1、单点登录简介

单点登录使得在多个应用系统中，用户只需登录一次就可以访问所有相互信任的应用系统，而CAS是实现单点登录框架，开始有耶鲁大学的一个组织开发，后来归apereo管，开源，遵循apache 2.0协议，主要为web应用系统提供一种可靠的单点登录方法，github代码地址：https://github.com/apereo/cas

从结构上看，cas包括两部分：CAS Server和CAS Client,CAS Server负责完成对用户的认证工作，CAS Client负责对客户端受保护资源的访问请求，需要对请求方进行身份认证时，重定向到CAS Server进行认证

2、基本原理与实现

假设有三个角色，用户(浏览器)、CAS Server（http://Server/cas）即统一认证中心和CAS Client（http://Client/cas），当用户访问http://Client/cas
时，过滤器会判断用户是否登录，没有登陆，则重定向到网站http://Server/cas,重定向到Server/cas后，输入用户名和密码，Server/cas将用户登陆的信息记录到服务器的session中，之后Server/cas给浏览器发送一个特殊的凭证，浏览器将凭证交给Client/cas,Client/cas则拿着浏览器交给他的凭证去Server/cas验证凭证是否有效，从而判断用户是否登录成功

当cas-server认证通过后，会返回给浏览器302，重定向的地址就是Referer中的service参数对应的值。后边通过get的方式携带了一个ticket令牌，这个ticket就是ST，同时会在Cookie中设置一个CASTGC，该Cookie是网站server/cas的cookie，只有访问这个网站才会携带这个cookie过去

cookie中的CASTGC：向cookie中添加该值的目的是当下次访问server/cas时，浏览器将cookie中的TGC携带到服务器中，服务器根据这个TGC，查找与之对应的TGT，从而判断用户是否登录过了，是否需要展示登录界面

TGT：Ticket Granted Ticket(大令牌、票根，他可以签发ST)

TGC：Ticket Granted Cookie（cookie中的value），存在Cookie中，根据他可以找到TGT，他就是TGT的ID,相当于一个句柄的作用

ST:Service Ticket(小令牌),是TGT生成的，默认用一次失效，即ticket值

cookie中相当于有TGT，再一次请求server/cas时，会在过过滤器中取到ticket的值，即通过TGT签发得到的ST，然后通过http方式调用server/cas验证该ticket是否有效

2.1、cas简单demo实现

2.11、证书生成

对于实现ssl的方式，需要生成证书，主要步骤如下：

- 1.生成证书，在cmd窗口输入以下命令：

```
keytool -genkey -alias ssodemo -keyalg RSA -keysize 1024 -keypass longming -validity 365 -keystore
c:\longming.keystore -storepass longming
```

-alias后面的别名自定义，-keypass指定证书密码，注意-storepass和前面的keypass的密码相同，不然tomcat配置https会访问失败 -keystore指定证书的位置，
执行命令后出现一些信息，其中第一个让你输入“您的名字和姓氏是什么”，必须输入在**C:\Windows\System32\drivers\etc\hosts**文件中加入的服务端的域名，因为cas只能通过域名来访问，不能通过ip访问，并且如果不这么做，在最后cas回调转入你想访问的客户端应用的时候，会出现No subject alternative names present错误信息

- 2.导出证书：在命令窗口输入以下命令

```
keytool -export -alias ssodemo -keystore c:\longming.keystore -file c:\ssodemo.crt -storepass longming
```

-alias后面的名称要与生成证书的命令里面的alias的名称一致. –keystore后面指定证书存放的位置，这里我放在C盘根目录，同时证书名称要与第一步对应的命令里的keystore名称一致.这里是longming.keystore，-file后面才crt路径，指定在c盘根目录. –storepass的证书密码要与上面输入的密码一致.

- 3.客户端导入证书：在cmd输入以下命令

```
keytool -import -keystore %JAVA_HOME%\jre\lib\security\cacerts -file c:\ ssodemo.crt -alias ssodemo
使用上面的我发生了错误，改成：
keytool -import -alias ssodemo -keystore cacerts -file d:\ssodemo.crt
说明：输入该命令的前提是需要进入C:\Program Files\Java\jre1.8.0_131\lib\security>，然后在输入该命令
```

-file指定证书的位置，也就是上一步导出证书的位置，即c:\ ssodemo.crt 命令中指定了JAVA_HOME，意思是将证书导入到客户端证书库，也就是jdk证书库中.因为客户端应用运行在本地，需要jdk的支持。

回车之后，会让你输入密钥库口令，注意，这里的密码必须要输入changeit，不能输入上面指定的密码longming，否则导入客户端证书会有问题，如果是多台机器演示，需要在每一台客户端导入该证书，步骤都是一样的。当看到提示“是否信任此证书”，输入y回车即可

2.12、配置

- 1、tomcat配置文件修改
  由于实现了ssl，需要在server.xml文件加上https的配置

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
               maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
         keystoreFile="D:/longming.keystore" keystorePass="longming"
               clientAuth="false" sslProtocol="TLS" />
```

- 2、修改cas server中的\WEB-INF\ deployerConfigContext.xml文件
  来支持mysql数据库交互验证，这里配置为本地数据库

```
<bean id="authenticationManager" class="org.jasig.cas.authentication.PolicyBasedAuthenticationManager">
        <constructor-arg>
            <map>
                <!--
                   | IMPORTANT
                   | Every handler requires a unique name.
                   | If more than one instance of the same handler class is configured, you must explicitly
                   | set its name to something other than its default name (typically the simple class name
                   -->
                <entry key-ref="proxyAuthenticationHandler" value-ref="proxyPrincipalResolver" />
                <!-- <entry key-ref="primaryAuthenticationHandler" value-ref="primaryPrincipalResolver" />
                <!-- key-ref指定自己的本地数据库访问 -->
                <entry key-ref="dbAuthHandler" value-ref="primaryPrincipalResolver"/>
            </map>
        </constructor-arg>

        <!-- Uncomment the metadata populator to allow clearpass to capture and cache the password
             This switch effectively will turn on clearpass.
        <property name="authenticationMetaDataPopulators">
           <util:list>
              <bean class="org.jasig.cas.extension.clearpass.CacheCredentialsMetaDataPopulator"
                    c:credentialCache-ref="encryptedMap" />
           </util:list>
        </property>
        -->

        <!--
           | Defines the security policy around authentication. Some alternative policies that ship with CA
           |
           | * NotPreventedAuthenticationPolicy - all credential must either pass or fail authentication
           | * AllAuthenticationPolicy - all presented credential must be authenticated successfully
           | * RequiredHandlerAuthenticationPolicy - specifies a handler that must authenticate its credent
           -->
        <property name="authenticationPolicy">
            <bean class="org.jasig.cas.authentication.AnyAuthenticationPolicy" />
        </property>
    </bean>
```

目的是指定自己的本地数据库访问
然后在加入2个bean的配置，如下：

```
<!-- 指定c3p0数据源 -->
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="com.mysql.jdbc.Driver" />
        <property ame="jdbcUrl" value="jdbc:mysql://127.0.0.1:3306/myProject?useUnicode=true&amp;characterE
        <property name="user" value="root" />
        <property name="password" value="root
</bean>
```

```
 <!--访问本地数据库  -->
    <bean id="dbAuthHandler"
      class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler"
      p:dataSource-ref="dataSource"
      p:sql="SELECT u.`password` FROM `user` u WHERE u.`user_name` = ?" />
```

重新访问cas，输入数据库中存在的用户名和密码，登陆成功，说明配置无误

- 3、创建一个Maven的web项目（充当与cas client）
  加入依赖：

```
<dependency>
    <groupId>org.jasig.cas.client</groupId>
    <artifactId>cas-client-core</artifactId>
    <version>3.2.1</version>
</dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.16</version>
</dependency>
```

在web.xml中加入过滤器：

```
<!-- 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://server.longming.com:8443/cas/login</param-value>
    </init-param>
    <init-param>
      <param-name>serverName</param-name>
      <param-value>http://client1.longming.com:18080</param-value>
    </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.Cas10TicketValidationFilter</filter-class>
    <init-param>
      <param-name>casServerUrlPrefix</param-name>
      <param-value>https://server.longming.com:8443/cas</param-value>
    </init-param>
    <init-param>
      <param-name>serverName</param-name>
      <param-value>http://client1.longming.com:18080</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
      -->
  <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
  -->
  <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>
```

访问：http://client1.longming.com:18080/，会自动跳转到CAS认证界面 https://server.longming.com:8443/cas/login?service=http%3A%2F%2Fclient1.longming.com%3A18080%2F,因为没有登陆过CAS认证系统，CAS认证系统拦截你访问的客户端应用，首先进入到认证系统登陆界面，同时URL后面加上你想访问的地址信息，登陆成功后，会跳转到 http://client1.longming.com:18080/http://client1.longming.com:18080/;jsessionid=976B04415BED56517271994CECE23015

错误解决：如果出现PKIX path building failed错误,可能原因是客户端run的jdk和server端run的jdk版本不同，仔细检查一下，run的是不是你导入证书的那个jdk，还有生成证书是需注意“您的名字和姓氏”这一项中，需天cas server的host name

对于基于springmvc框架实现的cas，可以在applicationContext.xml加入过滤器

```
<!-- CAS 基本属性配置-->
<beans:bean id="serviceProperties"
            class="org.springframework.security.cas.ServiceProperties">
    <beans:property name="service"
                    value="${cas.service}"/>
    <beans:property name="sendRenew" value="false"/>
</beans:bean>

<!-- CAS Filter 配置 -->
<beans:bean id="casFilter"
            class="org.springframework.security.cas.web.CasAuthenticationFilter">
    <beans:property name="authenticationManager" ref="authenticationManager"/>
    <beans:property name="authenticationSuccessHandler" ref="successHandler"/>
</beans:bean>

<beans:bean id="successHandler" class="com.hand.hap.security.CustomAuthenticationSuccessHandler">
    <beans:property name="defaultTargetUrl" value="/index"/>
</beans:bean>


<beans:bean id="casEntryPoint"
            class="org.springframework.security.cas.web.CasAuthenticationEntryPoint">
    <beans:property name="loginUrl" value="${cas.ssoserver.loginurl}"/>
    <beans:property name="serviceProperties" ref="serviceProperties"/>
</beans:bean>


<authentication-manager alias="authenticationManager">
    <authentication-provider ref="casAuthenticationProvider"/>
    <authentication-provider user-service-ref="customUserDetailsService">
        <password-encoder ref="passwordManager"/>
    </authentication-provider>
</authentication-manager>

<beans:bean id="casAuthenticationProvider"
            class="org.springframework.security.cas.authentication.CasAuthenticationProvider">
    <beans:property name="serviceProperties" ref="serviceProperties" />
    <beans:property name="authenticationUserDetailsService" ref="customAuthenticationUserDetailsService"/>
    <beans:property name="ticketValidator">
        <beans:bean class="org.jasig.cas.client.validation.Cas20ServiceTicketValidator">
            <beans:constructor-arg index="0" value="${cas.ssoserver.url}" />
        </beans:bean>
    </beans:property>
    <beans:property name="key" value="an_id_for_this_auth_provider_only"/>
</beans:bean>


<beans:bean id="customAuthenticationUserDetailsService" class="com.hand.hap.security.CustomAuthenticationUs
</beans:bean>


<beans:bean id="singleLogoutFilter" class="org.jasig.cas.client.session.SingleSignOutFilter"/>
<!-- This filter redirects to the CAS Server to signal Single Logout should be performed -->
<beans:bean id="requestSingleLogoutFilter"
```

```
            class="org.springframework.security.web.authentication.logout.LogoutFilter">
    <beans:constructor-arg value="${cas.ssoserver.logouturl}"/>
    <beans:constructor-arg>
        <beans:bean class="org.springframework.security.web.authentication.logout.SecurityContextLogoutHand
    </beans:constructor-arg>
    <beans:property name="filterProcessesUrl" value="/logout"/>
</beans:bean>

<beans:bean id="csrfCasSecurityRequestMatcher" class="com.hand.hap.security.CsrfSecurityRequestMatcher">
    <beans:property name="excludeUrls">
        <beans:list>
            <beans:value>/login</beans:value>
            <beans:value>/websocket/**</beans:value>
        </beans:list>
    </beans:property>
</beans:bean>

<beans:bean id="captchaVerifierFilter" class="com.hand.hap.security.CaptchaVerifierFilter">
    <beans:property name="captchaField" value="verifiCode"/>
</beans:bean>
<beans:bean id="loginFailureHandler" class="com.hand.hap.security.LoginFailureHandler"/>
```

3、关闭HTTPS，使用http的方法

1.修改WEB-INF/deployerConfigContext.xml

```
<bean id="proxyAuthenticationHandler"
    class="org.jasig.cas.authentication.handler.support.HttpBasedServiceCredentialsAuthenticationHandler"
    p:httpClient-ref="supportsTrustStoreSslSocketFactoryHttpClient"
    p:requireSecure="false" />
```

在p:httpClient-ref="supportsTrustStoreSslSocketFactoryHttpClient"后增加p:requireSecure="false"

2.修改WEB-INF/spring-configuration/ticketGrantingTicketCookieGenerator.xml

```
<bean id="ticketGrantingTicketCookieGenerator" class="org.jasig.cas.web.support.CookieRetrievingCookieGener
        c:casCookieValueManager-ref="cookieValueManager"
        p:cookieSecure="false"
        p:cookieMaxAge="-1"
        p:cookieName="TGC"
        p:cookiePath=""/>
```

将p:cookieSecure="true"修改为p:cookieSecure="false"

3.修改WEB-INF/spring-configuration/warnCookieGenerator.xml

```
<bean id="warnCookieGenerator" class="org.jasig.cas.web.support.CookieRetrievingCookieGenerator"
        p:cookieHttpOnly="false"
        p:cookieSecure="false"
        p:cookieMaxAge="-1"
        p:cookieName="CASPRIVACY"
        p:cookiePath=""/>
```

将p:cookieSecure="true"修改为p:cookieSecure="false"

4.修改注册服务WEB-INF/classes/services/HTTPSandIMAPS-10000001.json
将"serviceId" : "^(https|imaps)//.*"修改为"serviceId" : "^(https|http|imaps)//.*"

少了第4步，会出现Application Not Authorized to Use CAS
The application you attempted to authenticate to is not authorized to use CAS的错误信息

4、自定义登陆页和登出后跳转页面

- 1、自定义登陆页
  修改src/main/resources/default_views.properties文件

```
# 默认登陆页面位置
#casLoginView.url=/WEB-INF/view/jsp/default/ui/casLoginView.jsp
# 自定义登陆页面
casLoginView.url=/WEB-INF/view/jsp/tiglle/ui/login.jsp
```

- 2、修改登出后跳转的页面
  修改webapp/WEB-INF/cas-servlet.xml文件

```
<bean id="logoutAction" class="org.jasig.cas.web.flow.LogoutAction"
    p:servicesManager-ref="servicesManager"
    p:followServiceRedirects="${cas.logout.followServiceRedirects:false}"/>
```

将cas.logout.followServiceRedirects:false修改为true，然后登出连接添加srevice=跳转页面

```
<a href="http://www.server.com/cas/logout?service=http://www.client1:18080/">退出登陆</a>
```