trivadis
Part of Accenture

Liferay und CAS

Einmal angemeldet und es geht los.

) Schulung)

) Beratung)

) Entwicklung)

) Artikel)

AUTOR

Benjamin Bratkus
Orientation in Objects GmbH

Veröffentlicht am: 1.10.2007

ABSTRACT

Im Zeitalter der Webanwendungen ist es dem Anwender ein Bedürfnis, seine Benutzerinformationen und seine Identität sicher zu halten.

Dennoch liegt es in der Natur des Menschen einen möglichst einfachen und bequemen Mechanismus zu finden, um die gewünschte Sicherheit mit einem möglichst geringem Maß an Aufwand zu erhalten.

In diesem Zusammenhang kommt der sogenannte CAS (Central Authentification Service) zum tragen. CAS bietet die Möglichkeit, sich an verschiedenste Benutzerdaten Bestände zu binden, um die Erkennung und die Identität des Benutzers zu gewährleisten.

Trivadis Germany GmbH

Weinheimer Str. 68
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.dekontakt@trivadis.com

EINLEITUNG

Der Mechanismus des Single Sign On ist keine Weltneuheit. Das Besondere an einem CAS ist, dass die CAS Anwendung die erfolgreiche Authentifizierung eines Benutzers sogenannte "Tickets" an die mit Ihr verbundenen Anwendungen melden kann.

So kann man mit Hilfe von CAS ein Single Sign On Szenario einfach realisieren.

Das Open Source Portal Liferay 4.2 [1] mit einer Menge an heterogenen integrierten Anwendungen unterstützt CAS. Es kann durch Konfiguration des Clients an die CAS Server Anwendung angebunden werden. Die Integration eines Portals wie Liferay 4.2 erweist sich als sinnvoll, da es neben dem Portal, das beispielsweise als zentrale Intranet Instanz genutzt wird, weitere Webapplikationen innerhalb der selben Domäne gibt, die aufgrund historischer oder technologischer Beweggründe nicht in das Portal integriert wurden, aber dennoch CAS unterstützen können. So könnte aus Sicht der Firma der Domänen Benutzer nun für CAS genutzt werden, um dem Mitarbeiter mit einer einzigen Anmeldung sofort Zugriff auf seine persönlichen Ressourcen zu beschaffen. Dieser Artikel stellt ein Beispiel zur Umsetzung des eben umschriebenen Szenarios dar und soll einen Überblick der Aufwände, die mit dieser Konfiguration verbunden sind, bringen.

CAS HISTORIE

Der Central Authentication Service wurde ursprünglich an der University of Yale entwickelt. Durch das grosse Interesse an diesem Mechanismus, wird CAS jetzt von der von der Community Java Architectures Special Interest Group (JA-SIG) als Open Source Projekt weitergeführt.[2]

GRUNDLAGEN DES SINGLE SIGN ON

Um eine serverseitige SSO Lösung zu realisieren sind verschiedene Lösungsansätze verfügbar. Zwei der möglichen Lösungsansätze kommen bei CAS zum Tragen. Zum einen handelt es sich um eine Ticket basierte Umsetzung von Single Sign On und zum anderen um einen zentralen Anmeldungsserver, der die mit diesem verbunden Anwendungen über die erfolgreiche Authentifizierung des Benutzers, unter Nutzung der Tickets, informiert.[3]

UMSETZUNG

Die Implementierung von CAS ist eine Java Servlet Anwendung. Das Servlet nutzt hier zwei markante URLs: Die Login- und die Validierungs-URL. Die Funktionsweise der Anwendung ist mit einem zentralen Anmeldungsserver in Kombination mit einem Ticket Server vergleichbar. Um CAS wie in diesem Beispiel beschrieben nutzen zu können, benötigt man als ersten Schritt die Server Anwendung des CAS Systems (hier Version 3.0.5). Diese Anwendung muss, abhängig von der Art und Weise der Benutzerverwaltung, erweitert werden. Für das konkrete Szenario werden die Benutzer in einem Active Directory verwaltet. Hinzu kommt die Konfiguration des Servlet Containers, der die Laufzeitumgebung für das CAS System bildet.

KONFIGURATION UND ERSTELLUNG DER CAS ANWENDUNG

Für den erfolgreichen Start der CAS Anwendung mit Anbindung an den Active Directory oder LDAP, muss die Anwendung um zwei zusätzliche Bibliotheken erweitert werden. Zum einen benötigt die Server Version 3.0.5, das namentlich nahezu gleiche "cas-server-ldap-3.0.5.jar" und zum anderen das "ldaptemplate-1.0.2.jar". Beide Bibliotheken müssen nach dem erfolgreichen Herunterladen der Server Anwendung und der eben genannten Bibliotheken, in dem Server in das Verzeichnis "localPlugins\lib" eingefügt werden. Anschliessend muss die "${project.home}/webapp/WEB-INF/deployerConfigContext.xml" für die Anbindung an das LDAP konfiguriert werden. Hier stehen von Hause aus zwei Mechanismen für die Anbindung an das LDAP zur Verfügung.

- LDAP fastbind authentication: Diese Authentifizierung des angegebenen Benutzers, ist die einfache Art und Weise, sich mit dem LDAP zu verbinden. Dem Anwender dieser Methode muss die Struktur des LDAP Servers klar beziehungsweise bekannt sein. Das bedeutet er muss wissen, wo die Benutzer der Domäne in dem LDAP Server Baum liegen, um den Erfolg dieser Methode zu gewährleisten.
- LDAP search-and-bind authentication: Diese Methode sucht nach dem Benutzer und Passwort Paar, welches der Anwender über die Benutzeroberfläche eingegeben hat. Hilfreich natürlich, wenn die Struktur des LDAP unzureichend bekannt ist, jedoch erfordert diese Methode genau einen zusätzlichen LDAP Benutzer der die Berechtigung zu suchen besitzt oder das LDAP oder Active Directory muss so konfiguriert sein, dass es anonym durchsuchbar ist.

In diesem Beispiel wird die "LDAP search-and-bind authentication" Methode verwendet.

```
// Erweiterung des AuthenticationMangers
// um den passsenden Authentication Handler
.....
<property name="authenticationHandlers">
  <list>
   <bean class="org.jasig.cas.authentication.handler.support.
       HttpBasedServiceCredentialsAuthenticationHandler" />
   <bean class="org.jasig.cas.adaptors.ldap.
               BindLdapAuthenticationHandler" >
    <property name="filter" value="sAMAccountName=%u" />
    <property name="searchBase"
              value="CN=Users,DC=schulung, DC=oio,DC=de" />
    <property name="contextSource" ref="contextSource" />
   </bean>
  </list>
</property>
........
//Eintrag für die referenzierte Context Source
<bean id="contextSource"
    class="org.jasig.cas.adaptors.ldap.
          util.AuthenticatedLdapContextSource">
  <property name="authenticatedReadOnly" value="true" />
  <property name="userName"
         value="CN=portadmin,CN=Users,
         DC=schulung,DC=oio,DC=de" />
  <property name="password" value="testsystem" />
  <property name="urls">
   <list>
    <value>ldap://testsystem.schulung.oio.de/</value>
    <value>ldap://testsystem.schulung.oio.de/</value>
   </list>
  </property>
  <property name="baseEnvironmentProperties">
   <map>
    <entry>
     <key>
      <value>java.naming.security.authentication</value>
     </key>
     <value>simple</value>
    </entry>
   </map>
  </property>
</bean>
```

Beispiel 1: Datei "deployerConfigContext.xml"

Nach der Erweiterung des CAS Systems muss dieses nun mit Apache Maven1 [4] gebaut werden und das erstellte Webarchiv in dem selben Tomcat, in dem das Portal läuft integriert werden.

SSL ZERTIFIKATSERZEUGUNG

Für den erfolgreichen Einsatz einer SSL Verbindung mit dem Apache Tomcat muss noch ein SSL Zertifikat erzeugt werden. Hierzu gibt es verschiedene Möglichkeiten ein solches zu bekommen. Für die Entwicklerversion in unserem Beispiel, wird das Java Keytool genutzt, das standardmässig mit Java geliefert wird. [5]

```
keytool -genkey -alias tomcat -keypass changeit -keyalg RSA
```

Beispiel 2: Erzeugung des SSL Zertifikats

Nach dem Kommando zur Zertifikatserzeugung muss der Dialog ausgefüllt werden, wichtig an dieser Stelle ist, dass als "first and last name" ein valider Rechnername(Host) eingegeben wird.

```
Enter keystore password: changeit
What is your first and last name?
[Unknown]: localhost
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]:
What is the name of your City or Locality?
[Unknown]:
What is the name of your State or Province?
[Unknown]:
What is the two-letter country code for this unit?
[Unknown]:
Is CN=localhost, OU=Unknown, O=Unknown, L=Unknown,
  ST=Unknown, C=Unknown correct?
[no]: yes
```

Beispiel 3: Dialog zur Zertifikatserzeugung

BEREITSTELLUNG DES ERZEUGTEN ZERTIFIKATS FÜR DEN SCHLÜSSELBUND DES BENUTZERS

Um das eben erzeugte Zertifikat sinnvoll nutzen zu können, muss es zuerst in eine Datei (%FILENAME% als Platzhalter, z.b. server.cert) exportiert und anschliessend in den Schlüsselbund importiert werden.

```
// Export
keytool -export -alias tomcat -keypass changeit -file
%FILE_NAME%
// Import
keytool -import -alias tomcat -file %FILE_NAME%
        -keypass changeit
        -keystore %JAVA_HOME%/jre/lib/security/cacerts
```

Beispiel 4: Bereitstellung des erzeugten Zertifikats für den Schlüsselbund

KONFIGURATION DES TOMCAT SERVLET CONTAINERS

Die Konnektoren des Tomcat müssen angepasst werden.

```
<Connector port="8443" maxHttpHeaderSize="8192"
           maxThreads="150" minSpareThreads="25"
           maxSpareThreads="75" enableLookups="false"
           disableUploadTimeout="true" acceptCount="100"
           scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS" />
```

Beispiel 5: Datei "server.xml" des Tomcat

Für eine produktive Umsetzung sollten aus Sicherheitsaspekten das CAS System und die Server, die die Laufzeitumgebungen der Anwendungen sind, getrennt sein.

EINBINDUNG DES CAS CLIENT IN LIFERAY

Anschliessend müssen die Anwendungen, in diesem Beispiel das Portal, mit dem Client der zur Interaktion mit dem CAS Server System benötigt wird, ausgestattet werden. Hierzu muss im Fall von Liferay die Client Bibliothek in das entsprechende Verzeichnis für Bibliotheken "ROOT/web-inf/lib" eingefügt und anschliessend der Server neu gestartet werden. In diesem Beispiel wird die Version 2.0.11 des Clients benötigt.

ANBINDUNG VON LIFERAY AN DAS ACTIVE DIRECTORY

Zur Authentifizierung muss neben der CAS Anwendung ebenso auch das Portal System so konfiguriert werden, dass es die Benutzer gegen das gleiche Active Directory oder LDAP System authentifiziert. Hierzu können zum einen die Administrative Benutzeroberfläche oder zum anderen die entsprechenden Konfigurationsdateien verwendet werden. Der Einfachheit halber wird hier nun die Konfiguration über die Benutzeroberfläche gezeigt. Die Angaben, die hier getätigt werden müssen, entsprechen den Angaben der CAS Anwendung.[6]

Abbildung 1: Administrative Portlet Anwendung zur Anbindung von Liferay an das LDAP oder Active Directory

KONFIGURATION VON LIFERAY FÜR CAS

Nachdem der CAS Client in das Verzeichnis für Bibliotheken von Liferay eingebunden wurde, kann nun die web.xml des Portals angepasst werden, so dass der CAS Server genutzt wird. Der Login Servlet Filter wird bereits von Liferay bereitgestellt (ab Version 4.2), muss aber noch aktiviert werden. An dieser Stelle muss noch einmal gesagt werden, dass die verwendeten Konfigurationen oder Einstellungen nicht für den produktiven Einsatz verwendet werden sollten.

```
<filter>
  <filter-name>CAS Filter</filter-name>
  <filter-class>
    edu.yale.its.tp.cas.client.filter.CASFilter
  </filter-class>
  <init-param>
    <param-name>
      edu.yale.its.tp.cas.client.filter.loginUrl
    </param-name>
    <param-value>
      https://localhost:8443/cas-web/login
    </param-value>
  </init-param>
  <init-param>
    <param-name>
      edu.yale.its.tp.cas.client.filter.validateUrl
    </param-name>
    <param-value>
      https://localhost:8443/cas-web/proxyValidate
    </param-value>
  </init-param>
  <init-param>
    <param-name>
      edu.yale.its.tp.cas.client.filter.serviceUrl
    </param-name>
    <param-value>
      http://localhost:8080/c/portal/login
    </param-value>
  </init-param>
</filter>
```

Beispiel 6: Erweiterungen der web.xml von Liferay

Anschliessend müssen für die Version 4.2 von Liferay noch zwei weitere Konfigurationen umgesetzt werden. Zum einen muss die "system-ext.properties" und zum anderen die "portal-ext.properties" erweitert werden.

```
//system-ext.properties -
// der CAS Filter muss aktiviert werden
com.liferay.filters.sso.cas.CASFilter=true
//portal-ext.properties -
// der Autologin muss auf den CAS Login gesetzt werden
auto.login.hooks=
   com.liferay.portal.security.auth.BasicAutoLogin,
   com.liferay.portal.security.auth.CASAutoLogin
```

Beispiel 7: Liferay Konfigurationsdateien

Es geht los! Beide Anwendungen sind nun aufeinander abgestimmt und der Benutzer kann sich über das Login des CAS an den Anwendungen anmelden.

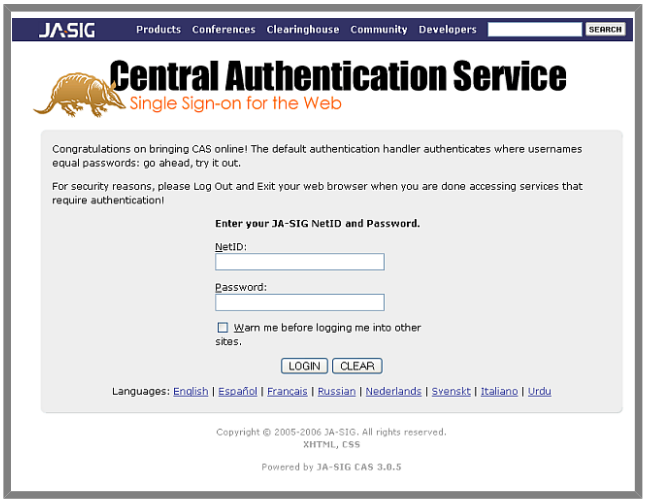

Abbildung 2: Anmeldungsmaske von CAS

ZUSAMMENFASSUNG

Durch die Verbindung von CAS mit weiteren Anwendungen, die eine Zusammengehörigkeit aufweisen, kann dem Benutzer das Arbeiten mit diesen Anwendungen erleichtert werden. Natürlich ist eine derartige Zentralisierung von sicherheitsrelevaten Daten ein heisses Thema und muss gut überlegt sein. Neben dem Portal, das in diesem Beispiel an das CAS System angebunden wurde, können viele weitere Open Source Anwendungen auf diesen Mechanismus zugreifen und damit arbeiten, so dass auch CAS in die Strecke Ihrer Open Source Produkte und Projekte passt. Ausserdem sind für viele Programmiersprachen die Client Bibliotheken vorhanden, so dass auch Anwendungen, die beispielsweise mit PHP entwickelt wurden, in dieses Szenario integriert werden können. Das CAS System und das Single Sign On sind in jedem Fall eine Bereicherung im Bereich Open Source und Benutzerauthentifizierung.

REFERENZEN

- [1] Liferay Portal
  (http://www.Liferay.com)
- [2] JA SIG
  (http://www.ja-sig.org/)
- [3] SSO frei Haus
  Rummeyer, Oliver ; Düsterhaus, Jörg
  (http://entwickler.com/itr/online_artikel/show.php3?nodeid=97&id=843)
- [4] Apache Maven
  (http://maven.apache.org)
- [5] Liferay Wiki CAS
  (http://wiki.liferay.com/index.php/Single_SignOn_-_Integrating_Liferay_With_CAS_Server)
- [6] Liferay Wiki LDAP
  (http://wiki.liferay.com/index.php/LDAP)