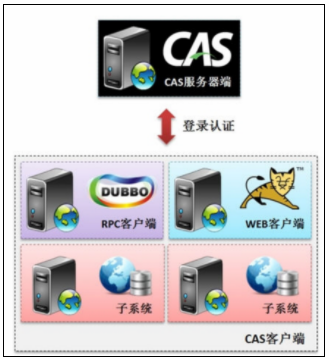
CAS使用简介

# 1、CAS基础配置

1. 单点登录

单点登录（Single Sign On，简称SSO）是目前比较流行的服务于企业业务整合的解决方案之一，SSO使得在多个应用系统中，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。



1. CAS（Central Authentication Service）

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（<http://www.jasig.org>）。

CAS具有以下特点：

* 开源的企业级单点登录解决方案；
* CAS Server为需要独立部署的Web应用；
* CAS Client支持非常多的客户端（值Web应用），包括Java、.Net、PHP、Perl、Ruby等；

1. CAS原理及协议

一旦你的项目之中使用了CAS，那么CAS将作为服务器端，而WEB容器（Tomcat）将作为客户端出现。

1. 普通的用户通过CAS客户端（Tomcat）进行WEB的访问；
2. 而后WEB端会将用户的请求转发给CAS服务器端；
3. 跳转到CAS服务器端之后用户可以输入用户名和密码进行系统登录；
4. 如果认证通过该请求会返回到WEB端，而后WEB端会接收一个CAS端生成的票根（ticket）数据，并且使用此数据进行CAS认证检测；
5. 票根（ticket）检测通过后会将CAS服务器端的用户名发送回WEB端。

清楚了CAS的流程之后下面来进行CAS服务器的配置。本次使用的CAS版本为“cas-server-webapp-4.1.7.war”，为了方便处理，建议将此文件更名为cas.war。其中war文件可以直接上传到Tomcat中。

1. 把cas.war拷贝至tomcat的webapps目录之中；
2. 启动tomcat，此时cas.war文件会在Tomcat启动之后自动解压缩为一个cas的目录；



1. cas默认了一个用户名：casuser/Mellon；



# 2、使用MySQL验证用户名和密码

1. cas服务器需要进行mysql数据库的链接，所以将c3p0、mysql-jdbc、mchange-commons-java、cas-server-support-jdbc组件包引入到cas的lib目录下；
2. 修改cas配置文件：deployerConfigContext.xml；

配置数据库连接池：

|  |
| --- |
| <!-- 配置使用的数据库连接池类型，同时设置卸载方法为close() --> <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">  <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/mohist\_examples"/>  <property name="driverClass" value="com.mysql.jdbc.Driver"/>  <property name="user" value="root"/>  <property name="password" value=""/>  <property name="maxPoolSize" value="1"/>  <property name="minPoolSize" value="1"/>  <property name="initialPoolSize" value="1"/>  <property name="maxIdleTime" value="20"/> </bean> |

增加数据库检测认证器

|  |
| --- |
| <!-- 增加数据库检测认证器 --> <bean id="dbAuthHandler" class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler">  <property name="dataSource" ref="dataSource"/>  <property name="sql" value="SELECT password FROM t\_mohist\_user WHERE username=? AND locked=0"/> </bean> |

修改认证授权

|  |
| --- |
| <!-- 原始配置 --> <entry key-ref="primaryAuthenticationHandler" value-ref="primaryPrincipalResolver" /> |
| <!-- 新配置项 --> <entry key-ref="dbAuthHandler" value-ref="primaryPrincipalResolver"/> |

# 3、在CAS中使用Sha256Hash加密处理

1. 需要为CAS做一个新的密码加密器；

CAS的密码加密器有自己的一个专属实现类，而这个实现类在“cas-server-core-api-4.1.7.jar”包里面；

|  |
| --- |
| package com.kuhnwei.mohist.examples.cas.util.encoder; import org.apache.shiro.crypto.hash.Sha256Hash; import org.jasig.cas.authentication.handler.PasswordEncoder; */\*\*  \* 自定义的密码加密器  \** ***@author*** *Kuhn Wei, email@kuhnwei.com  \** ***@version*** *2018/3/30 19:46  \*/* public class CustomPasswordEncoder implements PasswordEncoder {  @Override  public String encode(String passwrod) {  String salt = "mohist-examples-cas-server";  passwrod = new Sha256Hash(passwrod, salt, 1024).toHex();  return passwrod;  } } |

1. 编辑deployerConfigContext.xml配置文件，引入自定义的加密器类操作；

先定义密码加密处理的bean生命

|  |
| --- |
| <bean id="passwordEncoder" class="com.kuhnwei.mohist.examples.cas.util.encoder.CustomPasswordEncoder"/> |

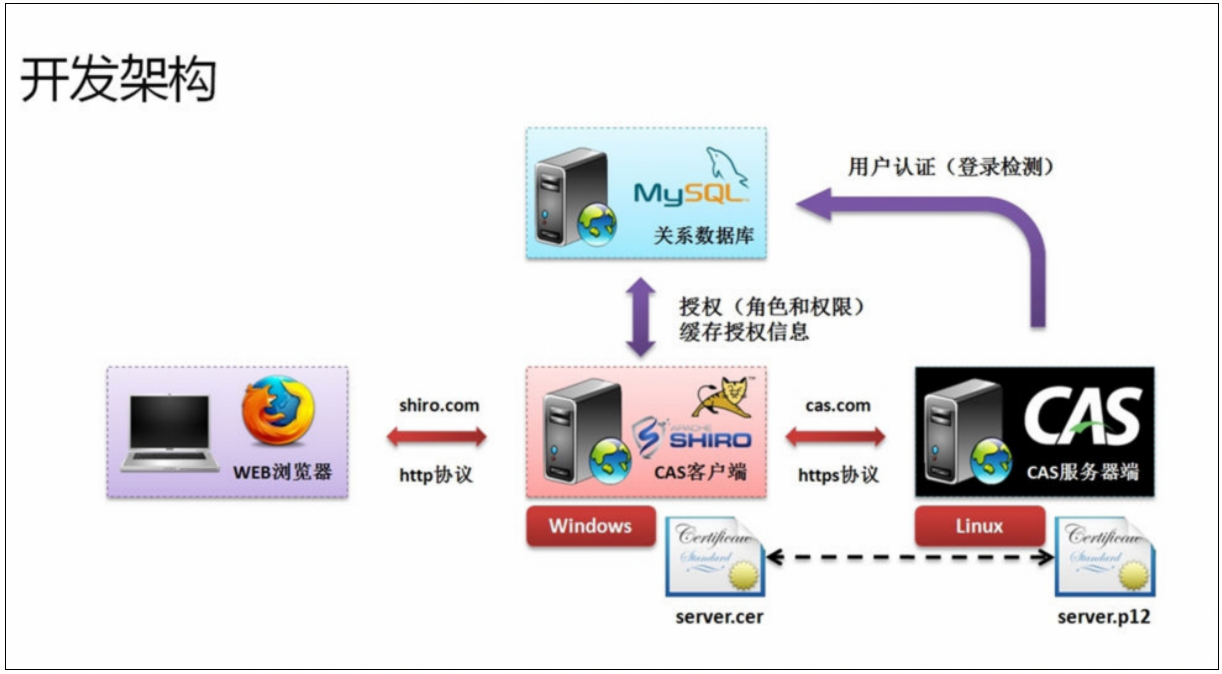
修改定义好的数据库认证配置

|  |
| --- |
| <bean id="dbAuthHandler" class="org.jasig.cas.adaptors.jdbc.QueryDatabaseAuthenticationHandler">  <property name="dataSource" ref="dataSource"/>  <property name="sql" value="SELECT password FROM t\_mohist\_user WHERE username=? AND locked=0"/>  <property name="passwordEncoder" ref="passwordEncoder"/> </bean> |

# 4、Shiro整合CAS认证处理

Shiro一定是先做最为流行的权限认证管理框架，但是现在已经不再是最为传统的shiro处理了，因为此时的处理牵扯到cas实现认证的操作部分，所以对于shiro而言，它依然需要实现认证与授权检测，但是这个时候的认证应该交由cas完成。

在现实的开发过程之中，客户端是不可能直接访问CAS服务器的。



所以需要另起一个整合了shiro的web项目，并引入cas相关的jar包把这个web项目当成cas客户端，在用户认证时调用到cas服务器端进行认证。

## 实现CasRealm

1. 如果要想在项目之中使用cas作为整个认证处理，那么首先需要导入shiro-cas开发包

|  |
| --- |
| <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-cas</artifactId>  <version>1.3.2</version> </dependency> |

1. 此时所有的查询内容都将交由CAS负责，而shiro只是负责角色与权限数据的获取；

实现CasRealm

|  |
| --- |
| package com.kuhnwei.mohist.examples.ssm.shiro.realm;  import com.kuhnwei.mohist.examples.ssm.shiro.domain.UserDO; import com.kuhnwei.mohist.examples.ssm.shiro.service.UserService; import org.apache.shiro.authc.AuthenticationException; import org.apache.shiro.authc.AuthenticationInfo; import org.apache.shiro.authc.AuthenticationToken; import org.apache.shiro.authc.SimpleAuthenticationInfo; import org.apache.shiro.authz.AuthorizationInfo; import org.apache.shiro.authz.SimpleAuthorizationInfo; import org.apache.shiro.cas.CasRealm; import org.apache.shiro.cas.CasToken; import org.apache.shiro.subject.PrincipalCollection; import org.apache.shiro.subject.SimplePrincipalCollection; import org.apache.shiro.util.CollectionUtils; import org.apache.shiro.util.StringUtils; import org.jasig.cas.client.authentication.AttributePrincipal; import org.jasig.cas.client.validation.Assertion; import org.jasig.cas.client.validation.TicketValidator; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import javax.annotation.Resource; import java.util.List; import java.util.Map; import java.util.Objects; import java.util.Set;  */\*\*  \** ***@author*** *Kuhn Wei, email@kuhnwei.com  \** ***@version*** *2018/3/30 21:13  \*/* public class UserCasRealm extends CasRealm {  private Logger logger = LoggerFactory.*getLogger*(getClass());  @Resource  private UserService userService;   @Override  protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {  logger.info("-------------- 用户登录认证 -----------------");  CasToken casToken = (CasToken) token;  if (Objects.*isNull*(casToken)) {  return null;  }  // cas一定要返回给用户一个票根，所以需要取得这个票根的内容  String ticket = (String) casToken.getCredentials();  // 对票根的有效性进行验证  if (!StringUtils.*hasText*(ticket)) {  return null;  }  TicketValidator ticketValidator = super.ensureTicketValidator();  try {  // 对票根做cas验证处理  Assertion casAssertion = ticketValidator.validate(ticket, super.getCasService());  // 通过cas取得用户信息  AttributePrincipal casPrincipal = casAssertion.getPrincipal();  // 取得当前登录的用户名  String username = casPrincipal.getName();  // 取出用户名之后需要将所有的相关信息（包括CAS相关信息）一起进行一个列表的创建  List<Object> principals = CollectionUtils.*asList*(username, casPrincipal.getAttributes());  PrincipalCollection principalCollection = new SimplePrincipalCollection(principals, super.getName());  return new SimpleAuthenticationInfo(principalCollection, ticket);  } catch (Exception e) {  e.printStackTrace();  }  return super.doGetAuthenticationInfo(token);  }   @Override  protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {  logger.info("-------------- 用户权限认证 -----------------");  // 取得用户登录名  String username = (String) principals.getPrimaryPrincipal();  // 定义授权信息的返回数据  SimpleAuthorizationInfo auth = new SimpleAuthorizationInfo();  try {  UserDO user = userService.getByUsername(username);  Map<String, Object> map = userService.listAuthByUserId(user.getId());  Set<String> allRoles = (Set<String>) map.get("allRoles");  Set<String> allPermissions = (Set<String>) map.get("allPermissions");  auth.setRoles(allRoles);  auth.setStringPermissions(allPermissions);  } catch (Exception e) {  e.printStackTrace();  }  return super.doGetAuthorizationInfo(principals);  } } |

此时发现对于使用Realm只有用户认证部分发生了变化，而角色和权限的操作依然像之前那样直接通过业务层读取。

## 4.2、环境配置

1. 修改当前的Realm使用类型

|  |
| --- |
| <bean id="userCasRealm" class="com.kuhnwei.mohist.examples.ssm.shiro.realm.UserCasRealm">  <!-- 配置cas服务器的地址 -->  <property name="casServerUrlPrefix" value="http://localhost:80/cas"/>  <!-- 定义客户端的回调地址 -->  <property name="casService" value="http://localhost:8080/shiro/shiro-cas"/> </bean> |
| <!-- 配置SecurityManager的管理 --> <bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">  <!-- 配置需要使用的Realms -->  <property name="realm" ref="userCasRealm"/>  <!---->  <property name="cacheManager" ref="cacheManager"/>  <!-- 定义要使用的session管理器 -->  <property name="sessionManager" ref="sessionManager"/>  <!-- 定义RememberMe的管理器 -->  <property name="rememberMeManager" ref="rememberMeManager"/> </bean> |

1. 配置cas的访问过滤器

|  |
| --- |
| <bean id="casFilter" class="org.apache.shiro.cas.CasFilter">  <!-- 登录失败后的访问路径 -->  <property name="failureUrl" value="http://localhost:80/cas/login?service=http://localhost:8080/shiro/shiro-cas"/> </bean> |

1. 配置cas注销的过滤器

|  |
| --- |
| <bean id="logout" class="org.apache.shiro.web.filter.authc.LogoutFilter">  <property name="redirectUrl" value="http://localhost:80/cas/logout?service=http://localhost:8080/shiro/index.jsp"/> </bean> |

1. 修改shiro过滤器

|  |
| --- |
| <bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">  <!-- 表示现在要配置的是一个安全管理器 -->  <property name="securityManager" ref="securityManager"/>  <!-- 出现错误之后的跳转路径的配置 -->  <property name="loginUrl" value="http://localhost:80/cas/login?service=http://localhost:8080/shiro/shiro-cas"/>  <!-- 配置shiro里面需要使用到的过滤器操作 -->  <property name="filters">  <map>  <entry key="logout" value-ref="logout"/>  <entry key="casFilter" value-ref="casFilter"/>  </map>  </property>  <!-- shiro里面需要针对于所有的路径进行配置，所有的配置需要通过文本的形式设置 -->  <property name="filterChainDefinitions">  <value>  /shiro-cas=casFilter  /lougout=logout  /\*=authc  </value>  </property> </bean> |

# 5、CAS配置RememberMe

如果要想配置RememberMe的操作需要考虑到两个配置部分：shiro（WEB端）、CAS。

1. 修改spring-shiro.xml文件，为其追加一个工厂类；

|  |
| --- |
| <bean id="casSubjectFactory" class="org.apache.shiro.cas.CasSubjectFactory"/> |
| <bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">  <!-- 配置需要使用的Realms -->  <property name="realm" ref="userCasRealm"/>  <property name="subjectFactory" ref="casSubjectFactory"/>  <property name="cacheManager" ref="cacheManager"/>  <!-- 定义要使用的session管理器 -->  <property name="sessionManager" ref="sessionManager"/>  <!-- 定义RememberMe的管理器 -->  <property name="rememberMeManager" ref="rememberMeManager"/> </bean> |

1. 修改cas服务器端的“deployerConfigContext.xml”配置文件；

|  |
| --- |
| <bean id="authenticationManager" class="org.jasig.cas.authentication.PolicyBasedAuthenticationManager">  <constructor-arg>  <map>  <entry key-ref="proxyAuthenticationHandler" value-ref="proxyPrincipalResolver" />  <entry key-ref="dbAuthHandler" value-ref="primaryPrincipalResolver"/>  </map>  </constructor-arg>  <property name="authenticationPolicy">  <bean class="org.jasig.cas.authentication.AnyAuthenticationPolicy" />  </property>  <property name="authenticationMetaDataPopulators">  <list>  <bean class="org.jasig.cas.authentication.principal.RememberMeAuthenticationMetaDataPopulator"/>  </list>  </property> </bean> |

1. 修改cas服务器端的“webflow\login\login-webflow.xml”文件；

|  |
| --- |
| <!—  修改前  <var name="credential" class="org.jasig.cas.authentication.UsernamePasswordCredential"/>  -->  <!—修改后--> <var name="credential" class="org.jasig.cas.authentication.RememberMeUsernamePasswordCredential"/> |
| <view-state id="viewLoginForm" view="casLoginView" model="credential">  <binder>  <binding property="username" required="true"/>  <binding property="password" required="true"/>  <binding property="rememberMe"/>  </binder>  <on-entry>  <set name="viewScope.commandName" value="'credential'"/>  </on-entry>  <transition on="submit" bind="true" validate="true" to="realSubmit"/> </view-state> |

1. 修改cas服务器端“spring-configuration\ticketExpirationPolicies.xml”文件；

删除如下内容：

|  |
| --- |
| <bean id="grantingTicketExpirationPolicy" class="org.jasig.cas.ticket.support.TicketGrantingTicketExpirationPolicy"  c:maxTimeToLive="${tgt.maxTimeToLiveInSeconds:28800}" c:timeToKill="${tgt.timeToKillInSeconds:7200}" c:timeUnit-ref="SECONDS" /> |

替换新的内容：

|  |
| --- |
| <bean id="grantingTicketExpirationPolicy" class="org.jasig.cas.ticket.support.RememberMeDelegatingExpirationPolicy">  <property name="sessionExpirationPolicy" ref="timeoutExpirationPolicy"/>  <property name="rememberMeExpirationPolicy" ref="timeoutExpirationPolicy"/> </bean> <bean id="timeoutExpirationPolicy" class="org.jasig.cas.ticket.support.TimeoutExpirationPolicy">  <constructor-arg index="0" value="100000"/> </bean> |

1. 修改cas服务器端“spring-configuration\ticketGrantingTicketCookieGenerator.xml”文件

|  |
| --- |
| <bean id="ticketGrantingTicketCookieGenerator" class="org.jasig.cas.web.support.CookieRetrievingCookieGenerator"  c:casCookieValueManager-ref="cookieValueManager"  p:cookieSecure="false"  p:cookieMaxAge="-1"  p:cookieName="TGC"  p:cookiePath=""  p:rememberMeMaxAge="100000"/> |

1. 修改cas服务器端的登录显示页面“view/jsp/default/ui/casLoginView.jsp”

|  |
| --- |
| <section class="row">  <input type="checkbox" name="rememberMe" id="rememberMe" value="true">  <label for="rememberMe">  <**spring:message** code="screen.rememberme.checkbox.title"/>  </label> </section> |

# 6、配置登录验证码

1. 引入google-kaptcha和filters的jar包；

|  |
| --- |
| <dependency>  <groupId>com.google.code</groupId>  <artifactId>kaptcha</artifactId>  <version>2.3</version> </dependency> <dependency>  <groupId>com.jhlabs</groupId>  <artifactId>filters</artifactId>  <version>2.0.235</version> </dependency> |

1. 在web.xml文件配置KaptchaServlet；

|  |
| --- |
| <servlet>  <servlet-name>KaptchaServlet</servlet-name>  <servlet-class>com.google.code.kaptcha.servlet.KaptchaServlet</servlet-class>  <init-param>  <!-- 是否显示边框 -->  <param-name>kaptcha.border</param-name>  <param-value>no</param-value>  </init-param>  <init-param>  <!-- 边框颜色 -->  <param-name>kaptcha.border.color</param-name>  <param-value>105,179,90</param-value>  </init-param>  <init-param>  <!-- 字体颜色 -->  <param-name>kaptcha.textproducer.font.color</param-name>  <param-value>red</param-value>  </init-param>  <init-param>  <!-- 图片宽度 -->  <param-name>kaptcha.image.width</param-name>  <param-value>100</param-value>  </init-param>  <init-param>  <!-- 图片高度 -->  <param-name>kaptcha.image.height</param-name>  <param-value>40</param-value>  </init-param>  <init-param>  <!-- 文字尺寸 -->  <param-name>kaptcha.textproducer.font.size</param-name>  <param-value>35</param-value>  </init-param>  <init-param>  <!-- 验证码保存在session的属性名 -->  <param-name>kaptcha.session.key</param-name>  <param-value>rand</param-value>  </init-param>  <init-param>  <!-- 字符间距 -->  <param-name>kaptcha.textproducer.char.space</param-name>  <param-value>2</param-value>  </init-param>  <init-param>  <!-- 验证码长度 -->  <param-name>kaptcha.textproducer.char.length</param-name>  <param-value>5</param-value>  </init-param>  <init-param>  <!-- 字体类型 -->  <param-name>kaptcha.textproducer.font.names</param-name>  <param-value>宋体,楷体,微软雅黑</param-value>  </init-param> </servlet> <servlet-mapping>  <servlet-name>KaptchaServlet</servlet-name>  <url-pattern>/captcha.jpg</url-pattern> </servlet-mapping> |

1. 创建“org.jasig.cas.authentication.RememberMeUsernamePasswordCredential”的子类，该子类主要定义了验证码属性；

|  |
| --- |
| package com.kuhnwei.mohist.examples.cas.authentication; import org.jasig.cas.authentication.RememberMeUsernamePasswordCredential; import javax.validation.constraints.NotNull; import javax.validation.constraints.Size;  */\*\*  \* 自定义的用户名密码凭证类，继承至org.jasig.cas.authentication.RememberMeUsernamePasswordCredential  \* 添加验证码的处理字段  \** ***@author*** *Kuhn Wei, email@kuhnwei.com  \** ***@version*** *2018/3/31 15:47  \*/* public class CaptchaRememberMeUsernamePasswordCredential extends RememberMeUsernamePasswordCredential {  */\*\*  \* 验证码  \*/* @NotNull  @Size(min = 5, message = "captcha.required")  private String captcha;   public String getCaptcha() {  return captcha;  }   public void setCaptcha(String captcha) {  this.captcha = captcha;  }   @Override  public boolean equals(Object o) {  if (this == o) return true;  if (o == null || getClass() != o.getClass()) return false;  if (!super.equals(o)) return false;   CaptchaRememberMeUsernamePasswordCredential that = (CaptchaRememberMeUsernamePasswordCredential) o;   return captcha != null ? captcha.equals(that.captcha) : that.captcha == null;  }   @Override  public int hashCode() {  int result = super.hashCode();  result = 31 \* result + (captcha != null ? captcha.hashCode() : 0);  return result;  } } |

1. 新建验证码相关的异常处理类，继承至org.jasig.cas.authentication.RootCasException；

验证码为空的异常

|  |
| --- |
| package com.kuhnwei.mohist.examples.cas.authentication; import org.jasig.cas.authentication.RootCasException; */\*\*  \* 用户输入的验证码为空时 的异常处理类  \** ***@author*** *Kuhn Wei, email@kuhnwei.com  \** ***@version*** *2018/3/31 16:08  \*/* public class NullCaptchaException extends RootCasException {  public static final String *CODE* = "captcha.required";  public NullCaptchaException() {  super(*CODE*);  } } |

验证码错误的异常

|  |
| --- |
| package com.kuhnwei.mohist.examples.cas.authentication; import org.jasig.cas.authentication.RootCasException; */\*\*  \* 用户输入的验证码错误 异常处理类  \** ***@author*** *Kuhn Wei, email@kuhnwei.com  \** ***@version*** *2018/3/31 16:08  \*/* public class BadCaptchaException extends RootCasException {  public static final String *CODE* = "error.authentication.captcha.bad";  public BadCaptchaException() {  super(*CODE*);  } } |

1. 新定义用于验证码检测时的Action类，继承至：org.jasig.cas.web.flow.AuthenticationViaFormAction

|  |
| --- |
| package com.kuhnwei.mohist.examples.cas.web.flow; import com.kuhnwei.mohist.examples.cas.authentication.BadCaptchaException; import com.kuhnwei.mohist.examples.cas.authentication.CaptchaRememberMeUsernamePasswordCredential; import com.kuhnwei.mohist.examples.cas.authentication.NullCaptchaException; import org.jasig.cas.authentication.Credential; import org.jasig.cas.authentication.RootCasException; import org.jasig.cas.web.flow.AuthenticationViaFormAction; import org.jasig.cas.web.support.WebUtils; import org.springframework.binding.message.MessageBuilder; import org.springframework.binding.message.MessageContext; import org.springframework.util.StringUtils; import org.springframework.webflow.execution.RequestContext; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpSession;  */\*\*  \* 用于验证码检测的Action类  \** ***@author*** *Kuhn Wei, email@kuhnwei.com  \** ***@version*** *2018/3/31 15:53  \*/* public class CaptchaAuthenticationViaFormAction extends AuthenticationViaFormAction {  public final String validatorCaptcha(final RequestContext context, final Credential credential, final MessageContext messageContext) throws Exception {  // 获取生成的验证码数据  HttpServletRequest request = WebUtils.*getHttpServletRequest*(context);  HttpSession session = request.getSession();  String rand = (String) session.getAttribute("rand");  // 取得生成的验证码后，验证码失效处理  session.removeAttribute("rand");  CaptchaRememberMeUsernamePasswordCredential crupc = (CaptchaRememberMeUsernamePasswordCredential) credential;  // 取得用户输入的验证码数据  String captcha = crupc.getCaptcha();  // 用户输入的验证码和生成的验证码进行校验  if (StringUtils.*isEmpty*(captcha) || StringUtils.*isEmpty*(rand)) {  this.errorDisplay(new NullCaptchaException(), messageContext);  return "error";  } else if (captcha.equalsIgnoreCase(rand)) {  return "success";  }  this.errorDisplay(new BadCaptchaException(), messageContext);  return "error";  }   */\*\*  \* 专门负责异常的信息提示  \** ***@param*** *e 异常对象  \** ***@param*** *messageContext 异常信息  \*/* private void errorDisplay(final RootCasException e, final MessageContext messageContext) {  messageContext.addMessage(new MessageBuilder().error().code(e.getCode()).defaultText(e.getCode()).build());  } } |

1. 修改“cas-servlet.xml”文件，主要是修改验证操作所使用的类；

com.kuhnwei.mohist.examples.cas.web.flow.CaptchaAuthenticationViaFormAction类就是用于此的；

原始内容

|  |
| --- |
| <bean id="authenticationViaFormAction" class="org.jasig.cas.web.flow.AuthenticationViaFormAction"  p:centralAuthenticationService-ref="centralAuthenticationService"  p:warnCookieGenerator-ref="warnCookieGenerator"/> |

更新后的内容

|  |
| --- |
| <bean id="authenticationViaFormAction"  class="com.kuhnwei.mohist.examples.cas.web.flow.CaptchaAuthenticationViaFormAction"  p:centralAuthenticationService-ref="centralAuthenticationService"  p:warnCookieGenerator-ref="warnCookieGenerator"/> |

1. 修改“webflow/login/login-webflow.xml”文件；

原始内容

|  |
| --- |
| <var name="credential" class="org.jasig.cas.authentication.RememberMeUsernamePasswordCredential"/> |

更新后的内容（第3步创建的Java类）

|  |
| --- |
| <var name="credential" class="com.kuhnwei.mohist.examples.cas.authentication.CaptchaRememberMeUsernamePasswordCredential"/> |

为项目增加验证码的输入验证处理

|  |
| --- |
| <action-state id="captchaSubmit">  <evaluate expression="authenticationViaFormAction.validatorCaptcha(flowRequestContext, flowScope.credential, messageContext)"/>  <transition on="error" to="generateLoginTicket"/>  <transition on="success" to="realSubmit"/> </action-state> |
| <view-state id="viewLoginForm" view="casLoginView" model="credential">  <binder>  <binding property="username" required="true"/>  <binding property="password" required="true"/>  <binding property="captcha" required="true"/>  <binding property="rememberMe"/>  </binder>  <on-entry>  <set name="viewScope.commandName" value="'credential'"/>  </on-entry>  <transition on="submit" bind="true" validate="true" to="captchaSubmit"/> </view-state> |

1. 修改资源文件“messages\*.properties”，添加验证码相关的信息，用于国际化处理

|  |
| --- |
| screen.welcome.label.captcha=验证码 screen.welcome.label.captcha.accesskey=c captcha.required=请输入验证码！ error.authentication.captcha.bad=验证码输入错误！ |

1. 修改登录页面“view/jsp/default/ui/casLoginView.jsp”

|  |
| --- |
| <section class="row">  <label for="captcha">  <**spring:message** code="screen.welcome.label.captcha"/>  </label>  <**spring:message** code="screen.welcome.label.captcha.accesskey" var="captchaAccesskey"/>  <**form:input** path="captcha" cssClass="required" cssErrorClass="error" id="captcha" size="5" tabindex="3" accesskey="**${**captchaAccesskey**}**" autocomplete="off" htmlEscape="true"/>  <img src="captcha.jpg" onclick="**this**.src='captcha.jpg?' + Math.random()"> </section> |

