

Pig整合MaxKey集成指南

一、集成功能

二、pig版本

1、pig后端版本：3.6

2、pig前端版本：最新代码

三、集成指导

1.pig-auth模块

pom.xml的更改覆盖了pig的Oauth2的token生成方案

PigTokenEndpoint中新增获取token的方法

TokenMananer工具类

2.pig-common

pom文件的更改

annotation包下面更改EnablePigResourceServer

Config包下

新增CasProperties主要是CAS得配置信息配置在NACOS中

SecurityConfig类主要是CAS的认证流程并且覆盖原本pig的认证流程

service包

utils包

vue包

api包

userlogin改造

router改造

store的user改造

Nacos配置文件新增

一、集成功能

1、pig集成maxkey中CAS的单点登录

2、pig集成maxke的组织架构信息等

二、pig版本

1、pig后端版本：3.6

gitee地址：<https://gitee.com/log4j/pig.git>

2、pig前端版本：最新代码

gitee地址：<https://gitee.com/log4j/pig-ui.git>

三、集成指导

1.pig-auth模块

pom.xml的更改覆盖了pig的Oauth2的token生成方案

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>com.pig4cloud</groupId>
7     <artifactId>pig</artifactId>
8     <version>3.6.7</version>
9   </parent>
10
11   <artifactId>pig-auth</artifactId>
12   <packaging>jar</packaging>
13
14   <description>pig 认证授权中心, 基于 spring security oAuth2</description>
15
16   <dependencies>
17     <dependency>
18       <groupId>io.jsonwebtoken</groupId>
19       <artifactId>jjwt</artifactId>
20       <version>0.7.0</version>
21     </dependency>
22     <!--注册中心客户端-->
23     <dependency>
24       <groupId>com.alibaba.cloud</groupId>
25       <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
26     </dependency>
27     <!--配置中心客户端-->
28     <dependency>
29       <groupId>com.alibaba.cloud</groupId>
30       <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
31     </dependency>
32     <!--断路器依赖-->
33     <dependency>
34       <groupId>com.pig4cloud</groupId>
35       <artifactId>pig-common-feign</artifactId>
36     </dependency>
37     <!--upms api、model 模块-->
38     <dependency>
39       <groupId>com.pig4cloud</groupId>
40       <artifactId>pig-upms-api</artifactId>
41     </dependency>
42     <dependency>
```

```

43     <groupId>com.pig4cloud</groupId>
44     <artifactId>pig-common-security</artifactId>
45 </dependency>
46 <dependency>
47     <groupId>org.springframework.boot</groupId>
48     <artifactId>spring-boot-starter-security</artifactId>
49 </dependency>
50 <!--freemarker-->
51 <dependency>
52     <groupId>org.springframework.boot</groupId>
53     <artifactId>spring-boot-starter-freemarker</artifactId>
54 </dependency>
55 <!--undertow容器-->
56 <dependency>
57     <groupId>org.springframework.boot</groupId>
58     <artifactId>spring-boot-starter-undertow</artifactId>
59 </dependency>
60 <!-- log -->
61 <dependency>
62     <groupId>com.pig4cloud</groupId>
63     <artifactId>pig-common-log</artifactId>
64 </dependency>
65 </dependencies>
66
67 <build>
68 <plugins>
69 <plugin>
70     <groupId>org.springframework.boot</groupId>
71     <artifactId>spring-boot-maven-plugin</artifactId>
72 </plugin>
73 <plugin>
74     <groupId>io.fabric8</groupId>
75     <artifactId>docker-maven-plugin</artifactId>
76 </plugin>
77 </plugins>
78 </build>
79
80 </project>

```

PigTokenEndpoint中新增获取token的方法

```
1 package com.pig4cloud.pig.auth.endpoint;
2
3 import cn.hutool.core.date.DatePattern;
4 import cn.hutool.core.date.TemporalAccessorUtil;
5 import cn.hutool.core.map.MapUtil;
6 import cn.hutool.core.util.StrUtil;
7 import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
8 import com.pig4cloud.pig.admin.api.entity.SysOauthClientDetails;
9 import com.pig4cloud.pig.admin.api.feign.RemoteClientDetailsService;
10 import com.pig4cloud.pig.admin.api.vo.TokenVo;
11 import com.pig4cloud.pig.auth.support.handler.PigAuthenticationFailureEventHandler;
12 import com.pig4cloud.pig.auth.utils.RedisUtils;
13 import com.pig4cloud.pig.auth.utils.TokenManager;
14 import com.pig4cloud.pig.common.core.constant.CacheConstants;
15 import com.pig4cloud.pig.common.core.constant.CommonConstants;
16 import com.pig4cloud.pig.common.core.util.R;
17 import com.pig4cloud.pig.common.core.util.RetOps;
18 import com.pig4cloud.pig.common.core.util.SpringContextHolder;
19 import com.pig4cloud.pig.common.security.annotation.Inner;
20 import com.pig4cloud.pig.common.security.service.PigUser;
21 import com.pig4cloud.pig.common.security.util.OAuth2EndpointUtils;
22 import com.pig4cloud.pig.common.security.util.OAuth2ErrorCodesExpand;
23 import com.pig4cloud.pig.common.security.util.OAuthClientException;
24 import lombok.RequiredArgsConstructor;
25 import lombok.SneakyThrows;
26 import lombok.extern.slf4j.Slf4j;
27 import org.springframework.cache.CacheManager;
28 import org.springframework.data.redis.core.RedisTemplate;
29 import org.springframework.http.HttpHeaders;
30 import org.springframework.http.HttpStatus;
31 import org.springframework.http.MediaType;
32 import org.springframework.http.converter.HttpMessageConverter;
33 import org.springframework.http.server.ServletServerHttpResponse;
34 import org.springframework.security.authentication.event.LogoutSuccessEvent;
35 import org.springframework.security.core.Authentication;
36 import org.springframework.security.core.context.SecurityContextHolder;
37 import org.springframework.security.oauth2.core.OAuth2AccessToken;
38 import org.springframework.security.oauth2.core.endpoint.OAuth2AccessTokenResponse;
39 import org.springframework.security.oauth2.core.endpoint.OAuth2ParameterNames;
40 import org.springframework.security.oauth2.core.http.converter.OAuth2AccessTokenResponseHttpMessageConverter;
```

```

41 import org.springframework.security.oauth2.server.authorization.OAuth2Aut
42 horization;
43 import org.springframework.security.oauth2.server.authorization.OAuth2Aut
44 horizationService;
45 import org.springframework.security.oauth2.server.authorization.OAuth2Tok
46 enType;
47 import org.springframework.security.oauth2.server.resource.InvalidBearerT
48 okenException;
49 import org.springframework.security.web.authentication.AuthenticationFail
50 ureHandler;
51 import org.springframework.security.web.authentication.preauth.PreAuthent
52 icatedAuthenticationToken;
53 import org.springframework.util.StringUtils;
54 import org.springframework.web.bind.annotation.*;
55 import org.springframework.web.servlet.ModelAndView;
56
57 import javax.annotation.Resource;
58 import javax.servlet.http.HttpServletRequest;
59 import javax.servlet.http.HttpServletResponse;
60 import java.security.Principal;
61 import java.util.HashMap;
62 import java.util.List;
63 import java.util.Map;
64 import java.util.Set;
65 import java.util.stream.Collectors;
66
67 /**
68  * @author lengleng
69  * @date 2019/2/1 删除token端点
70  */
71 @Slf4j
72 @RestController
73 @RequiredArgsConstructor
74 @RequestMapping("/token")
75 public class PigTokenEndpoint {
76
77     private final HttpMessageConverter<OAuth2AccessTokenResponse> accessToken
78     HttpResponseConverter = new OAuth2AccessTokenResponseHttpMessageConverter
79     ();
80
81     private final AuthenticationFailureHandler authenticationFailureHandler =
82     new PigAuthenticationFailureEventHandler();
83
84     private final OAuth2AuthorizationService authorizationService;
85
86     private final RemoteClientDetailsService clientDetailsService;
87
88     private final RedisTemplate<String, Object> redisTemplate;

```

```

80
81 private final CacheManager cacheManager;
82
83 @Resource
84 private RedisUtils redisUtils;
85
86
87 @Resource
88 private TokenManager tokenManager;
89
90 private final static String SPRING_SESSION_PREFIX = "spring:session:sessi
91 ons:%s";
92 private final static String PIG_TOKEN_PREFIX = "pig:token:%s:%s";
93 private final static String ASSCEE_TOKEN = "access_token";
94 private final static String REFRESH_TOKEN = "refresh_token";
95
96 private long tokenExpiration = 24 * 60 * 60 * 1000;
97
98 /**
99  * 认证页面
100  *
101  * @param modelAndView
102  * @param error 表单登录失败处理回调的错误信息
103  * @return ModelAndView
104  */
105 @GetMapping("/login")
106 public ModelAndView require(ModelAndView modelAndView, @RequestParam(requ
107 ired = false) String error) {
108 modelAndView.setViewName("ftl/login");
109 modelAndView.addObject("error", error);
110 return modelAndView;
111 }
112
113 @GetMapping("/confirm_access")
114 public ModelAndView confirm(Principal principal, ModelAndView modelAndView,
115 @RequestParam(OAuth2ParameterNames.CLIENT_ID) String clientId,
116 @RequestParam(OAuth2ParameterNames.SCOPE) String scope,
117 @RequestParam(OAuth2ParameterNames.STATE) String state) {
118 SysOAuthClientDetails clientDetails = RetOps.of(clientDetailsService.getClient
119 DetailsById(clientId))
120 .getData()
121 .orElseThrow(() -> new OAuthClientException("clientId 不合法"));
122
123 Set<String> authorizedScopes = StringUtils.commaDelimitedListToSet(client
124 Details.getScope());
125 modelAndView.addObject("clientId", clientId);

```

```

123     modelAndView.addObject("state", state);
124     modelAndView.addObject("scopeList", authorizedScopes);
125     modelAndView.addObject("principalName", principal.getName());
126     modelAndView.setViewName("ftl/confirm");
127     return modelAndView;
128 }
129
130
131 /**
132  * 退出并删除token
133  *
134  * @param authHeader Authorization
135  */
136 @DeleteMapping("/logout")
137 public R<Boolean> logout(HttpServletRequest request, @RequestHeader(value
138     = HttpHeaders.AUTHORIZATION, required = false) String authHeader) {
139     if (StrUtil.isBlank(authHeader)) {
140         return R.ok();
141     }
142     String sessionId = request.getSession().getId();
143     if (StrUtil.isBlank(sessionId)) {
144         return R.ok();
145     }
146     boolean isSuccess = redisUtils.deleteKey(generateSessionId(sessionId));
147     if (isSuccess) {
148         return R.ok();
149     } else {
150         return R.failed();
151     }
152 }
153
154 /**
155  * 校验token
156  *
157  * @param token 令牌
158  */
159 @SneakyThrows
160 @GetMapping("/check_token")
161 public void checkToken(String token, HttpServletResponse response, HttpSe
162     rvletRequest request) {
163
164     ServletServerHttpResponse httpResponse = new ServletServerHttpResponse(re
165         sponse);
166     if (StrUtil.isBlank(token)) {
167         httpResponse.setStatus(HttpStatus.UNAUTHORIZED);
168         this.authenticationFailureHandler.onAuthenticationFailure(request, respon
169             se,

```



```
167     new InvalidBearerTokenException(OAuth2ErrorCodesExpand.TOKEN_MISSING));
168     return;
169 }
170 OAuth2Authorization authorization = authorizationService.findByToken(token,
171     OAuth2TokenType.ACCESS_TOKEN);
172 // 如果令牌不存在 返回401
173
```

TokenMananer工具类

```
1 package com.pig4cloud.pig.auth.utils;
2
3 import io.jsonwebtoken.CompressionCodecs;
4 import io.jsonwebtoken.Jwts;
5 import io.jsonwebtoken.SignatureAlgorithm;
6 import org.springframework.stereotype.Component;
7
8 import java.util.Date;
9
10 @Component
11 public class TokenManager {
12     private long tokenExpiration = 24 * 60 * 60 * 1000;
13     private final static String TOKEN_SIGN_KEY = "MAKKEY_PIG";
14
15     public String createToken(String subject, String username, String id)
16     {
17         String token = Jwts.builder()
18             .setSubject(subject)
19             .claim("nickname", username)
20             .claim("id", id)
21             .setExpiration(new Date(System.currentTimeMillis() + token
22 Expiration))
23             .signWith(SignatureAlgorithm.HS512, TOKEN_SIGN_KEY)
24             .compressWith(CompressionCodecs.GZIP).compact();
25         return token;
26     }
27
28     public String getUserFromToken(String token) {
29         String user = Jwts.parser().setSigningKey(TOKEN_SIGN_KEY).parseClaimsJws(token).getBody().getSubject();
30         return user;
31     }
32
33     public void removeToken(String token) {
34         //jwttoken无需删除，客户端扔掉即可。
35     }
36 }
```

2.pig-common

pom文件的更改

主要新增 CAS的MAVEN包

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <groupId>com.pig4cloud</groupId>
7         <artifactId>pig-common</artifactId>
8         <version>3.6.7</version>
9     </parent>
10
11     <artifactId>pig-common-security</artifactId>
12     <packaging>jar</packaging>
13
14     <description>pig 安全工具类</description>
15
16
17     <dependencies>
18         <dependency>
19             <groupId>org.springframework.session</groupId>
20             <artifactId>spring-session-data-redis</artifactId>
21         </dependency>
22         <!-- spring security cas -->
23         <dependency>
24             <groupId>org.springframework.security</groupId>
25             <artifactId>spring-security-cas</artifactId>
26         </dependency>
27         <!-- 工具类核心包 -->
28         <dependency>
29             <groupId>com.pig4cloud</groupId>
30             <artifactId>pig-common-core</artifactId>
31         </dependency>
32         <dependency>
33             <groupId>cn.hutool</groupId>
34             <artifactId>hutool-extra</artifactId>
35         </dependency>
36         <!-- UPMS API -->
37         <dependency>
38             <groupId>com.pig4cloud</groupId>
39             <artifactId>pig-upms-api</artifactId>
40         </dependency>
41         <!-- common utils -->
42         <dependency>
43             <groupId>org.springframework.cloud</groupId>
```

```

44         <artifactId>spring-cloud-commons</artifactId>
45     </dependency>
46     <!--feign 工具类-->
47     <dependency>
48         <groupId>org.springframework.cloud</groupId>
49         <artifactId>spring-cloud-starter-openfeign</artifactId>
50     </dependency>
51     <dependency>
52         <groupId>org.springframework.security</groupId>
53         <artifactId>spring-security-oauth2-jose</artifactId>
54     </dependency>
55     <dependency>
56         <groupId>org.springframework.security</groupId>
57         <artifactId>spring-security-oauth2-authorization-server</artif
58 actId>
59         <version>${spring.authorization.version}</version>
60     </dependency>
61     <dependency>
62         <groupId>org.springframework</groupId>
63         <artifactId>spring-webmvc</artifactId>
64     </dependency>
65 </dependencies>
</project>

```

annotation包下面更改EnablePigResourceServer

```
1 package com.pig4cloud.pig.common.security.annotation;
2
3 import com.pig4cloud.pig.common.security.component.PigResourceServerAutoCo
nfiguration;
4 import com.pig4cloud.pig.common.security.component.PigResourceServerConfig
uration;
5 import com.pig4cloud.pig.common.security.component.ResourceAuthExceptionEn
tryPoint;
6 import com.pig4cloud.pig.common.security.config.*;
7 import org.springframework.context.annotation.Import;
8
9 import java.lang.annotation.*;
10
11 /**
12  * @author lengleng
13  * @date 2022-06-04
14  * <p>
15  * 资源服务注解
16  */
17 @Documented
18 @Inherited
19 @Target({ElementType.TYPE})
20 @Retention(RetentionPolicy.RUNTIME)
21 //@Import({ PigResourceServerAutoConfiguration.class, PigResourceServerCon
figuration.class })
22 @Import({PigResourceServerAutoConfiguration.class, CasProperties.class, Se
curityConfig.class})
23 public @interface EnablePigResourceServer {
24
25 }
26
```

Config包下

新增CasProperties主要是CAS得配置信息配置在NACOS中

```
1 package com.pig4cloud.pig.common.security.config;
2
3 import lombok.Data;
4 import org.springframework.beans.factory.annotation.Value;
5 import org.springframework.stereotype.Component;
6
7 @Data
8 @Component
9 public class CasProperties {
10
11     /**
12      * 密钥
13      */
14     @Value("${cas.key}")
15     private String casKey;
16
17     /**
18      * cas服务端地址
19      */
20     @Value("${cas.server.host.url}")
21     private String casServerUrl;
22
23     /**
24      * cas服务端地址
25      */
26     @Value("${cas.server.host.grant_url}")
27     private String casGrantingUrl;
28
29     /**
30      * cas服务端登录地址
31      */
32     @Value("${cas.server.host.login_url}")
33     private String casServerLoginUrl;
34
35     /**
36      * cas服务端登出地址 并回跳到制定页面
37      */
38     @Value("${cas.server.host.logout_url}")
39     private String casServerLogoutUrl;
40
41     /**
42      * cas客户端地址
43      */
44     @Value("${cas.service.host.url}")
45     private String casServiceUrl;
```

```
46
47 ▼
48     /**
49     * cas客户端地址登录地址
50     */
51     @Value("${cas.service.host.login_url}")
52     private String casServiceLoginUrl;
53 ▼
54     /**
55     * cas客户端地址登出地址
56     */
57     @Value("${cas.service.host.logout_url}")
58     private String casServiceLogoutUrl;
59 }
```

SecurityConfig类主要是CAS的认证流程并且覆盖原本pig的认证流程


```
1 package com.pig4cloud.pig.common.security.config;
2
3 import cn.hutool.core.util.ArrayUtil;
4 import com.pig4cloud.pig.common.security.component.PermitAllUrlProperties
5 ;
6 import com.pig4cloud.pig.common.security.component.PigBearerTokenExtracto
7 r;
8 import com.pig4cloud.pig.common.security.component.ResourceAuthExceptioE
9 ntryPoint;
10 import lombok.RequiredArgsConstructor;
11 import lombok.extern.slf4j.Slf4j;
12 import org.jasig.cas.client.session.SingleSignOutFilter;
13 import org.jasig.cas.client.validation.Cas20ServiceTicketValidator;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.context.annotation.Bean;
16 import org.springframework.context.annotation.Configuration;
17 import org.springframework.data.redis.core.RedisTemplate;
18 import org.springframework.security.cas.ServiceProperties;
19 import org.springframework.security.cas.authentication.CasAuthenticationP
20 rovider;
21 import org.springframework.security.cas.web.CasAuthenticationEntryPoint;
22 import org.springframework.security.cas.web.CasAuthenticationFilter;
23 import org.springframework.security.config.annotation.authentication.buil
24 ders.AuthenticationManagerBuilder;
25 import org.springframework.security.config.annotation.method.configuratio
26 n.EnableGlobalMethodSecurity;
27 import org.springframework.security.config.annotation.web.builders.HttpSe
28 curity;
29 import org.springframework.security.config.annotation.web.configuration.E
30 nableWebSecurity;
31 import org.springframework.security.config.annotation.web.configuration.W
32 ebSecurityConfigurerAdapter;
33 import org.springframework.security.core.userdetails.AuthenticationUserDe
34 tailsService;
35 import org.springframework.security.web.authentication.logout.LogoutFilte
36 r;
37 import org.springframework.security.web.authentication.logout.SecurityCon
38 textLogoutHandler;
39
40 @Slf4j
41 @Configuration
42 @EnableWebSecurity // 启用web权限
43 @EnableGlobalMethodSecurity(prePostEnabled = true) // 启用方法验证
44 @RequiredArgsConstructor
45 public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

```

34
35     @Autowired
36     private CasProperties casProperties;
37
38     @Autowired
39     private AuthenticationUserDetailsService casUserDetailsService;
40
41     private final PermitAllUrlProperties permitAllUrl;
42
43
44     /**
45      * 定义认证用户信息获取来源，密码校验规则等
46      */
47     @Override
48     protected void configure(AuthenticationManagerBuilder auth) throws Exception {
49         super.configure(auth);
50         auth.authenticationProvider(casAuthenticationProvider());
51     }
52
53     /**
54      * 定义安全策略
55      */
56     @Override
57     protected void configure(HttpSecurity http) throws Exception {
58
59         http.authorizeRequests()// 配置安全策略
60             .antMatchers(ArrayUtil.toArray(permitAllUrl.getUrls(), String.class)).permitAll()
61             .anyRequest().authenticated()// 其余的所有请求都需要验证
62             .and().logout().permitAll()// 定义logout不需要验证
63             .and().formLogin();// 使用form表单登录
64
65         http.exceptionHandling()
66             .authenticationEntryPoint(casAuthenticationEntryPoint())
67             .and()
68             .addFilter(casAuthenticationFilter())
69             .addFilterBefore(casLogoutFilter(), LogoutFilter.class)
70             .addFilterBefore(singleSignOutFilter(), CasAuthentication
71 Filter.class);
72         // 取消跨站请求伪造防护
73         http.csrf().disable();
74         // 防止iframe 造成跨域
75         http.headers().frameOptions().disable();
76         // http.csrf().disable(); //禁用CSRF
77     }
78     /**

```

```

79     * 认证的入口
80     */
81     @Bean
82     public CasAuthenticationEntryPoint casAuthenticationEntryPoint() {
83         CasAuthenticationEntryPoint casAuthenticationEntryPoint = new Cas
84 AuthenticationEntryPoint();
85         casAuthenticationEntryPoint.setLoginUrl(casProperties.getCasServe
86 rLoginUrl());
87         casAuthenticationEntryPoint.setServiceProperties(servicePropertie
88 s());
89         return casAuthenticationEntryPoint;
90     }
91     /**
92     * 指定service相关信息
93     */
94     @Bean
95     public ServiceProperties serviceProperties() {
96         ServiceProperties serviceProperties = new ServiceProperties();
97         //设置cas客户端登录完整的url
98         serviceProperties.setService(casProperties.getCasServiceUrl() + c
99 asProperties.getCasServiceLoginUrl());
100        serviceProperties.setSendRenew(false);
101        serviceProperties.setAuthenticateAllArtifacts(true);
102        return serviceProperties;
103    }
104    /**
105    * CAS认证过滤器
106    */
107    @Bean
108    public CasAuthenticationFilter casAuthenticationFilter() throws Excep
109 tion {
110        CasAuthenticationFilter casAuthenticationFilter = new CasAuthenti
111 cationFilter();
112        casAuthenticationFilter.setAuthenticationManager(authenticationMa
113 nager());
114        casAuthenticationFilter.setFilterProcessesUrl(casProperties.getCa
115 sServiceUrl() + casProperties.getCasServiceLoginUrl());
116        casAuthenticationFilter.setServiceProperties(serviceProperties())
117 ;
118        return casAuthenticationFilter;
119    }
120    /**
121    * cas 认证 Provider
122    */
123    @Bean

```

```

118     public CasAuthenticationProvider casAuthenticationProvider() {
119         CasAuthenticationProvider casAuthenticationProvider = new CasAuth
120 enticationProvider();
121         casAuthenticationProvider.setAuthenticationUserDetailsService(cas
122 UserDetailService);
123         // //这里只是接口类型，实现的接口不一样，都可以的。
124         casAuthenticationProvider.setServiceProperties(serviceProperties(
125 ));
126         casAuthenticationProvider.setTicketValidator(cas20ServiceTicketVa
127 lidator());
128         casAuthenticationProvider.setKey("casAuthenticationProviderKey");
129         return casAuthenticationProvider;
130     }
131
132     @Bean
133     public Cas20ServiceTicketValidator cas20ServiceTicketValidator() {
134         return new Cas20ServiceTicketValidator(casProperties.getCasGranti
135 ngUrl());
136     }
137
138     /**
139      * 单点登出过滤器
140      */
141     @Bean
142     public SingleSignOutFilter singleSignOutFilter() {
143         SingleSignOutFilter singleSignOutFilter = new SingleSignOutFilter
144 ();
145         singleSignOutFilter.setLogoutCallbackPath(casProperties.getCasSer
146 verUrl());
147         singleSignOutFilter.setIgnoreInitConfiguration(true);
148         return singleSignOutFilter;
149     }
150
151     /**
152      * 请求单点退出过滤器
153      */
154     @Bean
155     public LogoutFilter casLogoutFilter() {
156         LogoutFilter logoutFilter = new LogoutFilter(casProperties.getCas
157 ServerLogoutUrl(), new SecurityContextLogoutHandler());
158         logoutFilter.setFilterProcessesUrl(casProperties.getCasServiceLog
159 outUrl());
160         return logoutFilter;
161     }
162 }

```

service包

新增PigUserDetailsServiceImpl类主要功能为CAS服务认证成功方法，判断用户是否存在，存在获取用户信息，不存在调用远程接口新增用户并同步组织架构信息

```
1  /*
2   * Copyright (c) 2020 pig4cloud Authors. All Rights Reserved.
3   *
4   * Licensed under the Apache License, Version 2.0 (the "License");
5   * you may not use this file except in compliance with the License.
6   * You may obtain a copy of the License at
7   *
8   *     http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to in writing, software
11  * distributed under the License is distributed on an "AS IS" BASIS,
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13  * See the License for the specific language governing permissions and
14  * limitations under the License.
15  */
16
17  package com.pig4cloud.pig.common.security.service;
18
19  import com.pig4cloud.pig.admin.api.dto.UserInfo;
20  import com.pig4cloud.pig.admin.api.feign.RemoteUserService;
21  import com.pig4cloud.pig.common.core.constant.CacheConstants;
22  import com.pig4cloud.pig.common.core.util.R;
23  import lombok.RequiredArgsConstructor;
24  import lombok.SneakyThrows;
25  import lombok.extern.slf4j.Slf4j;
26  import org.springframework.cache.Cache;
27  import org.springframework.cache.CacheManager;
28  import org.springframework.context.annotation.Primary;
29  import org.springframework.security.cas.authentication.CasAssertionAuthenticationToken;
30  import org.springframework.security.core.userdetails.AuthenticationUserDetailsService;
31  import org.springframework.security.core.userdetails.UserDetails;
32  import org.springframework.security.core.userdetails.UsernameNotFoundException;
33
34  import java.util.*;
35
36  /**
37   * 用户详细信息
38   *
39   * @author lengleng hccake
40   */
41  @Slf4j
```

```

42 @Primary
43 @RequiredArgsConstructor
44 public class PigUserDetailsServiceImpl implements PigUserDetailsService, A
45   uthenticationUserDetailsService<CasAssertionAuthenticationToken> {
46
47     private final RemoteUserService remoteUserService;
48
49     private final CacheManager cacheManager;
50
51     /**
52      * 用户名密码登录
53      *
54      * @param username 用户名
55      * @return
56      */
57     @Override
58     @SneakyThrows
59     public UserDetails loadUserByUsername(String username) {
60         Cache cache = cacheManager.getCache(CacheConstants.USER_DETAILS);
61         if (cache != null && cache.get(username) != null) {
62             return (PigUser) cache.get(username).get();
63         }
64         return getUserDetails(remoteUserService.info(username), username);
65     }
66
67     @Override
68     public int getOrder() {
69         return Integer.MIN_VALUE;
70     }
71
72     @Override
73     public UserDetails loadUserDetails(CasAssertionAuthenticationToken tok
74 en) throws UsernameNotFoundException {
75         log.info("getCredentials:{}", token.getCredentials());
76         String username = token.getName();
77         Cache cache = cacheManager.getCache(CacheConstants.USER_DETAILS);
78         if (cache != null && cache.get(username) != null) {
79             return (PigUser) cache.get(username).get();
80         }
81         R<UserInfo> result = remoteUserService.saveIfNotExist(token.getAss
82 ertion().getPrincipal().getAttributes());
83         return getUserDetails(result, username);
84     }
85
86     private UserDetails getUserDetails(R<UserInfo> result, String username
87 ) {
88         Cache cache = cacheManager.getCache(CacheConstants.USER_DETAILS);
89         UserDetails userDetails = getUserDetails(result);

```

```

86         if (cache != null) {
87             cache.put(username, userDetails);
88         }
89         return userDetails;
90     }
91 }
92

```

utils包

新增BeanCreator方法，主要创建SysUser对象工具类

```

Java |
1  package com.pig4cloud.pig.admin.utils;
2
3  import com.pig4cloud.pig.admin.api.entity.SysUser;
4
5  import java.util.Map;
6
7  public class BeanCreator {
8
9      private static final String DEFAULT_PASSWORD = "pigmax123456";
10
11     public static SysUser createSysUserByMap(Map<String, Object> map) {
12         SysUser sysUser = new SysUser();
13         String username = (String) map.get("username");
14         String phone = (String) map.get("mobile");
15         String deptId = (String) map.get("departmentId");
16         sysUser.setUsername(username);
17         sysUser.setPhone(phone);
18         sysUser.setDeptId(Long.parseLong(deptId));
19         sysUser.setPassword(DEFAULT_PASSWORD);
20         return sysUser;
21     }
22
23 }

```

vue包

主要是对pig前端页面的修改

api包

新增获取token的方法

```
1 import { validateNull } from '@/util/validate'
2 import request from '@/router/axios'
3 import store from '@/store'
4 import qs from 'qs'
5 import { getStore, setStore } from '@/util/store.js'
6 import website from '@/config/website'
7
8
9 const scope = 'server'
10
11 export const loginByUsername = (username, password, code, randomStr) =>
  {
12   const grant_type = 'password'
13   const dataObj = qs.stringify({ 'username': username, 'password': password })
14
15   const basicAuth = 'Basic ' + window.btoa(website.formLoginClient)
16
17   // 保存当前选中的 basic 认证信息
18   setStore({
19     name: 'basicAuth',
20     content: basicAuth,
21     type: 'session'
22   })
23
24   return request({
25     url: '/auth/oauth2/token',
26     headers: {
27       isToken: false,
28       Authorization: basicAuth
29     },
30     method: 'post',
31     params: { randomStr, code, grant_type, scope },
32     data: dataObj
33   })
34 }
35
36 export const loginByMobile = (smsForm) => {
37   const grant_type = 'app'
38
39   const basicAuth = 'Basic ' + window.btoa(website.smsLoginClient)
40
41   // 保存当前选中的 basic 认证信息
42   setStore({
43     name: 'basicAuth',
```

```

44     content: basicAuth,
45     type: 'session'
46   })
47
48   return request({
49     url: '/auth/oauth2/token',
50     headers: {
51       isToken: false,
52       'Authorization': basicAuth
53     },
54     method: 'post',
55     params: { phone: smsForm.phone, code: smsForm.code, grant_type, sco
56   }
57   })
58 }
59
60 export const ssoLogin = (ticket, service) => {
61   return request({
62     url: '/auth/token/sso_login_get_token',
63     method: 'get',
64     params: { ticket, service }
65   })
66 }
67
68 export const refreshToken = refresh_token => {
69   const grant_type = 'refresh_token'
70   // 获取当前选中的 basic 认证信息
71   const basicAuth = getStore({ name: 'basicAuth' })
72
73   return request({
74     url: '/auth/oauth2/token',
75     headers: {
76       isToken: false,
77       Authorization: basicAuth
78     },
79     method: 'post',
80     params: { refresh_token, grant_type, scope }
81   })
82 }
83
84 export const getUserInfo = () => {
85   return request({
86     url: '/admin/user/info',
87     method: 'get'
88   })
89 }
90
91 export const logout = () => {

```

```

91     return request({
92         url: '/auth/token/logout',
93         method: 'delete'
94     })
95 }
96 }
97
98 /**
99  * 校验令牌，若有效期小于半小时自动续期
100  *
101  * 定时任务请求后端接口返回实际的有效时间，不进行本地计算避免 客户端和服务端机器时钟不
102  * 一致
103  * @param refreshLock
104  */
105 export const checkToken = (refreshLock, $store) => {
106     const token = store.getters.access_token
107     // 获取当前选中的 basic 认证信息
108     const basicAuth = getStore({ name: 'basicAuth' })
109
110     if (validatenull(token) || validatenull(basicAuth)) {
111         return
112     }
113
114     request({
115         url: '/auth/token/check_token',
116         headers: {
117             isToken: false,
118             Authorization: basicAuth
119         },
120         method: 'get',
121         params: { token }
122     }).then(response => {
123         const expire = response.data.expire
124         if (expire) {
125             const expiredPeriod = expire * 1000 - new Date().getTime()
126             console.log('当前token过期时间', expiredPeriod, '毫秒')
127             //小于半小时自动续约
128             if (expiredPeriod <= website.remainingTime) {
129                 if (!refreshLock) {
130                     refreshLock = true
131                     $store.dispatch('RefreshToken')
132                     .catch(() => {
133                         clearInterval(this.refreshTime)
134                     })
135                     refreshLock = false
136                 }
137             }
138         }
139     }).catch(error => {

```

```
138     console.error(error)
139   })
140 }
141
142 ▾   /**
143     * 注册用户
144     */
145 ▾   export const registerUser = (userInfo) => {
146     return request({
147       url: '/admin/register/user',
148       method: 'post',
149       data: userInfo
150     })
151   }
152
153
154 ▾   /**
155     * 发送短信
156     */
157 ▾   export const sendSmsCode = (form) => {
158     return request({
159       url: '/admin/app/sms',
160       method: 'post',
161       data: form
162     })
163   }
164 }
```

userlogin改造

```
1 <template>
2   <el-form
3     ref="loginForm"
4     class="login-form"
5     status-icon
6     :rules="loginRules"
7     :model="loginForm"
8     label-width="0"
9   >
10  <el-form-item prop="username">
11    <el-input
12      v-model="loginForm.username"
13      auto-complete="off"
14      placeholder="请输入用户名"
15      @keyup.enter.native="handleLogin"
16    >
17      <template #prefix>
18        <i class="icon-yonghu"></i>
19      </template>
20    </el-input>
21  </el-form-item>
22  <el-form-item prop="password">
23    <el-input
24      v-model="loginForm.password"
25      size="small"
26      type="password"
27      auto-complete="off"
28      show-password
29      placeholder="请输入密码"
30      @keyup.enter.native="handleLogin"
31    >
32      <template #prefix>
33        <i class="icon-mima"></i>
34      </template>
35    </el-input>
36  </el-form-item>
37  <el-form-item v-if="website.validateCode" prop="code">
38    <el-input
39      v-model="loginForm.code"
40      :maxlength="code.len"
41      auto-complete="off"
42      placeholder="请输入验证码"
43      @keyup.enter.native="handleLogin"
44    >
```

```

46     <template #prefix>
47     <i class="icon-yanzhengma"></i>
48     </template>
49     <template #append>
50     <div class="login-code">
51     <span
52     v-if="code.type === 'text'"
53     class="login-code-img"
54     @click="refreshCode"
55     >{{ code.value }}</span
56     >
57     
63     </div>
64     </template>
65     </el-input>
66     </el-form-item>
67     <el-form-item>
68     <el-button
69     type="primary"
70     class="login-submit"
71     @click.native.prevent="handleLogin"
72     >登录
73     </el-button
74     >
75     </el-form-item>
76
77     </el-form>
78     </template>
79
80     <script>
81     import { randomLenNum } from '@/util'
82     import { mapGetters } from 'vuex'
83     // import { ssoLogin } from '@/api/login'
84     export default {
85     name: 'userlogin',
86     data() {
87     return {
88     loginForm: {
89     username: 'admin',
90     password: '123456',
91     code: '',
92     randomStr: ''
93     },

```

```

94         checked: false,
95     code: {
96         src: '/code',
97         value: '',
98         len: 4,
99         type: 'image'
100     },
101     loginRules: {
102         username: [
103             { required: true, message: '请输入用户名', trigger: 'bl
104 ur' },
105             { pattern: /^[a-z\u4e00-\u9fa5\d]*?$/, message: '请
106 输入小写字母', trigger: 'blur' }
107         ],
108         password: [
109             { required: true, message: '请输入密码', trigger: 'blu
110 r' },
111             { min: 6, message: '密码长度最少为6位', trigger: 'blur'
112         }
113     ],
114     code: [{ required: true, message: '请输入验证码', trigger
115 : 'blur' }]
116     },
117     },
118     created() {
119         var params = new URLSearchParams(window.location.search);
120         var myParam = params.get('ticket');
121         var service = params.get('service');
122         if(myParam!=''&&myParam!=null){
123             this.ssoLogin(myParam,service)
124         }
125     },
126     updated(){
127         var params = new URLSearchParams(window.location.search);
128         var myParam = params.get('ticket');
129         var service = params.get('service');
130         if(myParam!=''&&myParam!=null){
131             this.ssoLogin(myParam,service)
132         }
133     },
134     computed: {
135         ...mapGetters(['tagWel', 'website'])
136     },
137     methods: {
138         async ssoLogin(myParam,service){
139             this.$store
140             .dispatch('SSOLogin', myParam,service)

```



```

137     .then(() => {
138     this.$router.push({ path: this.tagWel.value })
139     })
140     .catch(() => {
141     this.refreshCode()
142     })
143     },
144     refreshCode() {
145     this.loginForm.code = ''
146     this.loginForm.randomStr = randomLenNum(this.code.len, true)
147     this.code.type === 'text'
148     ? (this.code.value = randomLenNum(this.code.len))
149     : (this.code.src = `${this.baseUrl}/code?randomStr=${this.log
inForm.randomStr}`)
150     },
151     handleLogin() {
152     this.$refs.loginForm.validate(valid => {
153     if (valid) {
154     this.$store
155     .dispatch('LoginByUsername', this.loginForm)
156     .then(() => {
157     this.$router.push({ path: this.tagWel.value })
158     })
159     .catch(() => {
160     this.refreshCode()
161     })
162     }
163     })
164     }
165     }
166     }
167     </script>
168
169     <style></style>

```

router改造

如果没权限返回登录页面

```
1 import axios from 'axios'
2 import { serialize } from '@/util'
3 import NProgress from 'nprogress' // progress bar
4 import errorCode from '@/const/errorCode'
5 import { ElMessage, ElMessageBox } from 'element-plus'
6 import 'nprogress/nprogress.css'
7 import qs from 'qs'
8 import store from '@/store'
9 import router from '@router/index.js'
10 import { baseUrl } from '@config/env' // progress bar style
11 axios.defaults.timeout = 30000
12 // 返回其他状态吗
13 axios.defaults.validateStatus = function(status) {
14   return status >= 200 && status <= 500 // 默认的
15 }
16 // 跨域请求, 允许保存cookie
17 axios.defaults.withCredentials = true
18 // NProgress Configuration
19 NProgress.configure({
20   showSpinner: false
21 })
22
23 // HTTPrequest拦截
24 axios.defaults.baseURL = baseUrl
25 axios.interceptors.request.use(config => {
26   NProgress.start() // start progress bar
27   const isToken = (config.headers || {}).isToken === false
28   const token = store.getters.access_token
29
30   if (token && !isToken) {
31     config.headers['Authorization'] = 'Bearer ' + token // token
32   }
33
34   // headers中配置serialize为true开启序列化
35   if (config.method === 'post' && config.headers.serialize) {
36     config.data = serialize(config.data)
37     delete config.data.serialize
38   }
39
40   if (config.method === 'get') {
41     config.paramsSerializer = function(params) {
42       return qs.stringify(params, { arrayFormat: 'repeat' })
43     }
44   }
45   return config
```

```

46   }, error => {
47     return Promise.reject(error)
48   })
49
50   // HTTPResponse拦截
51   axios.interceptors.response.use(res => {
52     NProgress.done()
53     const status = Number(res.status) || 200
54     const message = res.data.msg || errorCode[status] || errorCode['default']
55     if (status === 401){
56       window.open("http://localhost:3000/")
57     }
58
59     // 后台定义 424 针对令牌过去的特殊响应码
60     if (status === 424) {
61       ElMessageBox.confirm('令牌状态已过期, 请点击重新登录', '系统提示', {
62         confirmButtonText: '重新登录',
63         cancelButtonText: '取消',
64         type: 'warning'
65       })
66     }.then(() => {
67       store.dispatch('LogOut').then(() => {
68         // 刷新登录页面, 避免多次弹框
69         window.location.reload()
70       })
71     }).catch(() => {
72     })
73     return
74   }
75
76   if (status !== 200 || res.data.code === 1) {
77     ElMessage({
78       message: message,
79       type: 'error'
80     })
81     return Promise.reject(new Error(message))
82   }
83
84   return res
85 }, error => {
86   // 处理 503 网络异常
87   console.log(error)
88   if (error.response.status === 503) {
89     ElMessage({
90       message: error.response.data.msg,
91       type: 'error'
92     })

```

```
93     }
94     NProgress.done()
95     return Promise.reject(new Error(error))
96   })
97 }
98 export default axios
```

store的user改造

```
1 import { setToken, setRefreshToken } from '@/util/auth'
2 import { getStore, setStore } from '@/util/store'
3 import { ssoLogin, loginByMobile, loginByUsername, getUserInfo, logout, re
  freshToken } from '@/api/login'
4 import { deepClone, encryption } from '@/util'
5 import { formatPath } from '@/router/avue-router'
6 import { getMenu } from '@/api/admin/menu'
7 const user = {
8   state: {
9     userInfo: getStore({
10       name: 'userInfo'
11     }) || {},
12     permissions: getStore({
13       name: 'permissions'
14     }) || [],
15     roles: [],
16     menu: getStore({
17       name: 'menu'
18     }) || [],
19     menuAll: getStore({ name: 'menuAll' }) || [],
20     access_token: getStore({
21       name: 'access_token'
22     }) || '',
23     refresh_token: getStore({
24       name: 'refresh_token'
25     }) || ''
26   },
27   actions: {
28     // SSO单点登陆
29     SSOLogin({ commit }, myParam, service) {
30       return new Promise((resolve, reject) => {
31         ssoLogin(myParam, service).then(response => {
32           const data = response.data.data
33           console.log(data)
34           commit('SET_ACCESS_TOKEN', data.access_token)
35           commit('SET_REFRESH_TOKEN', data.refresh_token)
36           commit('CLEAR_LOCK')
37           resolve()
38         }).catch(error => {
39           reject(error)
40         })
41       })
42     },
43     // 根据用户名登录
44     LoginByUsername({ commit }, userInfo) {
```

```

45     const user = encryption({
46         data: userInfo,
47         key: 'thanks,pig4cloud',
48         param: ['password']
49     })
50     return new Promise((resolve, reject) => {
51         loginByUsername(user.username, user.password, user.code, user.randomStr).then(response => {
52             const data = response.data
53             commit('SET_ACCESS_TOKEN', data.access_token)
54             commit('SET_REFRESH_TOKEN', data.refresh_token)
55             commit('CLEAR_LOCK')
56             resolve()
57         }).catch(error => {
58             reject(error)
59         })
60     })
61 },
62 // 根据手机号登录
63 LoginByPhone({ commit }, smsForm) {
64     return new Promise((resolve, reject) => {
65         loginByMobile(smsForm).then(response => {
66             const data = response.data
67             commit('SET_ACCESS_TOKEN', data.access_token)
68             commit('SET_REFRESH_TOKEN', data.refresh_token)
69             commit('CLEAR_LOCK')
70             resolve()
71         }).catch(error => {
72             reject(error)
73         })
74     })
75 },
76
77 // 刷新token
78 RefreshToken({ commit, state }) {
79     return new Promise((resolve, reject) => {
80         refreshToken(state.refresh_token).then(response => {
81             const data = response.data
82             commit('SET_ACCESS_TOKEN', data.access_token)
83             commit('SET_REFRESH_TOKEN', data.refresh_token)
84             commit('CLEAR_LOCK')
85             resolve()
86         }).catch(error => {
87             reject(error)
88         })
89     })
90 },
91 // 查询用户信息

```

```

92   GetUserInfo({ commit }) {
93     return new Promise((resolve, reject) => {
94       getUserInfo().then((res) => {
95         const data = res.data.data || {}
96         commit('SET_USER_INFO', data.sysUser)
97         commit('SET_ROLES', data.roles || [])
98         commit('SET_PERMISSIONS', data.permissions || [])
99         resolve(data)
100      }).catch(() => {
101        reject()
102      })
103    })
104  },
105  // 登出
106  Logout({ commit }) {
107    return new Promise((resolve, reject) => {
108      logout().then(() => {
109        commit('SET_MENUALL_NULL', [])
110        commit('SET_MENU', [])
111        commit('SET_PERMISSIONS', [])
112        commit('SET_USER_INFO', {})
113        commit('SET_ACCESS_TOKEN', '')
114        commit('SET_REFRESH_TOKEN', '')
115        commit('SET_ROLES', [])
116        commit('DEL_ALL_TAG')
117        commit('CLEAR_LOCK')
118        resolve()
119      }).catch(error => {
120        reject(error)
121      })
122    })
123  },
124  // 注销session
125  FedLogout({ commit }) {
126    return new Promise(resolve => {
127      commit('SET_MENU', [])
128      commit('SET_MENUALL_NULL', [])
129      commit('SET_PERMISSIONS', [])
130      commit('SET_USER_INFO', {})
131      commit('SET_ACCESS_TOKEN', '')
132      commit('SET_REFRESH_TOKEN', '')
133      commit('SET_ROLES', [])
134      commit('DEL_ALL_TAG')
135      commit('CLEAR_LOCK')
136      resolve()
137    })
138  },
139  // 获取系统菜单

```

```

140 GetMenu({ commit }, obj = {}) {
141     // 记录用户点击顶部信息，保证刷新的时候不丢失
142     commit('LIKE_TOP_MENUID', obj)
143     return new Promise(resolve => {
144         getMenu(obj.id).then((res) => {
145             const data = res.data.data
146             const menu = deepClone(data)
147             menu.forEach(ele => formatPath(ele, true))
148             commit('SET_MENUALL', menu)
149             commit('SET_MENU', menu)
150             resolve(menu)
151         })
152     })
153 },
154 //顶部菜单
155 GetTopMenu() {
156     return new Promise(resolve => {
157         resolve([])
158     })
159 }
160 },
161 mutations: {
162     SET_ACCESS_TOKEN: (state, access_token) => {
163         state.access_token = access_token
164         setToken(access_token)
165         setStore({
166             name: 'access_token',
167             content: state.access_token,
168             type: 'session'
169         })
170     },
171     SET_REFRESH_TOKEN: (state, rfToken) => {
172         state.refresh_token = rfToken
173         setRefreshToken(rfToken)
174         setStore({
175             name: 'refresh_token',
176             content: state.refresh_token,
177             type: 'session'
178         })
179     },
180     SET_USER_INFO: (state, userInfo) => {
181         state.userInfo = userInfo
182         setStore({
183             name: 'userInfo',
184             content: userInfo,
185             type: 'session'
186         })
187     },

```



```

188     SET_MENUALL: (state, menuAll) => {
189         const menu = state.menuAll
190         menuAll.forEach(ele => {
191             if (!menu.find(item => item.label === ele.label && item.path ===
192     ele.path)) {
193                 menu.push(ele)
194             }

```

Nacos配置文件新增

在application-dev.yml新增配置信息

```

YAML |
1  #cas配置
2  cas:
3    #密钥
4    key: n0c9MTcwMjIwMjMxNzE2NDMwOTAskV
5    server:
6      host:
7        grant_url: http://sso.maxkey.top/sign/authz/cas
8        #cas服务端地址 这是我的cas服务端地址 需要修改成你们的cas服务端地址
9        url: http://sso.maxkey.top/maxkey/authz/cas
10       #cas服务端登录地址
11       login_url: http://sso.maxkey.top/maxkey/#/passport/login?redirect_ur
12       i=aHR0cDovL3Nzby5tYXhrZXkudG9wL3NpZ24vYXV0aHovY2FzLzQxMDY1ZmUzLWFlNjctNDE3
13       Mi1hNDYwLWZkMDA3OWU4ODI5NA
14       #cas服务端登出地址 service参数后面跟就是需要跳转的页面/接口 这里指定的是cas客
15       户端登录接口
16       logout_url: ${cas.server.host.url}/logout?service=${cas.service.hos
17       t.url}${cas.service.host.login_url}
18     service:
19       host:
20         #cas客户端地址
21         url: http://localhost:8080
22         #cas客户端地址登录地址
23         login_url: /login
24         #cas客户端地址登出地址
25         logout_url: /logout

```