# 前提

什么是单点登录

单点登录（Single Sign On），简称为 SSO，是目前比较流行的企业业务整合的解决方案之一。SSO的定义是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。

系统存在诸多子系统，而这些子系统是分别部署在不同的服务器中，那么使用传统方式的session是无法解决的，我们需要使用相关的单点登录技术来解决。

什么是CAS

CAS 是 Yale 大学发起的一个开源项目，旨在为 Web 应用系统提供一种可靠的单点登录方法，CAS 在 2004 年 12 月正式成为 JA-SIG 的一个项目。CAS 具有以下特点：

【1】开源的企业级单点登录解决方案。

【2】CAS Server 为需要独立部署的 Web 应用。

【3】CAS Client 支持非常多的客户端(这里指单点登录系统中的各个 Web 应用)，包括 Java, .Net, PHP, Perl, Apache, uPortal, Ruby 等。

CAS 包含两个部分： CAS Server 和 CAS Client。CAS Server 需要独立部署，主要负责对用户的认证工作；CAS Client 负责处理对客户端受保护资源的访问请求，需要登录时，重定向到 CAS Server

SSO单点登录访问流程主要有以下步骤：

1. 访问服务：SSO客户端发送请求访问应用系统提供的服务资源。

2. 定向认证：SSO客户端会重定向用户请求到SSO服务器。

3. 用户认证：用户身份认证。

4. 发放票据：SSO服务器会产生一个随机的Service Ticket。

5. 验证票据：SSO服务器验证票据Service Ticket的合法性，验证通过后，允许客户端访问服务。

6. 传输用户信息：SSO服务器验证票据通过后，传输用户认证结果信息给客户端。

# 使用步骤

## 第一步 服务端部署步骤

Cas服务端其实就是一个war包。

第一步 获取war包cas-server-webapp-4.0.0.war，部署tomcat中，进行正常登录有页面

在资源\cas\source\cas-server-4.0.0-release\cas-server-4.0.0\modules目录下

原始用户名和密码 casuser /Mellon

第二步 配置端口

1. tomcat 目录 conf\server.xml中配置 9100 客户端需要用到此端口

2、修改CAS配置文件：修改cas的WEB-INF/cas.properties

server.name=http://localhost:9100客户端需要用到此端口

第三步 去除https认证

CAS默认使用的是HTTPS协议，如果使用HTTPS协议需要SSL安全证书（需向特定的机构申请和购买） 。如果对安全要求不高或是在开发测试阶段，可使用HTTP协议。我们这里讲解通过修改配置，让CAS使用HTTP协议。

1. 修改cas的WEB-INF/deployerConfigContext.xml

<bean class="org.jasig.cas.authentication.handler.support.HttpBasedServiceCredentialsAuthenticationHandler"

p:httpClient-ref="httpClient" p:requireSecure="false"/> 添加新属性，requireSecure属性是否需要安全验证，即HTTPS

1. 修改cas的/WEB-INF/spring-configuration/ticketGrantingTicketCookieGenerator.xml

<bean id="ticketGrantingTicketCookieGenerator" class="org.jasig.cas.web.support.CookieRetrievingCookieGenerator"

p:cookieSecure="false" 改为：false：HTTPS验证相关，true为采用HTTPS验证，false为不采用https验证。

p:cookieMaxAge="3600" COOKIE的最大生命周期，-1为无生命周期，即只在当前打开的窗口有效，单位秒

p:cookieName="CASTGC"

p:cookiePath="/cas" /> </bean>

(3) 修改cas的WEB-INF/spring-configuration/warnCookieGenerator.xml

<bean id="warnCookieGenerator" class="org.jasig.cas.web.support.CookieRetrievingCookieGenerator"

p:cookieSecure=" false " 和配置第二步一样改为false

p:cookieMaxAge="3600"和配置2一样改为3600秒

p:cookieName="CASPRIVACY"

p:cookiePath="/cas" />

## 第二步 客户端的配置

### 原生

只配置web.xml配置 (就是不使用spring框架)即可

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

version="2.5">

<!-- ======================== 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 一个服务器用户退出了，全部退出-->

<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>

<!-- 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，服务端的地址在哪里，要和服务器端口一致-->

<param-value>http://localhost:9100/cas/login</param-value>

</init-param>

<init-param>

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

<!-- 这里是客户端的ip -->

<param-value>http://localhost:9001</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CASFilter</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>

<!--这里的server是服务端的IP， 服务端的地址在哪里，要和服务器端口一致 -->

<param-value>http://localhost:9100/cas</param-value>

</init-param>

<init-param>

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

<!-- 这里是客户端的ip -->

<param-value>http://localhost:9001</param-value>

</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 ======================== -->

</web-app>

#### 测试 单点登陆

创建子个工程：两个war包：两个个工程web.xml中配置单点服务cas；端口8080和8081

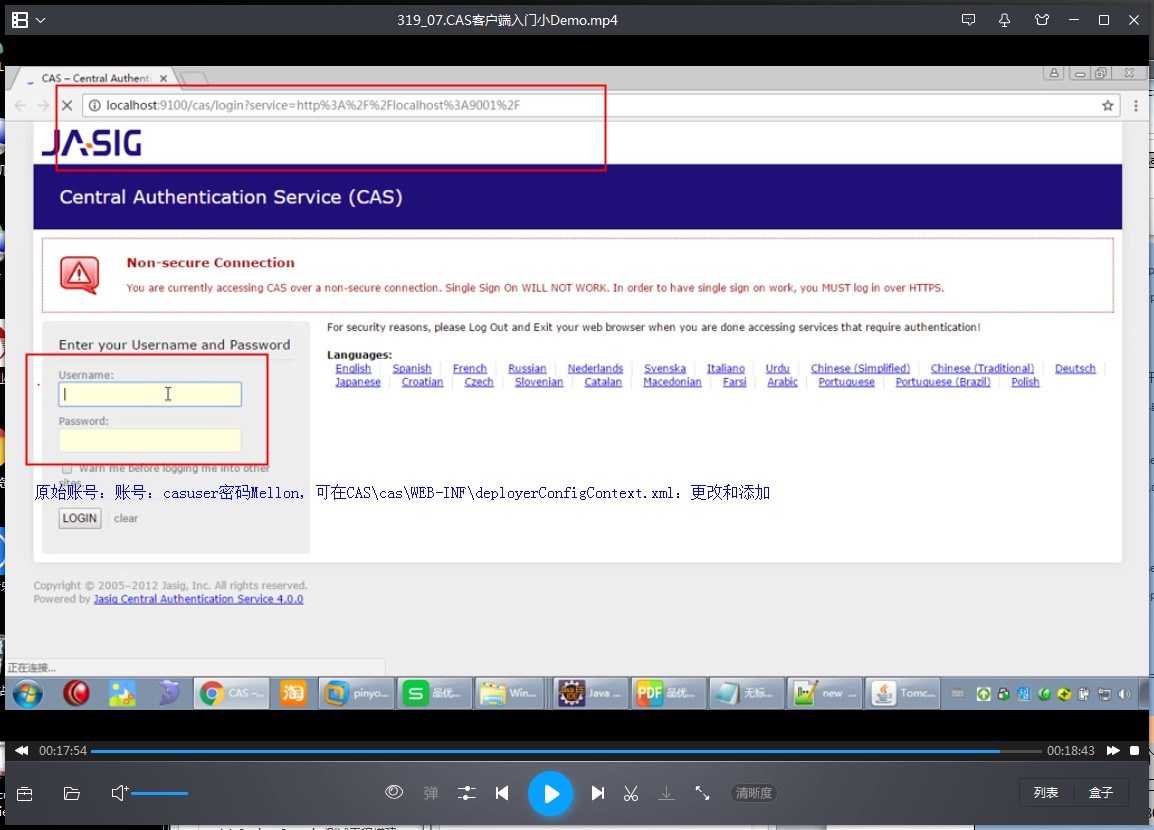
两个工程创建jsp页面:index.jsp

内容：欢迎来到一品优购 <%=request. getRemoteUser()%> 注意：要在web中有配置了才能使用此方法

内容：欢迎来到二品优购 <%=request. getRemoteUser()%> 注意：要在web中有配置了才能使用此方法

启动服务端：此端口是9100，工程名：cas

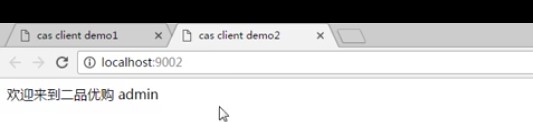
进入网址localhost:8080：跳转到web.xml配置的服务端地址localhost:9100/cas自带登录页面



登录成功后index工程一的页面：

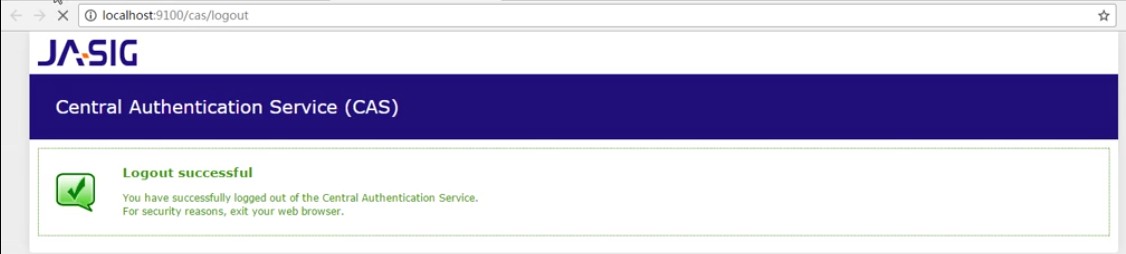


测试工程二：进入网址localhost:8081：就不会再进入登录页面了，而是跳转登录成功页面



#### 测试 单点退出

系统本身自带一个退出地址：localhost:9100/cas/logout



刷新之前两个登录的页面：就会退回两个原始登录页面

进入这个页面不友好：进行改退出后的页面操作

修改改服务端WEB-ING目录下配置文件修改cas-servlet.xml

<bean id="logoutAction" class="org.jasig.cas.web.flow.LogoutAction"

p:servicesManager-ref="servicesManager"

p:followServiceRedirects="${cas.logout.followServiceRedirects:true}"/>改为true

改为true后，可以在退出时跳转页面到目标页面，修改index.jsp的退出链接

<a href="http://localhost:9100/cas/logout?service=http://www.baidu.com">退出登录</a>

## 第三步 数据源的配置

系统中的都是写死的，现在从数据库user表里取数据去做验证

（1）修改cas服务端中web-inf下deployerConfigContext.xml ，添加如下配置

1、在此xml最后配置c3p0数据源，第3步要用到

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"

p:driverClass="com.mysql.jdbc.Driver"

p:jdbcUrl="jdbc:mysql://127.0.0.1:3306/pinyougoudb?characterEncoding=utf8"

p:user="root"

p:password="123456" />

2、MD5加密密码类，第3步要用到

<bean id="passwordEncoder"

class="org.jasig.cas.authentication.handler.DefaultPasswordEncoder"

c:encodingAlgorithm="MD5"

p:characterEncoding="UTF-8" />

3、管理认证：属性需要数据源，sql语句，密码适配器：用户输入密码更换为MD5的密码，

<bean id="dbAuthHandler"

class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler"

p:dataSource-ref="dataSource"

p:sql="select password from tb\_user where username = ?"

p:passwordEncoder-ref="passwordEncoder"/>

4、在此xml中找到下面内容：更换原始写死的账户密码处理器，换成自己写的第三步处理器

<bean id="authenticationManager" class="org.jasig.cas.authentication.PolicyBasedAuthenticationManager">

<constructor-arg>

<map>

<entry key-ref="proxyAuthenticationHandler" value-ref="proxyPrincipalResolver" />

· <!—引入第三步配置bean -->

<entry key-ref=" dbAuthHandler " value-ref="primaryPrincipalResolver" />

</map>

</constructor-arg>

其中

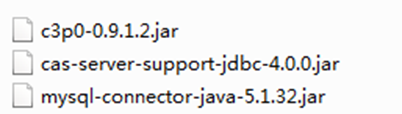
<entry key-ref="primaryAuthenticationHandler" value-ref="primaryPrincipalResolver" />

这句是使用固定的用户名和密码，我们在下面可以看到这两个bean ,如果我们使用数据库认证用户名和密码，需要将这句注释掉。

添加下面这一句配置

<entry key-ref="dbAuthHandler" value-ref="primaryPrincipalResolver"/>

（2）将以下三个jar包放入webapps\cas\WEB-INF\lib下



## 第四步 登录界面更换

（1）将自己的登陆页login.html拷贝到cas系统下WEB-INF\view\jsp\default\ui 目录下

（2）将css js等文件夹拷贝到 cas目录下

（3） 将原来的casLoginView.jsp 改名（可以为之后的修改操作做参照），将login.html改名为casLoginView.jsp

编辑casLoginView.jsp 内容

（1）添加指令

<%@ page pageEncoding="UTF-8" %>

<%@ page contentType="text/html; charset=UTF-8" %>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

（2）修改form标签

<form:form method="post" id="fm1" commandName="${commandName}" htmlEscape="true" class="sui-form">

......

</form:form>

（3）修改用户名框

<form:input id="username" tabindex="1"

accesskey="${userNameAccessKey}" path="username" autocomplete="off" htmlEscape="true"

placeholder="邮箱/用户名/手机号" class="span2 input-xfat" />

（4）修改密码框

<form:password id="password" tabindex="2" path="password"

accesskey="${passwordAccessKey}" htmlEscape="true" autocomplete="off"

placeholder="请输入密码" class="span2 input-xfat" />

（5）修改登陆按钮

<input type="hidden" name="lt" value="${loginTicket}" /> //票据

<input type="hidden" name="execution" value="${flowExecutionKey}" />

<input type="hidden" name="\_eventId" value="submit" />

<input class="sui-btn btn-block btn-xlarge btn-danger" accesskey="l" value="登陆" type="submit" />

### 错误信息提示

在表单内加入错误提示框

<form:errors path="\*" id="msg" cssClass="errors" element="div" htmlEscape="false" />

测试：输入错误的用户名和密码，提示是英文。

这个提示信息是在WEB-INF\classes目录下的messages.properties文件中

authenticationFailure.AccountNotFoundException=Invalid credentials.

authenticationFailure.FailedLoginException=Invalid credentials.

设置国际化为zn\_CN ,修改cas-servlet.xml

<bean id="localeResolver" class="org.springframework.web.servlet.i18n.CookieLocaleResolver" p:defaultLocale="zh\_CN" />

把上面两句话拷贝到messages\_zh\_CN.properties下，并改为中文提示（转码）

第一个是用户名不存在时的错误提示 (自己定义值)

第二个是密码错误的提示 (自己定义值)

authenticationFailure.AccountNotFoundException=\u7528\u6237\u4E0D\u5B58\u5728.

authenticationFailure.FailedLoginException=\u5BC6\u7801\u9519\u8BEF.