
- package: com.autentia

Y lo mismo para la segunda pero cambiando el artifactId por cas-example-2.

Ahora importamos los proyectos a Eclipse gracias al plugin de m2Eclipse, y simplemente editamos el fichero index.jsp que se crea por defecto, para poner un texto que diga "Esto es cas-example-1" en el caso de la primera aplicación y "Esto es cas-example-2" en el caso de la segunda aplicación, a fin de distinguirlos cuando los despleguemos. Además cada uno de ellos va a llevar una sentencia que recupera el usuario logado de la request, este es el contenido para el primer ejemplo:

```
view plain  print  ?
01. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
02.     pageEncoding="ISO-8859-1"%>
03. <!DOCTYPE html PUBLIC "-
    //W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
04. <html>
05. <head>
06. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
07. <title>Cas Example 1
08. </head>
09. <body>
10.
11. <p>Esto es el ejemplo de CAS número 1</p>
12.
13. <p>Este es el usuario logado: <%=request.getRemoteUser() %></p>
14.
15. </body>
16. </html>
```

Venga ahora vosotros el ejemplo 2.

4. Configuración del servidor de CAS para permitir el SSO

Una de las primeras cosas que deben quedar claro en la implementación de un SSO con CAS es que sólo puede funcionar a través de un canal seguro, es decir, que tendremos que configurar nuestro Tomcat para habilitar el protocolo HTTPS.

Para ello antes de nada necesitamos un certificado. Para este tutorial nos valdrá con un "dummy" el cual vamos a crear abriendo un terminal y siguiendo los siguientes pasos:

1. Situarnos en el directorio JAVA_HOME/bin
2. Generar la key de nuestro certificado, para ello ejecutar: sudo keytool -genkey -alias tomcat -keyalg RSA

   NOTA: La keypass changeit es la de por defecto para Mac

   Esto hace que el sistema pregunta una serie de datos:

   - Enter keystore password: changeit (Válido para Mac)
   - What is your first and last name?: raguilera.com (Aquí lo que pide realmente es un nombre de dominio, no pongais vuestro nombre y dos apellidos porque va a ser un dominio muy largo ;-). Tampoco es aconsejable poner localhost, siempre podéis mapear un nombre de dominio con la dirección 127.0.0.1 en el fichero /etc/hosts de vuestras máquinas).
   - What is the name of your organizational unit?: Desarrollo (Se refiere al departamento)
   - What is the name of your organization? Autentia (Se refiere al nombre de la empresa)
   - What is the name of your City or Locality?: Alcalá de Henares (Capital del imperio español cuando no se ponía el sol)
   - What is the name of your State or Province?: Madrid
   - What is the two-letter country code for this unit?: ES
   - En este momento aparece un resumen con los datos introducidos al que tendremos que responder afirmativamente para que la creación de la key tenga éxito.
3. Exportar el certificado a un fichero, para ello ejecutamos: sudo keytool -export -alias tomcat -file server_cas.crt. Nos pedirá la contraseña que es changeit y si todo es correcto nos responderá afirmativamente a la creación del fichero.
4. Importamos el certificado al almacén de certificados de Java, asegurándonos de que estamos situados en la versión de Java que está utilizando nuestro Tomcat. Para ello ejecutamos: sudo keytool -import -alias tomcat -file server_cas.crt -keystore ../lib/security/cacerts. Nos pregunta si queremos confiar en este certificado, le decimos que si y ya está.

El siguiente paso es editar el fichero TOMCAT_HOME/conf/server.xml para descomentar las líneas que se refieren al HTTPS y dejarlas de esta forma:

```
view plain  print  ?
01. <connector port="8443" maxthreads="200" scheme="https" secure="true" sslenabled="true" keystoref
02. </connector>
```

Lo más importante de esta configuración es definir el puerto y la localización del fichero .keystore, que en general se crea como un archivo oculto dentro de la carpeta del usuario.

5. Configuración de las aplicaciones web

La única configuración que requiere CAS es editar el fichero web.xml para incluir los siguientes filtros:

Portlet (III): Añadiendo validación al formulario

```
view plain  print  ?
01.  <filter>
02.          <filter-name>CAS Authentication Filter</filter-name>
03.          <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
04.          <init-param>
05.              <param-name>casServerLoginUrl</param-name>
06.              <param-value>https://raguilera.com:8443/cas/login</param-value>
07.          </init-param>
08.          <init-param>
09.              <param-name>service</param-name>
10.              <param-value>http://raguilera.com:8080/cas-example-1</param-value>
11.          </init-param>
12.      </filter>
13.
14.      <filter>
15.          <filter-name>CAS Validation Filter</filter-name>
16.          <filter-
     class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-
     class>
17.          <init-param>
18.              <param-name>casServerUrlPrefix</param-name>
19.              <param-value>https://raguilera.com:8443/cas/</param-value>
20.          </init-param>
21.          <init-param>
22.              <param-name>service</param-name>
23.              <param-value>http://raguilera.com:8080/cas-example-1</param-value>
24.          </init-param>
25.      </filter>
26.
27.      <filter>
28.          <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
29.          <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-
     class>
30.      </filter>
31.
32.      <filter-mapping>
33.          <filter-name>CAS Authentication Filter</filter-name>
34.          <url-pattern>/*</url-pattern>
35.      </filter-mapping>
36.
37.      <filter-mapping>
38.          <filter-name>CAS Validation Filter</filter-name>
39.          <url-pattern>/*</url-pattern>
40.      </filter-mapping>
41.
42.      <filter-mapping>
43.          <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
44.          <url-pattern>/*</url-pattern>
45.      </filter-mapping>
```

Esta es la explicación de cada uno de los filtros:

- **CAS Authentication Filter:** se encarga de detectar si el usuario está logado, en caso contrario redirecciona el flujo a la URL especificada en el parámetro "casServerLoginUrl". En el parámetro "service" le indicamos la URL donde queremos redireccionar el flujo una vez el usuario haya sido correctamente autenticado por CAS. En resumen, este filtro le asigna un ticket al usuario correctamente logado.
- **CAS Validation Filter:** se encarga de validar que el ticket que aporta el usuario es válido es decir que lo tiene asignado y no ha expirado. Para ello requiere que especifiquemos el sufijo por el que empieza nuestro CAS en el parámetro "casServerUrlPrefix" y le indiquemos el mismo "service" que en la autenticación si no queremos que nos de un error de que el ticket proporcionado no corresponde con el servicio.
- **CAS HttpServletRequest Wrapper Filter:** una vez el ticket es validado este filtro se encarga de "actualizar" la request con los datos de usuario logado para que podamos recuperarlos interrogando a la request con la sentencia request.getRemoteUser()

Cada aplicación tiene que hacer referencia a las dependencias cas-client-core-3.2.0.jar y commons-logging-1.1.1.jar o si no introducidlas dentro de la carpeta TOMCAT_HOME/lib.

6. Probando el resultado

En mi caso las dos aplicaciones van a correr en el mismo Tomcat que el servidor de CAS, por lo que para probar el resultado tenemos que desplegar las dos aplicaciones en este servidor y arrancarlo.

Lo siguiente será abrir un navegador y poner la siguiente URL: http://raguilera.com:8080/cas-example-1. Tendremos que ver como haciendo esto el navegador nos redirecciona a la URL de login de CAS para que introduzcamos unas credenciales correctas, por defecto que el usuario y la password sean iguales.

Cuando nos hayamos logado correctamente veremos que el navegador nos redirecciona a la URL que le hemos indicado en "service" y que nos muestra el contenido y el nombre del usuario que está logado.

Ahora en el mismo navegador (o en otro pestaña del mismo navegador) abrimos la URL: http://raguilera.com:8080/cas-example-2 y tendremos que ver que sin logarnos aparece el contenido con el nombre del usuario logado.

Podéis probar a hacerlo al revés, pero no sin antes eliminar la cache y las cookies del navegador.

7. Conclusiones

A mi este tipo de herramientas "satelites" me encantan, te permiten modularizar tu aplicación sin tener que cambiar ni una línea de código sólo a través de filtros o listener de Tomcat. En este caso esta herramienta nos resuelve el problema concreto de tener que implementar un SSO para todas nuestras aplicaciones, e incluso centralizar la seguridad tanto de aplicaciones web como de aplicaciones de escritorio. Ya os digo yo que los mayores problemas para seguir este tutorial los vais a encontrar a la hora de crear el certificado.

Cualquier duda o sugerencia en la zona de comentarios.

Saludos.

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

Comentarios