JEESZ 分布式框架单点登录集成方案

提醒：由于文档写的比较早，后面修复的 bug 没有同步到文档中，故文档只是作为一个基础的参考，实施的时候请以附件配置项为准。

第一节：单点登录简介

第一步：了解单点登录

SSO 主要特点是: SSO 应用之间使用 Web 协议(如 HTTPS) ，并且只有一个登录入口.
SSO 的体系中有下面三种角色:
1) User(多个)
2) Web 应用(多个)
3) SSO 认证中心(一个)

SSO 实现包含以下三个原则:
1) 所有的登录都在 SSO 认证中心进行。
2) SSO 认证中心通过一些方法来告诉 Web 应用当前访问用户究竟是不是通过认证的用户.
3) SSO 认证中心和所有的 Web 应用建立一种信任关系.

CAS 的基本原理 CAS(Central Authentication Service) 是 Yale 耶鲁大学发起的构建 Web SSO 的 Java 开源项目。
1.CAS 术语解释:
SSO－Single Sign On 单点登录
TGT－Ticket Granting Ticket 用户身份认证凭证票据
ST－Service Ticket 服务许可凭证票据
TGC－Ticket Granting Cookie 存放用户身份认证凭证票据的 cookie.

第二步：了解单点登录体系结构

1) CAS Server 负责完成对用户信息的认证，需要单独部署，CAS Server 会处理用户名/密码等凭证(Credentials).

2) CAS Client 部署在客户端，当有对本地 Web 应用受保护资源的访问请求，并且需要对请求方进行身份认证，重定向到 CAS Server 进行认证.

第三步：单点登录环境准备工作

1) cas-server-3.5.0-release.zip（CAS 服务端）
2) cas-client-3.3.3-release.zip（CAS 客户端）
3) apache-tomcat-7.0.40
4) cas-client-core-3.2.1.jar
5) cas-server-core-3.5.0.jar
6) cas-server-support-jdbc-3.5.0.jar

第二节：单点登录环境搭建与部署

第一步：环境部署

1. 通过 Java JDK 生成证书三部曲

证书对于实现此单点登录非常之重要，证书是服务器端和客户端安全通信的凭证，本教程只是演示，所有用了 JDK 自带的证书生成工具 keytool。

当然在实际项目中你可以到专门的证书认证中心购买证书。
中文官方网站：http://www.verisign.com/cn/
使用 JDK 自带的 keytool 生成证书

第一步生成证书:

keytool -genkey -alias mycacerts -keyalg RSA -keystore C:/common/keys/keycard

注意：输入相关信息用于生成证书.其中名字与姓氏这一最好写你的 域名，如果在单击测试你可以在 C:\Windows\System32\drivers\etc\hosts 文件中映射一个虚拟域名，
注意不要写 IP。

第二步导出证书:

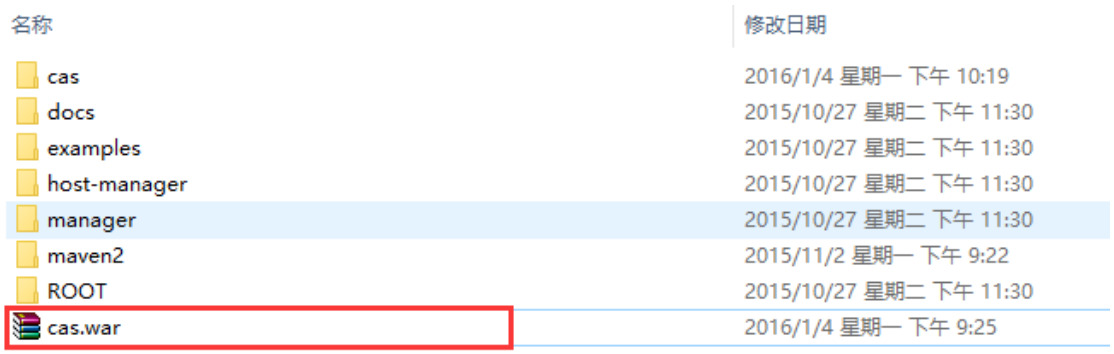
keytool -export -file C:/common/keys/keycard.crt -alias mycacerts -keystore C:/common/keys/keycard

第三步导入到 JDK 安装目录证书:

keytool -import -keystore C:/"Program Files"/Java/jdk1.6.0_32/jre/lib/security/cacerts -file C:/common/keys/keycard.crt -alias mycacerts

2. 解压 cas-server-3.5.0-release.zip 文件，

在 cas-server-3.5.0-release\cas-server-3.5.0\modules 目录下找到 cas-server-webapp-3.5.0.war 文件并命名为 cas.war，并复制到在 Tomcat 根目录的 webapps 目录下，
如下图：



3. 修改 host 文件（C:\Windows\System32\drivers\etc）hosts 文件中添加添加以下配置
127.0.0.1 jeesz.cn (配置自己的域名.)

注意：如果想在一台 PC 机上模拟这个单点登录，就必须域名重定向，如果是多台ＰＣ机，可以不配置此项，下文有用到 fast-web.cn，可以用相应ＰＣ机的ＩＰ代替

4. 修改 Tomcat 文件下的 server.xml(apache-tomcat-7.0.40\conf\server.xml) 添加以下内容：

```
<Host name="jeesz.cn"  appBase="cas" unpackWARs="true" autoDeploy="true"></Host>
```

在 server.xml 文件中把

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
                maxThreads="150" scheme="https" secure="true"
                clientAuth="false" sslProtocol="TLS" />
```

修改成如下：

```
<Connector
        port="8443"
        protocol="org.apache.coyote.http11.Http11Protocol"
        maxThreads="150"
        SSLEnabled="true"
        scheme="https"
        secure="true"
        clientAuth="false"
        sslProtocol="TLS"
        keystoreFile="C:/common/keys/keycard" <!—证书路径--->
        keystorePass="xxxxxx "                <!—证书密码-->
        ciphers="TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_RSA_WITH_AES_
        128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE_RSA_WITH_
        AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_128_C
        BC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA"
```

/>

5.启动 Tomcat 服务，查看信息，（如果有报错，可以根据信息查找错误），打开浏览器，输入 http://jeesz.cn:8080/cas 如果出现以下界面，则代表 CAS 服务端配置成功。

注：这个是最简单的 CAS 服务，只要输入的用户名跟密码一样，就可以正常登陆，在我们实际开发中，这个验证因为跟数据库作比较，接下来，我们就配置数据库校验。

第二步：配置数据库验证

1.在 apache-tomcat-7.0.2\webapps\cas\WEB-INF 目录下找到 deployerConfigContext.xml 文件，找到如下代码：

```
<property name="authenticationHandlers">
```

添加下面代码：

```
<bean class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler">
    <!--这里 sql 属性是从 user 表中根据 cas 登陆名查找密码-->
    <property name="sql" value="select password from user where username=?" />
    <property name="dataSource" ref="dataSource" />
</bean>
```

2.增加数据源 dataSource，

在 deployerConfigContext.xml，（跟上面同一个文件）找到

```
<bean id="serviceRegistryDao" class="org.jasig.cas.services.InMemoryServiceRegistryDaoImpl" />
```

，在下面添加如下代码：

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName">
        <value>com.mysql.jdbc.Driver</value>
    </property>
    <property name="url">
    <value>jdbc:mysql://127.0.0.1:3306/sso</value> <!---根据自己的数据库 URL 地址-->
    </property>
    <property name="username">
        <value>root</value> <!--根据自己的数据库用户名-->
    </property>
    <property name="password">
        <value></value> <!--根据自己的数据库密码-->
    </property>
</bean>
```

3.数据库添加用户表及数据（这里用的 mysql），比如在 mysql 数据库中有 t_user 表

4.增加 jar 包，cas-client-core-3.2.1.jar、cas-server-core-3.5.0.jar、cas-server-support-jdbc-3.5.0.jar 包拷贝到 apache-tomcat-7.0.2\webapps\cas\WEB-INF\lib 目录下。

5.重启 Tomcat，打开浏览器，输入 http://jeesz.cn:8080/，输入数据库里的用户名和密码，如果出现如下界面，则配置成功。

现在我们的 CAS 服务端已经配置好了，接下来，我们配置客户端

第二节：配置自己的 Web 工程（客户端）

1.在 host 文件下，添加如下代码：

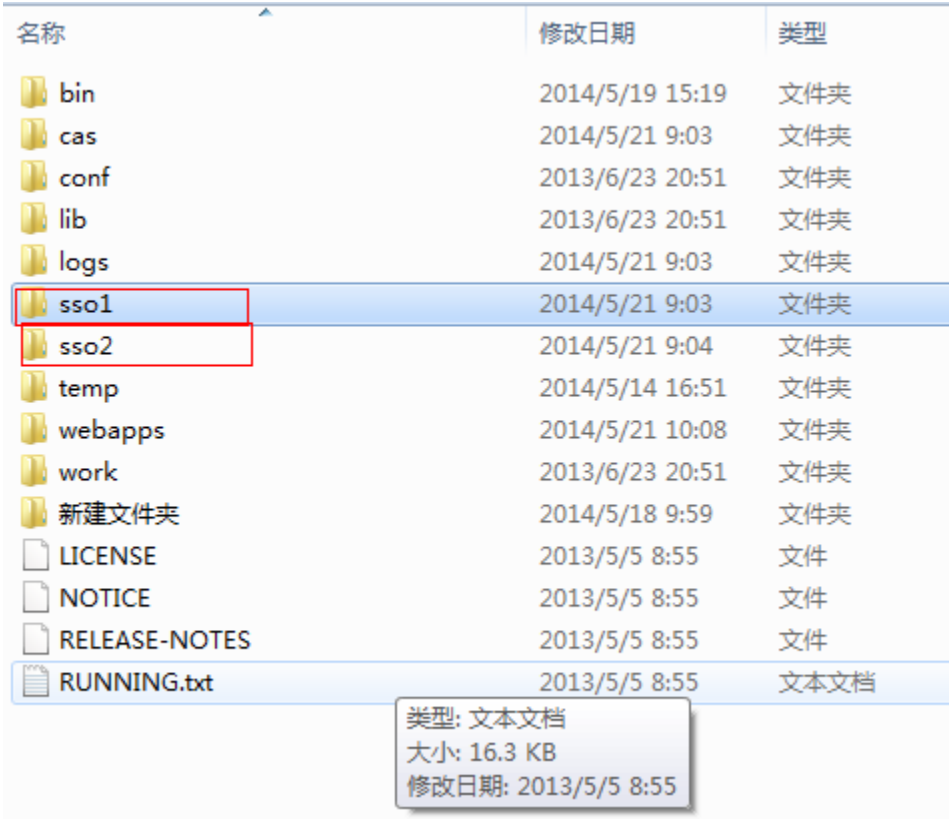
127.0.0.1 www.sso1.com

127.0.0.1 www.sso2.com

注意：这个网址最好不要用互联网已经存在的域名，否则你将无法访问该地址。

如果想在一台 PC 机上模拟这个单点登录，就必须域名重定向，如果是多台ＰＣ机，可以不配置此项，下文有用到 www.sso1.com，www.sso2.com，可以用相应ＰＣ机的ＩＰ代替

1.在 Tomcat 根目录下创建一个 sso1,sso2 目录。如下如：



2 在 eclipse 新建两个 web 工程，分别为 sso1,sso2。

3 在自己的 Web 工程里加入 cas-client-core.jar，commons-logging-1.1.jar，（解压 cas-client-3.2.0-release.zip，在 cas-client-3.2.0-release.zip\cas-client-3.2.0\modules，找到该 JAR 包）分别加入到 sso1,sso2 工程的 lib 里。

4.修改 sso1 下的 web.xml。添加如下代码：

```
<!--SSO 客户端配置 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>SingleSignOutFilter</filter-name>
        <filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>
</filter>
<filter-mapping>
        <filter-name>SingleSignOutFilter</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>CASValidationFilter</filter-name>
        <filter-class>
                org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
        <init-param>
                <param-name>casServerUrlPrefix</param-name>
                <param-value>https://jeesz.cn:8443/cas</param-value>
        </init-param>
        <init-param>
                <param-name>serverName</param-name>
                <param-value>http://www.sso1.com</param-value>
        </init-param>
        <init-param>
                <param-name>useSession</param-name>
                <param-value>true</param-value>
        </init-param>
        <init-param>
                <param-name>exceptionOnValidationFailure</param-name>
                <param-value>false</param-value>
        </init-param>
        <init-param>
                <param-name>redirectAfterValidation</param-name>
                <param-value>true</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CASValidationFilter</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>
            <param-value>https://jeesz.cn:8443/cas/login</param-value>
        </init-param>
        <init-param>
            <param-name>serverName</param-name>
            <param-value>http://www.sso1.com</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>CASFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 允许通过 HttpServletRequest 的 getRemoteUser()方法获得 SSO 登录用户的登录名，可选配置。 -->
    <filter>
        <filter-name>CASHttpServletRequestWrapperFilter</filter-name>
        <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>CASHttpServletRequestWrapperFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 该过滤器可以通过 org.jasig.cas.client.util.AssertionHolder 来获取用户的登录名。 比如
AssertionHolder.getAssertion().getPrincipal().getName()。 -->
    <filter>
        <filter-name>CASAssertionThreadLocalFilter</filter-name>
        <filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>CASAssertionThreadLocalFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 拦截成功登录 SSO 系统之后返回的数据并做相关处理. -->
    <filter>
        <filter-name>SSO4InvokeContextFilter</filter-name>
        <filter-class>com.common.core.busi.other.filter.SSO4InvokeContextFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>SSO4InvokeContextFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

5.修改 sso2 下的 web.xml。添加如下代码：

```
<!--SSO 客户端配置 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>SingleSignOutFilter</filter-name>
    <filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>SingleSignOutFilter</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>CASValidationFilter</filter-name>
    <filter-class>
        org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
    <init-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>https://jeesz.cn:8443/cas</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://www.sso2.com</param-value>
    </init-param>
    <init-param>
        <param-name>useSession</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>exceptionOnValidationFailure</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>redirectAfterValidation</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
```

```
        <filter-name>CASValidationFilter</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>
            <param-value>https://jeesz.cn:8443/cas/login</param-value>
        </init-param>
        <init-param>
            <param-name>serverName</param-name>
            <param-value>http://www.sso2.com</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>CASFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 允许通过 HttpServletRequest 的 getRemoteUser()方法获得 SSO 登录用户的登录名，可选配置。 -->
    <filter>
        <filter-name>CASHttpServletRequestWrapperFilter</filter-name>
        <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>CASHttpServletRequestWrapperFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 该过滤器可以通过 org.jasig.cas.client.util.AssertionHolder 来获取用户的登录名。 比如
AssertionHolder.getAssertion().getPrincipal().getName()。 -->
    <filter>
        <filter-name>CASAssertionThreadLocalFilter</filter-name>
        <filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>CASAssertionThreadLocalFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 拦截成功登录 SSO 系统之后返回的数据并做相关处理. -->
```

```
<filter>
    <filter-name>SSO4InvokeContextFilter</filter-name>
    <filter-class>com.common.web.filter.SSO4InvokeContextFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>SSO4InvokeContextFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

6.编辑 sso1,sso2 index.jsp 页面,复制如下代码：

sso1 index.jsp:

```
<h3>这个是 www.sso1.com</h3>
<dl>
    <dt>你好我是 sso1 页面:</dt>
    <h3><%= request.getRemoteUser()== null ? "null" : request.getRemoteUser() %></h3>
</dl>
<br/>
```

Sso2 index.jsp:

```
<h3>这个是 www.sso2.com</h3>
<dl>
    <dt>你好我是 sso2 页面:</dt>
    <h3><%= request.getRemoteUser()== null ? "null" : request.getRemoteUser() %></h3>
</dl>
<br/>
```

7.通过 eclipse 发布到 Tomcat 服务器上去。找到 Tomcat，webapps 下的 sso1,sso2 文件夹，分别复制 sso1 和 sso2 下面的所有文件，找到 Tomcat 根目录下的 sso1,sso2(就是我们前面步骤新建的目录)，在 sso1,sso2 下新建 ROOT 目录，把刚刚复制的文件粘贴。

配置 Tomcat 下 server.xml 文件，加入如下代码：

```
<Host name="www.sso1.com"    appBase="sso1" unpackWARs="true" autoDeploy="true"></Host>
<Host name="www.sso2.com"    appBase="sso2" unpackWARs="true" autoDeploy="true" ></Host>
```

8.重启 Tomcat，打开浏览器输入网址：www.sso1.com:8080，输入用户名与密码，如果出现以下信息，则成功

这个是www.ss2.com

欢迎您:

admin

在浏览器上输入 www.sso1.com:8080，你会发现跳回到登录页面，不要担心。打开 ticketGrantingTicketCookieGenerator.xml 路径在 apache-tomcat-7.0.40\cas\ROOT\WEB-INF\spring-config

uration。找到 p:cookieSecure="true"，将其修改为 p:cookieSecure="false"，重启 Tomcat，测试一下。

注：我们以上步骤返回的只有一个用户名，CAS 服务器默认返回该信息。

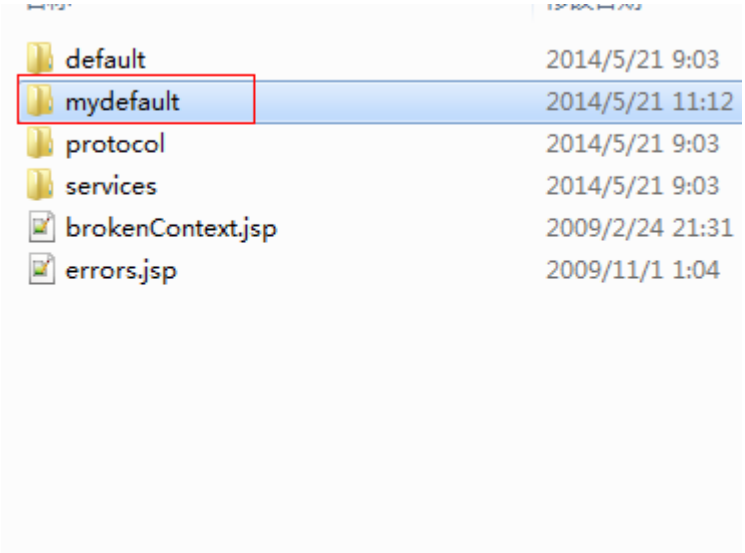
第三节：修改单点登录默认界面

第一步：找到它自己的页面

打开 cas 工程，找到 G:\SoftWare\tomcat\apache-tomcat-7.0.2\apache-tomcat-7.0.2\webapps\cas\WEB-INF\view\jsp\default 这个目录，将其复制一份重命名放在 G:\SoftWare\tomcat\apache-tomcat-7.0.2\apache-tomcat-7.0.2\webapps\cas\WEB-INF\view\jsp\目录下，

第二步：修改默认登录页面

将其命名为 mydefalut 如下图：



然后将 apache-tomcat-7.0.2\webapps\cas\WEB-INF\classes
目录下的 default_views.properties 复制一份，重命名为 mydefault_views.properties。修改里面的内容，将所有里面的路径中的 default 改成 mydefault。然后修改 WEB-INF 下的 cas.properties 将里面的 cas.viewResolver.basename 后面的值修改成 mydefault_views 即：
cas.viewResolver.basename=mydefault_views
如果要修改登录界面，只需要修改 casLoginView.jsp 即可。

第四节：单点登出配置

第一步：配置单点登录监听器和过滤器

1.Tomcat 的 sso1,sso2 的工程里，在 web.xml 里分别添加一下代码：

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

注意：这些代码应该放在 web.xml 文件其他过滤器的最前面。

2.在 sso1,sso2 的 index.jsp 页面添加以下代码：

```
<br/>
<a href = "http://jeesz.cn:8080/logout">单点退出</a>
```

3.重启 Tomcat，访问 http://jeesz.cn:8080/点击单点退出，如果成功，就会出现如下界面：

一般注销是跳到原项目的登录页面，所以我们需要对 CAS 做如下配置：

1. 修改服务端 cas-servlet.xml 配置（apache-tomcat-7.0.40\cas\ROOT\WEB-INF），找到

<bean id="logoutController" class="org.jasig.cas.web.LogoutController" />

增加属性 p:followServiceRedirects="true"

2.修改客户端，sso1,sso2 index.jsp 文件将原来的单点登出的 URL 修改成：

Sso1:http://jeesz.cn:8080/logout?service=http://www.sso1.com:8080

Sso2:http://jeesz.cn:8080/logout?service=http://www.sso2.com:8080

重启 Tomcat，测试正常。

第五节：多项目集成单点登录配置

第一步：单点登录系统与其他项目集成

在 WEB 项目中的 WEB-INF 目录下的 web.xml 文件，添加以下配置.

<!--SSO 客户端配置 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>SingleSignOutFilter</filter-name>
    <filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>SingleSignOutFilter</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>CASValidationFilter</filter-name>
    <filter-class>
        org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
    <init-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>https://jeesz.cn:8443/cas</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://www.sso3.com:6060</param-value>      <!—客户端 URL 地址-->
    </init-param>
    <init-param>
        <param-name>useSession</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>exceptionOnValidationFailure</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>redirectAfterValidation</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>CASValidationFilter</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>
        <param-value>https://jeesz.cn:8443/cas/login</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http:// www.sso3.com:6060</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>CASFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- 允许通过 HttpServletRequest 的 getRemoteUser()方法获得 SSO 登录用户的登录名，可选配置。 -->
<filter>
    <filter-name>CASHttpServletRequestWrapperFilter</filter-name>
    <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>CASHttpServletRequestWrapperFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- 该过滤器可以通过 org.jasig.cas.client.util.AssertionHolder 来获取用户的登录名。 比如
AssertionHolder.getAssertion().getPrincipal().getName()。 -->
<filter>
    <filter-name>CASAssertionThreadLocalFilter</filter-name>
    <filter-class>org.jasig.cas.client.util.AssertionThreadLocalFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>CASAssertionThreadLocalFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- 拦截成功登录 SSO 系统之后返回的数据并做相关处理. -->
<filter>
    <filter-name>SSO4InvokeContextFilter</filter-name>
    <filter-class>com.common.web.filter.SSO4InvokeContextFilter</filter-class>
</filter>
```

```
<filter-mapping>
    <filter-name>SSO4InvokeContextFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

当 sso 验证完成之后，客户端系统需要接收 sso 系统返回的结果时，需要定义一个过滤器获取返回结果，然后针对返回结果做相关处理.

注意：如果不需要做处理时，此处 Filter 也可以不用定义.

```
package com.common.web.filter;
import java.io.IOException;
import java.util.Date;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.jasig.cas.client.util.AssertionHolder;
import org.springframework.web.context.WebApplicationContext;
import org.springframework.web.context.support.WebApplicationContextUtils;
import com.common.base.pbi.CommonConstants;
import com.common.base.util.DateUtil;
import com.common.base.util.IDUtil;
import com.common.base.util.IPUtil;
import com.common.base.util.UserUtil;
import com.common.core.busi.historylogin.manager.HistoryLoginManager;
import com.common.core.busi.login.manager.LoginManager;
import com.common.entity.common.AbstractEntity;
import com.common.entity.historylogin.HistoryLoginEntity;
import com.common.entity.user.UserEntity;

/**
 *  当成功登录 SSO 系统时将会返回登录的 userid 根据此 userid 建立 session 会话;
 * @ClassName: SessionFilter
 * @Description: TODO(这里用一句话描述这个类的作用)
 * @author shaozhen
 * @date 2014-1-23
 *
```

```
 */
public class SSO4InvokeContextFilter implements Filter{
    private final static Log log = LogFactory.getLog(SSO4InvokeContextFilter.class);
    private WebApplicationContext applicationContext;

    public SSO4InvokeContextFilter() {
        super();
    }

    /**
     * 过滤器注销时，触发此方法;
     */
    public void destroy() {
        //暂时不做任何处理;
    }

    /**
     * 根据用户 id 获取用户信息并且把用户信息放入 session 会话中;
     * @Title: doFilter
     * @Description: TODO(这里用一句话描述这个方法的作用)
     * @Params
     * @throws
     */
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws
IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest)req;
        HttpServletResponse response = (HttpServletResponse)res;
        HttpSession session =  request.getSession();
        //从 session 中获取登陆用户;
        Object userObject = session.getAttribute(CommonConstants.USER_INFO_SESSION);
        if(userObject == null){
            //获取用户名;
            String userName = AssertionHolder.getAssertion().getPrincipal().getName();
            LoginManager loginManager = applicationContext.getBean(LoginManager.class);
            UserEntity userEntity = loginManager.loginByUserName(userName);
            session.setAttribute(CommonConstants.USER_INFO_SESSION,userEntity);

    session.setAttribute(CommonConstants.IS_SYSTEM_ADMIN,userEntity.getUserType()==1?tr
ue:false);
            UserUtil.setLoginUserInfo(userEntity);
            //根据用户名查询出用户信息，并放入 session 中;
            log.info("UserName:["+userName  +"] 登 陆 成 功 ， 客 户 端  IP  地 址 为
["+IPUtil.getIpAddr(request)+"]，登陆时间为["+DateUtil.dateToString(new Date())+"]");
            //添加登录记录;
```

```
                HistoryLoginEntity historyLoginEntity = new HistoryLoginEntity();
                historyLoginEntity.setUserId(userName);
                historyLoginEntity.setHid(IDUtil.generateId());
                historyLoginEntity.setLoginCount("1");
                setCommonValue(request,historyLoginEntity);
                boolean hlBol = applicationContext.getBean(HistoryLoginManager.class).addLoginRecord(historyLoginEntity);
                log.debug("登录历史记录["+(hlBol?"成功":"失败")+"].");
            }
            chain.doFilter(request, response);
        }

        /**
         * 设置公共属性;
         * @Title: setCommonValue
         * @Description: TODO(这里用一句话描述这个方法的作用)
         * @throws
         */
        private void setCommonValue(HttpServletRequest request,AbstractEntity entity){
            if(request != null){
                //获取当前对象;
                UserEntity userEntity = (UserEntity) request.getSession().getAttribute(CommonConstants.USER_INFO_SESSION);
                if(entity !=null){
                    String currUser = userEntity.getUserId();
                    //设置创建人、创建日期、修改人、修改时间
                    entity.setCreatedBy(currUser);
                    entity.setModifiedBy(currUser);
                    entity.setCreationDate(DateUtil.getNowDate());
                    entity.setModifiedDate(DateUtil.getNowDate());
                }
            }
        }

        /**
         * 初始化 Spring 上下文;
         */
        @Override
        public void init(FilterConfig filterConfig) throws ServletException {
            WebApplicationContext applicationContext = WebApplicationContextUtils.getWebApplicationContext(filterConfig.getServletContext());
            this.applicationContext = applicationContext;
        }
}
```

至此一个简单的单点登录已经配置好了，如果在配置过程中有任何疑问请联系
技术支持 QQ：2137028325！

文档撰写：邵震
2016-02-25