**浙江港航综合管理与服务平台**

**——SSO单点登录**

1. **前言**

SSO统一用户认证系统是基于Web Service技术，提供了浙江省港航管理局综合管理与信息服务平台的用户统一管理、统一认证、异构系统与平台单点登录、统一安全控制与认证的系统平台。通过该系统，可以圆满解决港航系统内信息孤岛问题、应用认证集成等问题，真正实现：一点认证、全网通行；一点管理，全网安全。

1. **SSO实现**

**2.1、实现方法**

项目采用的SSO实现方法是Yelu大学研发的CAS(Central Authentication Server)，也是目前企业中运用较多的开源SSO服务器。

**2.2、实现环境**

* Tomcat-7.0.59
* JDK1.7
* CAS Server版本：cas-server-3.5.2 Release
* CAS Client版本：cas-client-3.2.1 Release
* 操作系统：windows7 64位

1. **SSO客户端配置**

CAS服务端采用舟山部署完成的CAS Server，本地项目针对web.xml进行配置，在web.xml里增加四个过滤器：

1. 配置用户认证工作过滤器

<!-- 该过滤器负责用户的认证工作，必须启用它 -->

<filter>

<filter-name>CASFilter</filter-name>

<filter-class>

org.jasig.cas.client.authentication.AuthenticationFilter

</filter-class>

<init-param>

<param-name>casServerLoginUrl</param-name>

<!--这里的server是服务端的IP -->

<param-value>

http://220.189.207.21:8290/casEG/login

</param-value>

</init-param>

<init-param>

<param-name>serverName</param-name>

<!--这里的server是本地项目的 一旦本地访问以下url，过滤器捕捉到，随即跳转到服务端的url-->

<param-value>http://localhost:8080</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CASFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

1. 配置Ticket校验过滤器

<filter>

<filter-name>CAS Validation Filter</filter-name>

<filter-class>

org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter

</filter-class>

<init-param>

<param-name>casServerUrlPrefix</param-name>

<!--这里的server是服务端的IP -->

<param-value>http://220.189.207.21:8290/casEG</param-value>

</init-param>

<init-param>

<param-name>serverName</param-name>

<param-value>http://localhost:8080</param-value>

</init-param>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CAS Validation Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

1. 配置实现HttpServletRequest请求的包裹过滤器

<filter>

<filter-name>CAS HttpServletRequest Wrapper Filter

</filter-name>

<filter-class> org.jasig.cas.client.util.HttpServletRequestWrapperFilter

</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS HttpServletRequest Wrapper Filter

</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

1. 配置开发者可以通过rg.jasig.cas.client.util.AssertionHolder来获取用户的登录名的过滤器

<filter>

<filter-name>CAS Assertion Thread Local Filter

</filter-name>

<filter-class>

org.jasig.cas.client.util.AssertionThreadLocalFilter

</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS Assertion Thread Local Filter

</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

启动服务后，当过滤器检测到http://localhost:8080的url时，判断当前是否登录状态，若没有登录，则会跳转到CAS 服务端的地址http://220.189.207.21:8290/casEG/login

**附录:**

**cas服务端以及cas客户端的demo**

//================================================================================

// 软件环境

//================================================================================

1.操作系统： windows7 64位 旗舰版 SP1

2.CAS版本： CAS Server 3.5.2 Release

3.TOMCAT

●版本： apache-tomcat-7.0.47

●认证中心安装路径/http端口号/https端口号： D:\cas\_sso\apache-tomcat-7.0.47-CasServer / 屏蔽 / 8444 // 由于本机已经启动了一个tomcat，所以8443改为8444

●客户端1安装路径/http端口号/https端口号： D:\cas\_sso\apache-tomcat-7.0.47-CasApp1

●客户端2安装路径/http端口号/https端口号： D:\cas\_sso\apache-tomcat-7.0.47-CasApp2

4.JDK

●版本：

java version "1.7.0\_25"

Java(TM) SE Runtime Environment (build 1.7.0\_25-b17)

Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)

●部署路径： D:\jdk1.7.0\_25

5.本机IP： 127.0.0.1

6.测试域名：

●认证中心： center.mysso.com 对应部署cas server的tomcat，这个虚拟域名还用于证书生成

●客户端1： app1.mysso.com 对应部署app1 的tomcat

●客户端2： app2.mysso.com 对应部署app2 的tomcat

7.JAVA\_HOME： D:\jdk1.7.0\_25

8.工作目录： D:\cas\_sso\

//================================================================================

//================================================================================

// 步骤1：设置本地域名

//================================================================================

编辑C:\Windows\System32\drivers\etc\hosts文件，增加如下3条（演示系统用修改hosts 文件的方法添加域名最简单方便）

127.0.0.1 center.mysso.com

127.0.0.1 app1.mysso.com

127.0.0.1 app2.mysso.com

●认证中心： center.mysso.com 对应部署cas server的tomcat，这个虚拟域名还用于证书生成

●客户端1： app1.mysso.com 对应部署app1 的tomcat

●客户端2： app2.mysso.com 对应部署app2 的tomcat

//================================================================================

//================================================================================

// 步骤2：确认JAVA\_HOME

//================================================================================

在本机环境变量中设置：JAVA\_HOME=D:\jdk1.7.0\_25

//================================================================================

//================================================================================

// 步骤3：安全证书配置

//================================================================================

1.生成证书（自签名根证书）

●证书别名： mysso

●访问密码： myssopwd

●keystore文件名： D:\cas\_sso\ca\mysso.keystore

●keystore访问密码： myssopwd

命令行：（keypass 和 storepass 两个密码要一致，否则下面tomcat配置https访问失败）

keytool -genkey -alias mysso -keyalg RSA -keysize 1024 -keypass myssopwd -validity 365 -keystore D:\cas\_sso\ca\mysso.keystore -storepass myssopwd

执行过程及输出：（注意提示输入要符合事先约定好的域名，详见注释）

D:\cas\_sso>keytool -genkey -alias mysso -keyalg RSA -keysize 1024 -keypass myssopwd -validity 365 -keystore D:\cas\_sso\ca\mysso.keystore -storepass myssopwd

您的名字与姓氏是什么？

[Unknown]： center.mysso.com // 输入的姓名和上面hosts文件中配置的一致（即：center.mysso.com，也可用主机名）

您的组织单位名称是什么？

[Unknown]： mysso.com

您的组织名称是什么？

[Unknown]： mysso

您所在的城市或区域名称是什么？

[Unknown]： SH

您所在的州或省份名称是什么？

[Unknown]： SH

该单位的两字母国家代码是什么

[Unknown]： CN

CN=center.mysso.com, OU=mysso.com, O=mysso, L=SH, ST=SH, C=CN 正确吗？

[否]： 是

2.导出证书

●证书文件名： D:\cas\_sso\ca\mysso.crt

命令行：（注意-alias值和-storepass值与上一步命令行一致）

keytool -export -alias mysso -keystore D:\cas\_sso\ca\mysso.keystore -file D:\cas\_sso\ca\mysso.crt -storepass myssopwd

执行过程及输出：

D:\cas\_sso>keytool -export -alias mysso -keystore D:\cas\_sso\ca\mysso.keystore -file D:\cas\_sso\ca\mysso.crt -storepass myssopwd

保存在文件中的认证 <D:\cas\_sso\mysso.crt>

3.将上一步生成的证书导入客户端（注意：仅用于J2EE类型接入应用端，CAS Server端不需要此步操作。另，导入证书后需要重启接入应用端）

命令行：（注意-alias与约定匹配）

keytool -import -keystore %JAVA\_HOME%\jre\lib\security\cacerts -file D:\cas\_sso\ca\mysso.crt -alias mysso -storepass changeit

(keytool -import -keystore %JAVA\_HOME%\jre\lib\security\cacerts -file D:\cas\_sso\ca\cnjysso.crt -alias cnjysso -storepass changeit)

执行过程及输出：（注意-storepass为jdk keystore：%JAVA\_HOME%\jre\lib\security\cacerts文件的密码，而非上几步使用的keystore的密码。密码默认为changeit，如曾经修改过则需要修改过的实际密码）

D:\cas\_sso>keytool -import -keystore %JAVA\_HOME%\jre\lib\security\cacerts -file D:\cas\_sso\ca\mysso.crt -alias mysso -storepass changeit

所有者:CN=center.mysso.com, OU=mysso.com, O=mysso, L=SH, ST=SH, C=CN

签发人:CN=center.mysso.com, OU=mysso.com, O=mysso, L=SH, ST=SH, C=CN

序列号:5281d850

有效期: Tue Nov 12 15:27:12 CST 2013 至Wed Nov 12 15:27:12 CST 2014

证书指纹:

MD5:39:A2:12:9A:CB:B3:56:EA:7D:FD:F7:73:95:78:21:F4

SHA1:9C:1B:ED:74:F2:84:3A:87:E7:5D:99:C1:2C:3D:E5:24:7A:53:35:CA

签名算法名称:SHA1withRSA

版本: 3

信任这个认证？ [否]： 是

认证已添加至keystore中

//================================================================================

//================================================================================

// 步骤4：部署CAS Server所在的Tomcat

//================================================================================

1.将下载的Tomcat压缩包解压部署为

D:\cas\_sso\apache-tomcat-7.0.47-CasServer

2.修改配置文件，配置HTTPS

D:\cas\_sso\apache-tomcat-7.0.47-CasServer\conf\server.xml

找到如下XML片段：

<!--

<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"

maxThreads="150" scheme="https" secure="true"

clientAuth="false" sslProtocol="TLS" />

-->

修改为：（注意：keystoreFile就是3.1中创建证书的路径，keystorePass就是3.1中创建证书的密码，由于本机已经启动了一个tomcat，所以8443改为8444）

<Connector port="8444" protocol="HTTP/1.1" SSLEnabled="true"

maxThreads="150" scheme="https" secure="true"

keystoreFile="D:\cas\_sso\ca\mysso.keystore" keystorePass="myssopwd"

clientAuth="false" sslProtocol="TLS" URIEncoding="UTF-8"/>

顺便在该文件中，把普通http访问端口屏蔽了（因为Cas Server无需http访问）

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

修改为

<!--

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

-->

3.启动tomcat

执行：

D:\cas\_sso\apache-tomcat-7.0.47-CasServer\bin\startup.bat

4.验证HTTPS配置

在浏览器中打开：

https://center.mysso.com:8444/

确认能够访问

5.部署CAS Server

●从官方网站下载最新版本压缩包（本次为：cas-server-3.5.2-release.zip）

●解压缩后提取

cas-server-3.5.2\modules\cas-server-webapp-3.5.2.war

重命名为

cas.war

放置到tomcat的

D:\cas\_sso\apache-tomcat-7.0.47-CasServer\webapps

目录下

●启动或重启tomcat

D:\cas\_sso\apache-tomcat-7.0.47-CasServer\bin\startup.bat

●在浏览器地址栏输入

https://center.mysso.com:8444/cas/login

并确认内容能正常显示（网页标题：Central Authentication Service (CAS)）

●在CAS Server页面中登录

CAS-server的默认验证规则：只要用户名和密码相同就认证通过（仅仅用于测试，生成环境需要根据实际情况修改），输入admin/admin 点击登录，就可以看到登录成功的页面。

登录成功后显示：

登录成功

您已经成功登录中央认证系统。

出于安全考虑，一旦您访问过那些需要您提供凭证信息的应用时，请操作完成之后关闭浏览器。

//================================================================================

//================================================================================

// 步骤5：部署CAS Client所在的Tomcat

//================================================================================

1.下载Cas-Client，在

\cas-client-3.2.1-release\cas-client-3.2.1\modules

中获取

cas-client-core-3.2.1.jar

2.下载apache common logging jar包

commons-logging-1.1.1.jar

根据需要，还可以配置log4j，需要:

log4j-1.2.14.jar

log4j.properties

3.配置apache-tomcat-7.0.47-CasApp1和apache-tomcat-7.0.47-CasApp2

修改各自conf目录下server.xml，修改如下XML片段

<Server port="8005" shutdown="SHUTDOWN">

………………………………

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

………………………………

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

分别修改为：

●apache-tomcat-7.0.47-CasApp1

<Server port="18005" shutdown="SHUTDOWN">

………………………………

<Connector port="18080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="18443" />

………………………………

<Connector port="18009" protocol="AJP/1.3" redirectPort="18443" />

●apache-tomcat-7.0.47-CasApp2

<Server port="28005" shutdown="SHUTDOWN">

………………………………

<Connector port="28080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="28443" />

………………………………

<Connector port="28009" protocol="AJP/1.3" redirectPort="28443" />

4.启动apache-tomcat-7.0.47-CasApp1和apache-tomcat-7.0.47-CasApp2

●执行各自bin目录下的startup.bat，确认终端未报错

D:\cas\_sso\apache-tomcat-7.0.47-CasApp1\bin\startup.bat

D:\cas\_sso\apache-tomcat-7.0.47-CasApp2\bin\startup.bat

●用浏览器打开各自的验证页面：（借以tomcat默认自带的 webapps\examples 作为演示的简单web项目）

http://app1.mysso.com:18080/examples/servlets/

http://app2.mysso.com:28080/examples/servlets/

如页面正常显示则表示app1和app2的tomcat配置成功

5.将cas-client-core-3.2.1.jar和commons-logging-1.1.1.jar复制到app1和app2所在的tomcat（cas client的listener需要commons-logging-1.1.1.jar）

目的目录分别为：

D:\cas\_sso\apache-tomcat-7.0.47-CasApp1\webapps\examples\WEB-INF\lib

D:\cas\_sso\apache-tomcat-7.0.47-CasApp2\webapps\examples\WEB-INF\lib

如果需要配置log4j，则将log4j-1.2.14.jar分别复制到

D:\cas\_sso\apache-tomcat-7.0.47-CasApp1\webapps\examples\WEB-INF\lib

D:\cas\_sso\apache-tomcat-7.0.47-CasApp2\webapps\examples\WEB-INF\lib

将log4j.properties分别复制到

D:\cas\_sso\apache-tomcat-7.0.47-CasApp1\webapps\examples\WEB-INF\classes

D:\cas\_sso\apache-tomcat-7.0.47-CasApp2\webapps\examples\WEB-INF\classes

6.修改app1的web.xml

所在路径：

D:\cas\_sso\apache-tomcat-7.0.47-CasApp1\webapps\examples\WEB-INF\web.xml

添加如下片段：（注意4个URL）

<!-- ======================== 单点登录开始 ======================== -->

<!-- 用于单点退出，该过滤器用于实现单点登出功能，可选配置-->

<listener>

<listener-class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</listener-class>

</listener>

<!-- 该过滤器用于实现单点登出功能，可选配置。 -->

<filter>

<filter-name>CAS Single Sign Out Filter</filter-name>

<filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS Single Sign Out Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter>

<filter-name>CAS Filter</filter-name>

<filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>

<init-param>

<param-name>casServerLoginUrl</param-name>

<param-value>https://center.mysso.com:8444/cas/login</param-value><!-- 这里根路径需要修改为认证中心URL，子路径不变 -->

</init-param>

<init-param>

<param-name>serverName</param-name>

<param-value>http://app1.mysso.com:18080</param-value><!-- 这里根路径需要修改为app1 URL -->

</init-param>

</filter>

<filter-mapping>

<filter-name>CAS Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 该过滤器负责对Ticket的校验工作，必须启用它 -->

<filter>

<filter-name>CAS Validation Filter</filter-name>

<filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>

<init-param>

<param-name>casServerUrlPrefix</param-name>

<param-value>https://center.mysso.com:8444/cas</param-value><!-- 这里根路径需要修改为认证中心URL，子路径不变 -->

</init-param>

<init-param>

<param-name>serverName</param-name>

<param-value>http://app1.mysso.com:18080</param-value><!-- 这里根路径需要修改为app1 URL -->

</init-param>

</filter>

<filter-mapping>

<filter-name>CAS Validation Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!--

该过滤器负责实现HttpServletRequest请求的包裹，

比如允许开发者通过HttpServletRequest的getRemoteUser()方法获得SSO登录用户的登录名，可选配置。

-->

<filter>

<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!--

该过滤器使得开发者可以通过org.jasig.cas.client.util.AssertionHolder来获取用户的登录名。

比如AssertionHolder.getAssertion().getPrincipal().getName()。

-->

<filter>

<filter-name>CAS Assertion Thread Local Filter</filter-name>

<filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>CAS Assertion Thread Local Filter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- ======================== 单点登录结束 ======================== -->

//================================================================================

//================================================================================

// 步骤6：验证SSO

//================================================================================

1.启动之前配置好的三个tomcat，执行各自bin目录下的startup.bat

D:\cas\_sso\apache-tomcat-7.0.47-CasServer\bin\startup.bat

D:\cas\_sso\apache-tomcat-7.0.47-CasApp1\bin\startup.bat

D:\cas\_sso\apache-tomcat-7.0.47-CasApp2\bin\startup.bat

2.测试预期

预期流程：

1）打开app1 url

2）-> 跳转cas server 验证

3）-> 显示app1的应用

4）-> 打开app2 url

5）-> 显示app2 应用

6）-> 注销cas server

7）-> 打开app1/app2 url

8）-> 重新跳转到cas server 验证

3.测试过程

1）打开app1 url：

浏览器输入地址：

http://app1.mysso.com:18080/examples/servlets/servlet/HelloWorldExample

这时候浏览器会跳转，由于证书是自制的，非系统认证，所以会有提示警告：（选择“继续浏览此网站”即可）

此网站的安全证书有问题。

此网站出具的安全证书不是由受信任的证书颁发机构颁发的。

安全证书问题可能显示试图欺骗您或截获您向服务器发送的数据。

建议关闭此网页，并且不要继续浏览该网站。

单击此处关闭该网页。

继续浏览此网站(不推荐)。

更多信息

如果通过单击链接到达此页面，请检查地址栏中的网站以确保该地址是您希望到达的页面。

转到如 https://example.com 等网站时，请尝试将 "www" 添加到地址中，变为 https://www.example.com。

如果选择忽略此错误并继续，请不要在网站中输入私人信息。

有关详细信息，请参阅 Internet Explorer 帮助中的“证书错误”。

2）-> 跳转cas server 验证

浏览器跳转至CAS认证中心

https://center.mysso.com:8444/cas/login?service=http%3A%2F%2Fapp1.mysso.com%3A18080%2Fexamples%2Fservlets%2Fservlet%2FHelloWorldExample

登录页面中输入admin/admin

3）-> 显示app1的应用

认证通过后，浏览器跳转回APP1

http://app1.mysso.com:18080/examples/servlets/servlet/HelloWorldExample;jsessionid=97A9D8573334F546F3672A7C76A41809

页面显示

HelloWorld!

4）打开app2 url

浏览器输入地址：

http://app2.mysso.com:28080/examples/servlets/servlet/HelloWorldExample

5）-> 显示app2 应用

打开URL后，浏览器地址会变成

http://app2.mysso.com:28080/examples/servlets/servlet/HelloWorldExample;jsessionid=E1129E111832C5A3798F124DD855E8A5

页面显示

HelloWorld!

6）-> 注销cas server

关闭浏览器，则当前Session失效

7）-> 打开app1/app2 url

浏览器输入地址：

http://app1.mysso.com:18080/examples/servlets/servlet/HelloWorldExample

or

http://app2.mysso.com:28080/examples/servlets/servlet/HelloWorldExample

8）-> 重新跳转到cas server 验证

浏览器跳转至CAS认证中心

https://center.mysso.com:8444/cas/login?service=http%3A%2F%2Fapp1.mysso.com%3A18080%2Fexamples%2Fservlets%2Fservlet%2FHelloWorldExample

验证通过！

//================================================================================