第7章 单点登录解决方案 CAS

学习目标

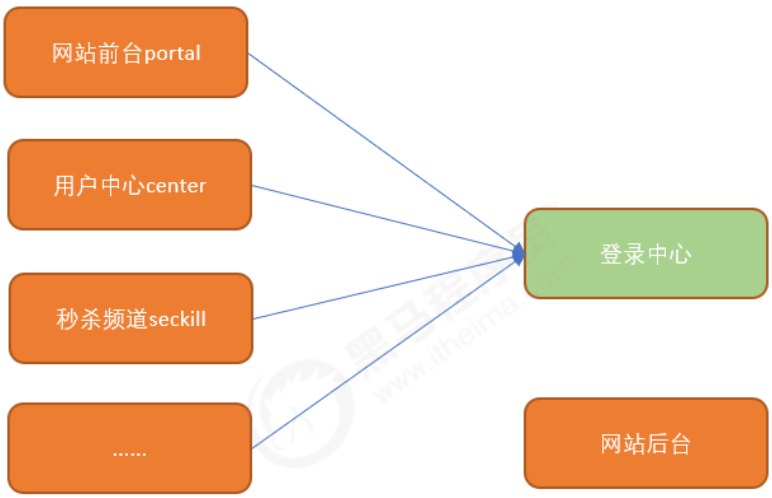
- 能够说出单点登录CAS在系统中的作用。
- 能够完成CAS服务端的部署与客户端的开发
- 能够对CAS服务端进行数据源配置、加密配置与自定义用户界面
- 能够完成用户中心单点登录的功能

1. 开源单点登录系统CAS

1.1 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。

我们目前的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。



当我们访问其中一个前台系统需要登录才可以访问的页面时，自动跳转到登录中心进行登录，登录后再次跳转回该前台系统，该前台系统可以获取登录名。

实现单点登录有多种方案：

（1）使用redis实现session共享

（2）使用开源的单点登录系统（推荐）

1.2 CAS简介

CAS（Central Authentication Service的缩写，中央认证服务）是耶鲁大学Technology and Planning实验室的Shawn Bayern 在2002年出的一个开源系统。刚开始名字叫Yale CAS。 Yale CAS 1.0的目标只是一个单点登录的系统，随着慢慢用开，功能就越来越多了，2.0就提供了多种认证的方式。

2004年12月，CAS转成JASIG(Java Administration Special Interesting Group)的一个项目，项目也随着改名为 JASIG CAS，这就是为什么现在有些CAS的链接还是有jasig的字样。

2012年，JASIG跟Sakai基金会合并，改名为Apereo基金会，所有CAS也随着改名为Apereo CAS。

cas官网： https://www.apereo.org/projects/cas

源码地址： https://github.com/apereo/cas/tree/5.3.x

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。

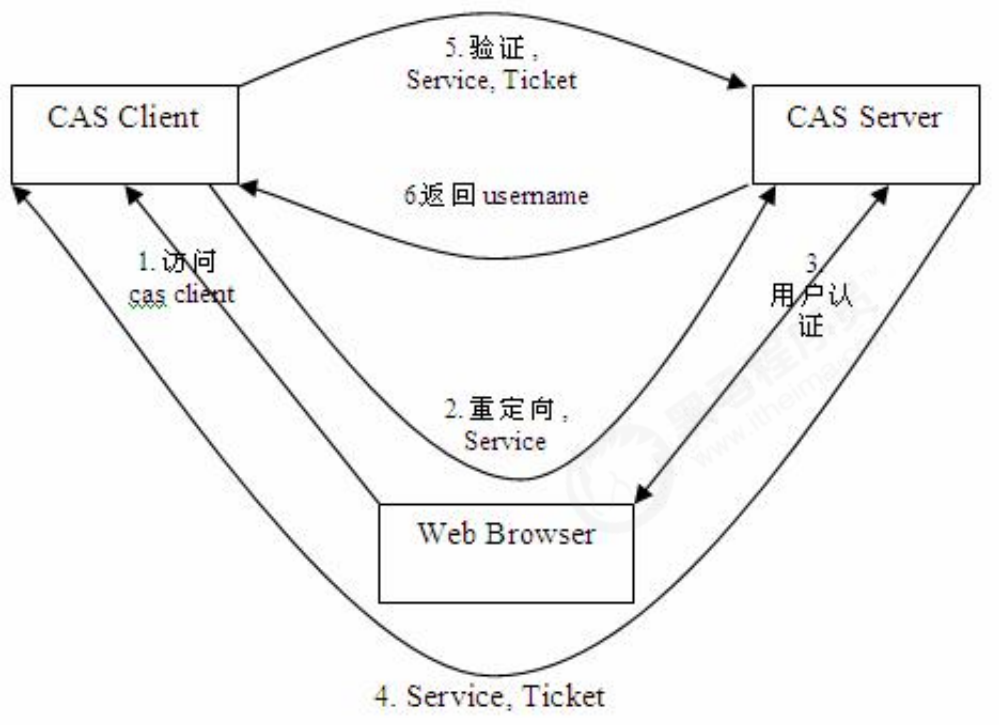
【3】CAS Client 支持非常多的客户端(这里指单点登录系统中的各个 Web 应用)，包括 Java, .Net, PHP, Perl,Apache, uPortal, Ruby 等。

从结构上看，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。

下图是 CAS 最基本的协议过程：



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ServiceTicket。

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。

2. CAS快速入门

2.1 CAS服务端部署

2.1.1 打包

（1）将cas-overlay-template-5.3.zip解压（资料中提供）

（2）修改pom.xml，添加配置

```
<!--数据库认证相关 start-->
<dependency>
    <groupId>org.apereo.cas</groupId>
    <artifactId>cas-server-support-jdbc</artifactId>
    <version>${cas.version}</version>
</dependency>
<dependency>
    <groupId>org.apereo.cas</groupId>
    <artifactId>cas-server-support-jdbc-drivers</artifactId>
    <version>${cas.version}</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.32</version>
</dependency>
<!--数据库认证相关 end-->
```

（3）进入cas-overlay-template-5.3目录，执行命令

```
mvn package
```

（4）把target下生成的war包重命名为cas.war放到tomcat　启动tomcat后自动解压war包

（5）打开浏览器输入 http://localhost:8080/cas ， 系统有内置的用户名和密码分别是 casuser 和 Mellon，输入后登录验证成功.

（6）由于使用的静态密码为casuser::Mellon不好记忆, 修改密码---　到apache-tomcat-8.5.33\webapps\cas\WEB-INF\classes目录的application.properties的最后面修改密码为如下

```
cas.authn.accept.users=admin::admin
```

2.1.2 配置兼容http协议

（1）由于CAS默认使用的是基于https协议,需要改为兼容使用http协议 到apache-tomcat-8.5.33\webapps\cas\WEB-INF\classes目录的application.properties添加如下的内容

```
cas.tgc.secure=false
cas.serviceRegistry.initFromJson=true
```

TGC：Ticket Granted Cookie（客户端用户持有，传送到服务器，用于验证）

存放用户身份认证凭证的cookie，在浏览器和CAS Server间通讯时使用，并且只能基于安全通道传输（Https），是CAS Server用来明确用户身份的凭证。

（2）到apache-tomcat-8.5.33\webapps\cas\WEB-INF\classes\services目录下的HTTPSandIMAPS-10000001.json 修改内容如下,即添加http

```
"serviceId" : "^(https|http|imaps)://.*",
```

修改以上配置后，启动cas服务端进行测试。

2.2 CAS客户端开发

2.2.1 原生方式

（1）新建demo工程，pom.xml 添加配置

```
    <packaging>war</packaging>
    <dependencies>
        <dependency>
            <groupId>org.jasig.cas.client</groupId>
            <artifactId>cas-client-core</artifactId>
            <version>3.3.3</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <configuration>
                    <!-- 指定端口 -->
                    <port>9001</port>
                    <!-- 请求路径 -->
                    <path>/</path>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

（2）新建web.xml

```
<?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 -->
    <listener>
        <listener-
class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</list
ener-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>
            <param-value>http://localhost:8080/cas/login</param-value>
            <!--这里的server是服务端的IP -->
        </init-param>
        <init-param>
            <param-name>serverName</param-name>
            <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>
            <param-value>http://localhost:8080/cas</param-value>
        </init-param>
        <init-param>
            <param-name>serverName</param-name>
            <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>
    <!-- ========================= 单点登录结束 ========================= --
>
</web-app>
```

（3）新增index.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>青橙</title>
</head>
<body>
<%=request.getRemoteUser()%> ，欢迎来到青橙团购
</body>
</html>
```

（4）测试： 启动cas服务端和客户端，浏览器输入http://localhost:9001 ，系统自动跳转到服务端的登录页面，输入正确的用户名和密码后，跳转回客户端工程。

（5）再创建一个一样的cas客户端程序（青橙秒杀），端口为9002，测试两个系统单点登录的效果。即登录其中一个系统，另外一个系统则不用登录。

2.2.2 与Spring Security整合

（1）创建 demo工程，pom.xml引入依赖（资源中提供）

```
    <packaging>war</packaging>
    <properties>
        <spring.version>5.0.5.RELEASE</spring.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-config</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-cas</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.jasig.cas.client</groupId>
            <artifactId>cas-client-core</artifactId>
            <version>3.3.3</version>
            <exclusions>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>log4j-over-slf4j</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <configuration>

                    <!-- 指定端口 -->
```

```
                    <port>9003</port>
                    <!-- 请求路径 -->
                    <path>/</path>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

（2）添加web.xml （资源中提供）

```
<?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">

     <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring-security.xml</param-value>
     </context-param>
     <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
     </listener>
     <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
     </filter>
     <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
     </filter-mapping>

</web-app>
```

（3）添加spring-security.xml （资源中提供）

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">

    <!--entry-point-ref  入口点引用 -->
    <http use-expressions="false" entry-point-
ref="casProcessingFilterEntryPoint">
        <intercept-url pattern="/**" access="ROLE_USER"/>
        <csrf disabled="true"/>
        <!-- custom-filter为过滤器， position 表示将过滤器放在指定的位置上，
before表示放在指定位置之前  ，after表示放在指定的位置之后  -->
        <custom-filter ref="casAuthenticationFilter"
position="CAS_FILTER" />
        <custom-filter ref="requestSingleLogoutFilter"
before="LOGOUT_FILTER"/>
        <custom-filter ref="singleLogoutFilter" before="CAS_FILTER"/>
    </http>

    <!-- CAS入口点 开始 -->
    <beans:bean id="casProcessingFilterEntryPoint"
class="org.springframework.security.cas.web.CasAuthenticationEntryPoint">
        <!-- 单点登录服务器登录URL -->
        <beans:property name="loginUrl"
value="http://localhost:8080/cas/login"/>
        <beans:property name="serviceProperties"
ref="serviceProperties"/>
    </beans:bean>
    <beans:bean id="serviceProperties"
class="org.springframework.security.cas.ServiceProperties">
        <!--service 配置自身工程的根地址+/login/cas   -->
        <beans:property name="service"
value="http://localhost:9003/login/cas"/>
    </beans:bean>
    <!-- CAS入口点 结束 -->
```

```
    <!-- 认证过滤器 开始 -->
    <beans:bean id="casAuthenticationFilter"
class="org.springframework.security.cas.web.CasAuthenticationFilter">
        <beans:property name="authenticationManager"
ref="authenticationManager"/>
    </beans:bean>
        <!-- 认证管理器 -->
    <authentication-manager alias="authenticationManager">
        <authentication-provider  ref="casAuthenticationProvider">
        </authentication-provider>
    </authentication-manager>
        <!-- 认证提供者 -->
    <beans:bean id="casAuthenticationProvider"
class="org.springframework.security.cas.authentication.CasAuthenticationP
rovider">
        <beans:property name="authenticationUserDetailsService">
            <beans:bean
class="org.springframework.security.core.userdetails.UserDetailsByNameSer
viceWrapper">
                <beans:constructor-arg ref="userDetailsService" />
            </beans:bean>
        </beans:property>
        <beans:property name="serviceProperties"
ref="serviceProperties"/>
        <!-- ticketValidator 为票据验证器 -->
        <beans:property name="ticketValidator">
            <beans:bean
class="org.jasig.cas.client.validation.Cas20ServiceTicketValidator">
                <beans:constructor-arg index="0"
value="http://localhost:8080/cas"/>
            </beans:bean>
        </beans:property>
        <beans:property name="key"
value="an_id_for_this_auth_provider_only"/>
    </beans:bean>
         <!-- 认证类 -->
    <beans:bean id="userDetailsService"
class="cn.itcast.demo.service.UserDetailServiceImpl"/>

    <!-- 认证过滤器 结束 -->
```

```
    <!-- 单点登出  开始  -->
    <beans:bean id="singleLogoutFilter"
class="org.jasig.cas.client.session.SingleSignOutFilter"/>
    <beans:bean id="requestSingleLogoutFilter"
class="org.springframework.security.web.authentication.logout.LogoutFilte
r">
        <beans:constructor-arg value="http://localhost:8080/cas/logout"/>
        <beans:constructor-arg>
            <beans:bean
class="org.springframework.security.web.authentication.logout.SecurityCon
textLogoutHandler"/>
        </beans:constructor-arg>
        <beans:property name="filterProcessesUrl" value="/logout/cas"/>
    </beans:bean>
    <!-- 单点登出  结束 -->
</beans:beans>
```

（4）添加UserDetailServiceImpl

```
package cn.itcast.demo.service;
import org.springframework.security.core.GrantedAuthority;
import
org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import java.util.ArrayList;
import java.util.List;

public class UserDetailServiceImpl implements UserDetailsService {

    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        List<GrantedAuthority> authorityList=new
ArrayList<GrantedAuthority>();
        authorityList.add(new SimpleGrantedAuthority("ROLE_USER"));
        return new User(username,"",authorityList);
    }
}
```

（5）添加index.jsp

```
<%@ page
import="org.springframework.security.core.context.SecurityContextHolder"
%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>青橙用户中心</title>
</head>
<body>
<%=SecurityContextHolder.getContext().getAuthentication().getName()%>  欢
迎来到青橙用户中心  <a href="/logout/cas">退出</a>
</body>
</html>
```

2.2.3 配置文件详解

use-expressions：是否启动SPEL表达式 默认是true。

entry-point-ref：security框架接入第三方服务的通用入口。

casEntryPoint中的url 是CAS Server用来输入用户凭证的入口，其中主机和端口配置成提供CAS Server服务的主机和端口即可。

serviceProperties中value的配置是web服务器的主机+/login/cas，该URL是service的值，会依据该值在CAS Server上生成ticket, 然后用来到WEB服务器上验证ticket。

requestSingleLogoutFilter用来指定登出操作的过滤器

3. 青橙-认证中心

3.1 数据源与加密设置

3.1.1 数据源设置

我们需要改变服务端的配置，从数据库中获取用户名和密码。设置方法：

修改application.properties，将 `cas.authn.accept.users=....` 加井号注释掉，再加入以下内容：

```
cas.authn.jdbc.query[0].url=jdbc:mysql://localhost:3306/qingcheng_user?
useUnicode=true&characterEncoding=UTF-8&autoReconnect=true&useSSL=false
cas.authn.jdbc.query[0].user=root
cas.authn.jdbc.query[0].password=123456
cas.authn.jdbc.query[0].sql=select password from tb_user where username=?
cas.authn.jdbc.query[0].fieldPassword=password
cas.authn.jdbc.query[0].driverClass=com.mysql.jdbc.Driver
```

3.1.2 加密配置

cas5.X 提供了4种加密配置：

```
cas.authn.jdbc.query[0].passwordEncoder.type= NONE|DEFAULT|STANDARD|BCRYPT
```

默认值为 NONE

这四种方式其实脱胎于spring security中的加密方式，spring security提供了MD5PasswordEncoder、SHAPasswordEncoder、StandardPasswordEncoder和BCryptPasswordEncoder。

NONE：说明对密码不做任何加密，也就是保留明文。

DEFAULT：启用DefaultPasswordEncoder。 MD5PasswordEncoder和SHAPasswordEncoder加密是编码算法加密。现在cas把他们归属于DefaultPasswordEncoder。DefaultPasswordEncoder需要带参数encodingAlgorithm：

```
cas.authn.accept.passwordEncoder.encodingAlgorithm=MD5|SHA
```

STANDARD：启用StandardPasswordEncoder加密方式 。1024次迭代的`SHA-256`散列哈希加密实现，并使用一个随机8字节的salt。

BCRYPT：启用BCryptPasswordEncoder加密方式。

我们在上实现用户注册功能

修改application.properties 配置文件

```
cas.authn.jdbc.query[0].passwordEncoder.type=BCRYPT
cas.authn.jdbc.query[0].passwordEncoder.characterEncoding=UTF-8
```

3.2 自定义用户界面

3.2.1 需求分析

将CAS的服务端页面更改为以下样式

3.2.2 代码实现

参考文档：

https://apereo.github.io/cas/5.3.x/installation/Configuration-Properties.html#themes

规范：

- 静态资源(js,css)存放目录为WEB-INF\classes\static
- html资源(thymeleaf模板)存放目录为WEB-INF\classes\templates
- 主题配置文件存放在WEB-INF\classes，并且命名为[theme_name].properties

（1）在apache-tomcat-8.5.37\webapps\cas\WEB-INF\classes\static\themes目录下创建qingcheng文件夹，将静态原型中的img和css文件夹拷入此文件夹

（2）在apache-tomcat-8.5.37\webapps\cas\WEB-INF\classes\templates目录下创建qingcheng文件夹，将静态原型中的login.html拷贝至此文件夹，并改名为casLoginView.html, 修改casLoginView.html内容

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
```

样式部分，改成绝对目录

```
<link rel="stylesheet" type="text/css"
href="/cas/themes/qingcheng/css/all.css" />
<link rel="stylesheet" type="text/css"
href="/cas/themes/qingcheng/css/pages-login.css" />
```

图片部分，改成绝对目录

```
<li><img src="/cas/themes/qingcheng/img/qq.png" width="35px"
height="35px" /></li>
<li><img src="/cas/themes/qingcheng/img/sina.png" /></li>
<li><img src="/cas/themes/qingcheng/img/ali.png" /></li>
<li><img src="/cas/themes/qingcheng/img/weixin.png" /></li>
```

表单部分：

```
<form class="sui-form" method="post" th:object="${credential}">
    <div class="input-prepend"><span class="add-on loginname"></span>
        <input id="inputName" type="text" placeholder="邮箱/用户名/手机号"
th:field="*{username}" class="span2 input-xfat">
    </div>
    <div class="input-prepend"><span class="add-on loginpwd"></span>
        <input id="inputPassword" type="password" placeholder="请输入密码"
th:field="*{password}" class="span2 input-xfat">
    </div>
    <div class="setting">
        <label class="checkbox inline">
            <input name="m1" type="checkbox" value="2" checked="">
            自动登录
        </label>
        <span class="forget">忘记密码？</span>
    </div>
    <div class="logined">
        <input type="hidden" name="execution"
th:value="${flowExecutionKey}" />
        <input type="hidden" name="_eventId" value="submit" />
        <input type="hidden" name="geolocation" />
        <button class="sui-btn btn-block btn-xlarge btn-danger">
登  录</button>
        <div th:if="${#fields.hasErrors('*')}">
            <span th:each="err : ${#fields.errors('*')}"
th:utext="${err}" style="color: red" />
        </div>
    </div>
</form>
```

th:object，用法：th:object="${brand}"，(用来接受后台传过来的对象)

th:field，用法：th:field="*{name}"，(用来绑定后台对象和表单数据)

***{}** 选择表达式（星号表达式）选择表达式与变量表达式有一个重要的区别：选择表达式计算的是选定的对象，而不是整个环境变量映射。也就是：只要是没有选择的对象，选择表达式与变量表达式的语法是完全一样的。那什么是选择的对象呢？是一个：th:object对象属性绑定的对象

（3）在apache-tomcat-8.5.37\webapps\cas\WEB-INF\classes下创建qingcheng.properties

```
cas.standard.css.file=/css/cas.css
```

（4）修改apache-tomcat-8.5.37\webapps\cas\WEB-INF\classes\application.properties

```
cas.theme.defaultThemeName=qingcheng
```

4. 青橙-单点登录

4.1 需求分析

新建用户中心，与认证中心对接实现单点登录。

将portal工程集成单点登录。

4.2 代码实现

4.2.1 CAS公共模块

（1）创建qingcheng_common_cas工程（ cas公共模块），pom.xml引入依赖

```
<dependency>
    <groupId>com.qingcheng</groupId>
    <artifactId>qingcheng_common_web</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-cas</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.9.4</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.4</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.9.4</version>
</dependency>
```

（2）qingcheng_common_cas工程添加配置文件spring-security.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
                        http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">

    <context:property-placeholder location="classpath*:*.properties" />

    <http pattern="/css/**" security="none"></http>
    <http pattern="/js/**" security="none"></http>
    <http pattern="/img/**" security="none"></http>

    <!--   entry-point-ref  入口点引用 -->
    <http use-expressions="false" entry-point-
ref="casProcessingFilterEntryPoint">
        <intercept-url pattern="/**" access="ROLE_USER"/>
        <csrf disabled="true"/>
        <!-- custom-filter为过滤器， position 表示将过滤器放在指定的位置上，
before表示放在指定位置之前  ，after表示放在指定的位置之后  -->
        <custom-filter ref="casAuthenticationFilter"
position="CAS_FILTER" />
        <custom-filter ref="requestSingleLogoutFilter"
before="LOGOUT_FILTER"/>
        <custom-filter ref="singleLogoutFilter" before="CAS_FILTER"/>
    </http>

    <!-- CAS入口点 开始 -->
    <beans:bean id="casProcessingFilterEntryPoint"
class="org.springframework.security.cas.web.CasAuthenticationEntryPoint">
        <!-- 单点登录服务器登录URL -->
        <beans:property name="loginUrl" value="${cas_url}/login"/>


        <beans:property name="serviceProperties"
```

```
ref="serviceProperties"/>
    </beans:bean>
    <beans:bean id="serviceProperties"
class="org.springframework.security.cas.ServiceProperties">
        <!--service 配置自身工程的根地址+/login/cas   -->
        <beans:property name="service" value="${service_url}/login/cas"/>
    </beans:bean>
    <!-- CAS入口点 结束 -->


    <!-- 认证过滤器 开始 -->
    <beans:bean id="casAuthenticationFilter"
class="org.springframework.security.cas.web.CasAuthenticationFilter">
        <beans:property name="authenticationManager"
ref="authenticationManager"/>
    </beans:bean>
        <!-- 认证管理器 -->
    <authentication-manager alias="authenticationManager">
        <authentication-provider  ref="casAuthenticationProvider">
        </authentication-provider>
    </authentication-manager>
        <!-- 认证提供者 -->
    <beans:bean id="casAuthenticationProvider"
class="org.springframework.security.cas.authentication.CasAuthenticationP
rovider">
        <beans:property name="authenticationUserDetailsService">
            <beans:bean
class="org.springframework.security.core.userdetails.UserDetailsByNameSer
viceWrapper">
                <beans:constructor-arg ref="userDetailsService" />
            </beans:bean>
        </beans:property>
        <beans:property name="serviceProperties"
ref="serviceProperties"/>
        <!-- ticketValidator 为票据验证器 -->
        <beans:property name="ticketValidator">
            <beans:bean
class="org.jasig.cas.client.validation.Cas20ServiceTicketValidator">
                <beans:constructor-arg index="0" value="${cas_url}"/>
            </beans:bean>
        </beans:property>
```

```
        <beans:property name="key"
value="an_id_for_this_auth_provider_only"/>
    </beans:bean>
         <!-- 认证类 -->
    <beans:bean id="userDetailsService"
class="com.qingcheng.service.UserDetailServiceImpl"/>

    <!-- 认证过滤器 结束 -->

    <!-- 单点登出  开始  -->
    <beans:bean id="singleLogoutFilter"
class="org.jasig.cas.client.session.SingleSignOutFilter"/>
    <beans:bean id="requestSingleLogoutFilter"
class="org.springframework.security.web.authentication.logout.LogoutFilte
r">
        <beans:constructor-arg value="${cas_url}/logout"/>
        <beans:constructor-arg>
            <beans:bean
class="org.springframework.security.web.authentication.logout.SecurityCon
textLogoutHandler"/>
        </beans:constructor-arg>
        <beans:property name="filterProcessesUrl" value="/logout/cas"/>
    </beans:bean>
    <!-- 单点登出  结束 -->

</beans:beans>
```

（3）qingcheng_common_cas工程创建UserDetailServiceImpl

```
public class UserDetailServiceImpl implements UserDetailsService {

    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        System.out.println("....UserDetailServiceImpl");
        //这个类没有校验用户密码的功能  ,给当前用户赋予权限
        List<GrantedAuthority> grantedAuthorityList=new ArrayList();
        grantedAuthorityList.add(new
SimpleGrantedAuthority("ROLE_USER"));
        return new User(username,"",grantedAuthorityList);
    }
}
```

4.2.2 搭建用户中心

（1）创建qingcheng_web_center（用户中心），pom.xml引入依赖

```
<dependency>
    <groupId>com.qingcheng</groupId>
    <artifactId>qingcheng_interface</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
    <groupId>com.qingcheng</groupId>
    <artifactId>qingcheng_common_cas</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

（2）新增web.xml，参照其它web工程 ，并添加spring-security配置

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath*:spring-security.xml</param-value>
</context-param>
<listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

（3）新增配置文件dubbo.properties

```
dubbo.application=center
```

（4）添加配置文件cas.properties

```
cas_url=http://localhost:8080/cas
service_url=http://localhost:9103
```

（5）将资源提供得用户中心相关页面拷贝至webapp下

4.2.3 用户中心显示登录名

（1）在qingcheng_common_cas工程新增类

```
@RestController
@RequestMapping("/login")
public class LoginController {

    /**
     * 获取用户名
     * @return
     */
    @RequestMapping("/username")
    public Map username(){
        String name =
SecurityContextHolder.getContext().getAuthentication().getName();//得到登
录人账号
        if("anonymousUser".equals(name)){
            name="";
        }
        System.out.println(name);
        Map map=new HashMap();
        map.put("username", name);
        return map;
    }
}
```

浏览器输入地址 http://localhost:9103/login/username.do 可看到返回结果：

```
{"username":"lijialong"}
```

（2）前端代码：

修改center-index.html，添加js代码

```
<script src="js/vue.js"></script>
<script src="js/axios.js"></script>
<script>
    new Vue({
        el: '#app',
        data(){
            return {
                username:''
            }
        },
        created(){
            axios.get(`/login/username.do`).then(response => {
                this.username=response.data.username;
            });
        }
    })
</script>
```

修改用户名显示部分

```
<span class="name">{{username}}</span>
```

4.2.4 portal工程集成单点登录

（1）qingcheng_portal的web.xml添加

```
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath*:spring-security.xml</param-value>
  </context-param>
  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  <filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

（2）引入依赖

```
<dependency>
    <groupId>com.qingcheng</groupId>
    <artifactId>qingcheng_common_cas</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

（3）添加配置文件cas.properties

```
cas_url=http://localhost:8080/cas
service_url=http://localhost:9102
```

（4）添加配置文件spring-security-portal.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:beans="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
                        http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">

    <http pattern="/index.do" security="none"></http>
    <http pattern="/search.do" security="none"></http>

</beans:beans>
```

4.2.5 首页显示登录名

（1）修改模板页面

```
<script src="js/vue.js"></script>
<script src="js/axios.js"></script>
<script>
    new Vue({
        el: '#app',
        data(){
            return {
                username:''
            }
        },
        created(){
            axios.get(`/login/username.do`).then(response => {
                this.username=response.data.username;
            });
        }
    })
</script>
```

添加表达式

```
<span v-if="username==''">
    请<a href="login.html" >登录</a>
</span>
<span v-if="username!=''">
    {{username}}<a href="/logout/cas" >退出登录</a>
</span>
```

（2）修改配置文件spring-security.xml，配置/login/username.do可以匿名访问

```
<intercept-url pattern="/login/username.do"
access="IS_AUTHENTICATED_ANONYMOUSLY"></intercept-url>
```

（3）修改LoginController的username方法

```
    /**
     * 获取用户名
     * @return
     */
    @GetMapping("/username")
    public Map username(){
        String username=
SecurityContextHolder.getContext().getAuthentication().getName();

        System.out.println("当前登录用户："+username);
        if("anonymousUser".equals(username)){ //未登录
            username="";
        }
        Map map=new HashMap();
        map.put("username",username);
        return map;
    }
```

4.2.6 退出登录返回首页

修改cas服务端的配置文件 application.properties，添加配置

```
cas.logout.redirectUrl=http://localhost:9102/index.do
```

附录A. Spring Security 内置过滤器表

别名	**Filter**类
CHANNEL_FILTER	ChannelProcessingFilter
SECURITY_CONTEXT_FILTER	SecurityContextPersistenceFilter
CONCURRENT_SESSION_FILTER	ConcurrentSessionFilter
LOGOUT_FILTER	LogoutFilter
X509_FILTER	X509AuthenticationFilter
PRE_AUTH_FILTER	AstractPreAuthenticatedProcessingFilter 的子类
CAS_FILTER	CasAuthenticationFilter
FORM_LOGIN_FILTER	UsernamePasswordAuthenticationFilter
BASIC_AUTH_FILTER	BasicAuthenticationFilter
SERVLET_API_SUPPORT_FILTER	SecurityContextHolderAwareRequestFilter
JAAS_API_SUPPORT_FILTER	JaasApiIntegrationFilter
REMEMBER_ME_FILTER	RememberMeAuthenticationFilter
ANONYMOUS_FILTER	AnonymousAuthenticationFilter
SESSION_MANAGEMENT_FILTER	SessionManagementFilter
EXCEPTION_TRANSLATION_FILTER	ExceptionTranslationFilter
FILTER_SECURITY_INTERCEPTOR	FilterSecurityInterceptor
SWITCH_USER_FILTER	SwitchUserFilter