Pig整合MaxKey集成指南

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

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>com.pig4cloud</groupId>
    <artifactId>pig</artifactId>
    <version>3.6.7</version>
  </parent>

  <artifactId>pig-auth</artifactId>
  <packaging>jar</packaging>

  <description>pig 认证授权中心，基于 spring security oAuth2</description>

  <dependencies>
    <dependency>
      <groupId>io.jsonwebtoken</groupId>
      <artifactId>jjwt</artifactId>
      <version>0.7.0</version>
    </dependency>
    <!--注册中心客户端-->
    <dependency>
      <groupId>com.alibaba.cloud</groupId>
      <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    </dependency>
    <!--配置中心客户端-->
    <dependency>
      <groupId>com.alibaba.cloud</groupId>
      <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
    </dependency>
    <!--断路器依赖-->
    <dependency>
      <groupId>com.pig4cloud</groupId>
      <artifactId>pig-common-feign</artifactId>
    </dependency>
    <!--upms api、model 模块-->
    <dependency>
      <groupId>com.pig4cloud</groupId>
      <artifactId>pig-upms-api</artifactId>
    </dependency>
    <dependency>
```

```
        <groupId>com.pig4cloud</groupId>
        <artifactId>pig-common-security</artifactId>
      </dependency>
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
      </dependency>
      <!--freemarker-->
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-freemarker</artifactId>
      </dependency>
      <!--undertow容器-->
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-undertow</artifactId>
      </dependency>
      <!-- log -->
      <dependency>
        <groupId>com.pig4cloud</groupId>
        <artifactId>pig-common-log</artifactId>
      </dependency>
    </dependencies>

    <build>
      <plugins>
        <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
        <plugin>
        <groupId>io.fabric8</groupId>
        <artifactId>docker-maven-plugin</artifactId>
      </plugin>
      </plugins>
      </build>

      </project>
```

PigTokenEndpoint中新增获取token的方法

Java

```java
package com.pig4cloud.pig.auth.endpoint;

import cn.hutool.core.date.DatePattern;
import cn.hutool.core.date.TemporalAccessorUtil;
import cn.hutool.core.map.MapUtil;
import cn.hutool.core.util.StrUtil;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.pig4cloud.pig.admin.api.entity.SysOauthClientDetails;
import com.pig4cloud.pig.admin.api.feign.RemoteClientDetailsService;
import com.pig4cloud.pig.admin.api.vo.TokenVo;
import com.pig4cloud.pig.auth.support.handler.PigAuthenticationFailureEventHandler;
import com.pig4cloud.pig.auth.utils.RedisUtils;
import com.pig4cloud.pig.auth.utils.TokenManager;
import com.pig4cloud.pig.common.core.constant.CacheConstants;
import com.pig4cloud.pig.common.core.constant.CommonConstants;
import com.pig4cloud.pig.common.core.util.R;
import com.pig4cloud.pig.common.core.util.RetOps;
import com.pig4cloud.pig.common.core.util.SpringContextHolder;
import com.pig4cloud.pig.common.security.annotation.Inner;
import com.pig4cloud.pig.common.security.service.PigUser;
import com.pig4cloud.pig.common.security.util.OAuth2EndpointUtils;
import com.pig4cloud.pig.common.security.util.OAuth2ErrorCodesExpand;
import com.pig4cloud.pig.common.security.util.OAuthClientException;
import lombok.RequiredArgsConstructor;
import lombok.SneakyThrows;
import lombok.extern.slf4j.Slf4j;
import org.springframework.cache.CacheManager;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.converter.HttpMessageConverter;
import org.springframework.http.server.ServletServerHttpResponse;
import org.springframework.security.authentication.event.LogoutSuccessEvent;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.oauth2.core.OAuth2AccessToken;
import org.springframework.security.oauth2.core.endpoint.OAuth2AccessTokenResponse;
import org.springframework.security.oauth2.core.endpoint.OAuth2ParameterNames;
import org.springframework.security.oauth2.core.http.converter.OAuth2AccessTokenResponseHttpMessageConverter;
```

```java
import org.springframework.security.oauth2.server.authorization.OAuth2Authorization;
import org.springframework.security.oauth2.server.authorization.OAuth2AuthorizationService;
import org.springframework.security.oauth2.server.authorization.OAuth2TokenType;
import org.springframework.security.oauth2.server.resource.InvalidBearerTokenException;
import org.springframework.security.web.authentication.AuthenticationFailureHandler;
import org.springframework.security.web.authentication.preauth.PreAuthenticatedAuthenticationToken;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.security.Principal;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;

/**
* @author lengleng
* @date 2019/2/1 删除token端点
*/
@Slf4j
@RestController
@RequiredArgsConstructor
@RequestMapping("/token")
public class PigTokenEndpoint {

private final HttpMessageConverter<OAuth2AccessTokenResponse> accessTokenHttpResponseConverter = new OAuth2AccessTokenResponseHttpMessageConverter();

private final AuthenticationFailureHandler authenticationFailureHandler = new PigAuthenticationFailureEventHandler();

private final OAuth2AuthorizationService authorizationService;

private final RemoteClientDetailsService clientDetailsService;

private final RedisTemplate<String, Object> redisTemplate;
```

```
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
```

```
private final CacheManager cacheManager;

@Resource
private RedisUtils redisUtils;


@Resource
private TokenManager tokenManager;

private final static String SPRING_SESSION_PREFIX = "spring:session:sessions:%s";
private final static String PIG_TOKEN_PREFIX = "pig:token:%s:%s";
private final static String ASSCEE_TOKEN = "access_token";
private final static String REFRESH_TOKEN = "refresh_token";

private long tokenExpiration = 24 * 60 * 60 * 1000;


/**
* 认证页面
*
* @param modelAndView
* @param error        表单登录失败处理回调的错误信息
* @return ModelAndView
*/
@GetMapping("/login")
public ModelAndView require(ModelAndView modelAndView, @RequestParam(required = false) String error) {
modelAndView.setViewName("ftl/login");
modelAndView.addObject("error", error);
return modelAndView;
}

@GetMapping("/confirm_access")
public ModelAndView confirm(Principal principal, ModelAndView modelAndView,
@RequestParam(OAuth2ParameterNames.CLIENT_ID) String clientId,
@RequestParam(OAuth2ParameterNames.SCOPE) String scope,
@RequestParam(OAuth2ParameterNames.STATE) String state) {
SysOauthClientDetails clientDetails = RetOps.of(clientDetailsService.getClientDetailsById(clientId))
.getData()
.orElseThrow(() -> new OAuthClientException("clientId 不合法"));

Set<String> authorizedScopes = StringUtils.commaDelimitedListToSet(clientDetails.getScope());
modelAndView.addObject("clientId", clientId);
```

```
modelAndView.addObject("state", state);
modelAndView.addObject("scopeList", authorizedScopes);
modelAndView.addObject("principalName", principal.getName());
modelAndView.setViewName("ftl/confirm");
return modelAndView;
}

/**
* 退出并删除token
*
* @param authHeader Authorization
*/
@DeleteMapping("/logout")
public R<Boolean> logout(HttpServletRequest request, @RequestHeader(value
 = HttpHeaders.AUTHORIZATION, required = false) String authHeader) {
if (StrUtil.isBlank(authHeader)) {
return R.ok();
}
String sessonId = request.getSession().getId();
if (StrUtil.isBlank(sessonId)) {
return R.ok();
}
boolean isSuccess = redisUtils.deleteKey(generateSessionId(sessonId));
if (isSuccess) {
return R.ok();
} else {
return R.failed();
}

}

/**
* 校验token
*
* @param token 令牌
*/
@SneakyThrows
@GetMapping("/check_token")
public void checkToken(String token, HttpServletResponse response, HttpSe
rvletRequest request) {

ServletServerHttpResponse httpResponse = new ServletServerHttpResponse(re
sponse);

if (StrUtil.isBlank(token)) {
httpResponse.setStatusCode(HttpStatus.UNAUTHORIZED);
this.authenticationFailureHandler.onAuthenticationFailure(request, respon
se,
```

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

TokenMananer工具类

```java
package com.pig4cloud.pig.auth.utils;

import io.jsonwebtoken.CompressionCodecs;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.stereotype.Component;

import java.util.Date;

@Component
public class TokenManager {
    private long tokenExpiration = 24 * 60 * 60 * 1000;
    private final static String TOKEN_SIGN_KEY = "MAKKEY_PIG";

    public String createToken(String subject, String username, String id)
{
        String token = Jwts.builder()
                .setSubject(subject)
                .claim("nickname", username)
                .claim("id", id)
                .setExpiration(new Date(System.currentTimeMillis() + token
Expiration))
                .signWith(SignatureAlgorithm.HS512, TOKEN_SIGN_KEY)
                .compressWith(CompressionCodecs.GZIP).compact();
        return token;
    }


    public String getUserFromToken(String token) {
        String user = Jwts.parser().setSigningKey(TOKEN_SIGN_KEY).parseCla
imsJws(token).getBody().getSubject();
        return user;
    }

    public void removeToken(String token) {
        //jwttoken无需删除，客户端扔掉即可。
    }

}
```

2.pig-common

pom文件的更改

主要新增 CAS的MAVEN包

Java

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>com.pig4cloud</groupId>
        <artifactId>pig-common</artifactId>
        <version>3.6.7</version>
    </parent>

    <artifactId>pig-common-security</artifactId>
    <packaging>jar</packaging>

    <description>pig 安全工具类</description>


    <dependencies>
        <dependency>
            <groupId>org.springframework.session</groupId>
            <artifactId>spring-session-data-redis</artifactId>
        </dependency>
        <!--        spring security cas-->
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-cas</artifactId>
        </dependency>
        <!--工具类核心包-->
        <dependency>
            <groupId>com.pig4cloud</groupId>
            <artifactId>pig-common-core</artifactId>
        </dependency>
        <dependency>
            <groupId>cn.hutool</groupId>
            <artifactId>hutool-extra</artifactId>
        </dependency>
        <!--UPMS API-->
        <dependency>
            <groupId>com.pig4cloud</groupId>
            <artifactId>pig-upms-api</artifactId>
        </dependency>
        <!--common utils-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
```

```
            <artifactId>spring-cloud-commons</artifactId>
        </dependency>
        <!--feign 工具类-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-oauth2-jose</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-oauth2-authorization-server</artifactId>
            <version>${spring.authorization.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
        </dependency>
    </dependencies>
</project>
```

annotation包下面更改EnablePigResourceServer

```java
package com.pig4cloud.pig.common.security.annotation;

import com.pig4cloud.pig.common.security.component.PigResourceServerAutoConfiguration;
import com.pig4cloud.pig.common.security.component.PigResourceServerConfiguration;
import com.pig4cloud.pig.common.security.component.ResourceAuthExceptionEntryPoint;
import com.pig4cloud.pig.common.security.config.*;
import org.springframework.context.annotation.Import;

import java.lang.annotation.*;

/**
 * @author lengleng
 * @date 2022-06-04
 * <p>
 * 资源服务注解
 */
@Documented
@Inherited
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
//@Import({ PigResourceServerAutoConfiguration.class, PigResourceServerConfiguration.class })
@Import({PigResourceServerAutoConfiguration.class, CasProperties.class, SecurityConfig.class})
public @interface EnablePigResourceServer {

}

```

Config包下

新增CasProperties主要是CAS得配置信息配置在NACOS中

Java

```java
package com.pig4cloud.pig.common.security.config;

import lombok.Data;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Data
@Component
public class CasProperties {

    /**
     * 秘钥
     */
    @Value("${cas.key}")
    private String casKey;

    /**
     * cas服务端地址
     */
    @Value("${cas.server.host.url}")
    private String casServerUrl;

    /**
     * cas服务端地址
     */
    @Value("${cas.server.host.grant_url}")
    private String casGrantingUrl;

    /**
     * cas服务端登录地址
     */
    @Value("${cas.server.host.login_url}")
    private String casServerLoginUrl;

    /**
     * cas服务端登出地址 并回跳到制定页面
     */
    @Value("${cas.server.host.logout_url}")
    private String casServerLogoutUrl;

    /**
     * cas客户端地址
     */
    @Value("${cas.service.host.url}")
    private String casServiceUrl;
```

```
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

```
    /**
     * cas客户端地址登录地址
     */
    @Value("${cas.service.host.login_url}")
    private String casServiceLoginUrl;

    /**
     * cas客户端地址登出地址
     */
    @Value("${cas.service.host.logout_url}")
    private String casServiceLogoutUrl;

}
```

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

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

```java
package com.pig4cloud.pig.common.security.config;

import cn.hutool.core.util.ArrayUtil;
import com.pig4cloud.pig.common.security.component.PermitAllUrlProperties;
import com.pig4cloud.pig.common.security.component.PigBearerTokenExtractor;
import com.pig4cloud.pig.common.security.component.ResourceAuthExceptionEntryPoint;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.jasig.cas.client.session.SingleSignOutFilter;
import org.jasig.cas.client.validation.Cas20ServiceTicketValidator;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.security.cas.ServiceProperties;
import org.springframework.security.cas.authentication.CasAuthenticationProvider;
import org.springframework.security.cas.web.CasAuthenticationEntryPoint;
import org.springframework.security.cas.web.CasAuthenticationFilter;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.AuthenticationUserDetailsService;
import org.springframework.security.web.authentication.logout.LogoutFilter;
import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;

@Slf4j
@Configuration
@EnableWebSecurity // 启用web权限
@EnableGlobalMethodSecurity(prePostEnabled = true) // 启用方法验证
@RequiredArgsConstructor
public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

```
34
35      @Autowired
36      private CasProperties casProperties;
37
38      @Autowired
39      private AuthenticationUserDetailsService casUserDetailService;
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 Ex
    ception {
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(), St
    ring.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
    Filter.class);
71          // 取消跨站请求伪造防护
72          http.csrf().disable();
73  //      // 防止iframe 造成跨域
74          http.headers().frameOptions().disable();
75          // http.csrf().disable(); //禁用CSRF
76      }
77
78      /**
```

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

```
    public CasAuthenticationProvider casAuthenticationProvider() {
        CasAuthenticationProvider casAuthenticationProvider = new CasAuthenticationProvider();
        casAuthenticationProvider.setAuthenticationUserDetailsService(casUserDetailService);
        // //这里只是接口类型，实现的接口不一样，都可以的。
        casAuthenticationProvider.setServiceProperties(serviceProperties());
        casAuthenticationProvider.setTicketValidator(cas20ServiceTicketValidator());
        casAuthenticationProvider.setKey("casAuthenticationProviderKey");
        return casAuthenticationProvider;
    }




    @Bean
    public Cas20ServiceTicketValidator cas20ServiceTicketValidator() {
        return new Cas20ServiceTicketValidator(casProperties.getCasGrantingUrl());
    }

    /**
     * 单点登出过滤器
     */
    @Bean
    public SingleSignOutFilter singleSignOutFilter() {
        SingleSignOutFilter singleSignOutFilter = new SingleSignOutFilter();
        singleSignOutFilter.setLogoutCallbackPath(casProperties.getCasServerUrl());
        singleSignOutFilter.setIgnoreInitConfiguration(true);
        return singleSignOutFilter;
    }

    /**
     * 请求单点退出过滤器
     */
    @Bean
    public LogoutFilter casLogoutFilter() {
        LogoutFilter logoutFilter = new LogoutFilter(casProperties.getCasServerLogoutUrl(), new SecurityContextLogoutHandler());
        logoutFilter.setFilterProcessesUrl(casProperties.getCasServiceLogoutUrl());
        return logoutFilter;
    }
}
```

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

Java

```java
/*
 * Copyright (c) 2020 pig4cloud Authors. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.pig4cloud.pig.common.security.service;

import com.pig4cloud.pig.admin.api.dto.UserInfo;
import com.pig4cloud.pig.admin.api.feign.RemoteUserService;
import com.pig4cloud.pig.common.core.constant.CacheConstants;
import com.pig4cloud.pig.common.core.util.R;
import lombok.RequiredArgsConstructor;
import lombok.SneakyThrows;
import lombok.extern.slf4j.Slf4j;
import org.springframework.cache.Cache;
import org.springframework.cache.CacheManager;
import org.springframework.context.annotation.Primary;
import org.springframework.security.cas.authentication.CasAssertionAuthenticationToken;
import org.springframework.security.core.userdetails.AuthenticationUserDetailsService;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

import java.util.*;

/**
 * 用户详细信息
 *
 * @author lengleng hccake
 */
@Slf4j
```

```java
@Primary
@RequiredArgsConstructor
public class PigUserDetailsServiceImpl implements PigUserDetailsService, AuthenticationUserDetailsService<CasAssertionAuthenticationToken> {

    private final RemoteUserService remoteUserService;

    private final CacheManager cacheManager;

    /**
     * 用户名密码登录
     *
     * @param username 用户名
     * @return
     */
    @Override
    @SneakyThrows
    public UserDetails loadUserByUsername(String username) {
        Cache cache = cacheManager.getCache(CacheConstants.USER_DETAILS);
        if (cache != null && cache.get(username) != null) {
            return (PigUser) cache.get(username).get();
        }
        return getUserDetails(remoteUserService.info(username), username);
    }

    @Override
    public int getOrder() {
        return Integer.MIN_VALUE;
    }

    @Override
    public UserDetails loadUserDetails(CasAssertionAuthenticationToken token) throws UsernameNotFoundException {
        log.info("getCredentials:{}", token.getCredentials());
        String username = token.getName();
        Cache cache = cacheManager.getCache(CacheConstants.USER_DETAILS);
        if (cache != null && cache.get(username) != null) {
            return (PigUser) cache.get(username).get();
        }
        R<UserInfo> result = remoteUserService.saveIfNotExist(token.getAssertion().getPrincipal().getAttributes());
        return getUserDetails(result, username);
    }

    private UserDetails getUserDetails(R<UserInfo> result, String username) {
        Cache cache = cacheManager.getCache(CacheConstants.USER_DETAILS);
        UserDetails userDetails = getUserDetails(result);
```

```
        if (cache != null) {
            cache.put(username, userDetails);
        }
        return userDetails;
    }
}
```

utils包

新增BeanCreator方法，主要创建SysUser对象工具类

```java
package com.pig4cloud.pig.admin.utils;

import com.pig4cloud.pig.admin.api.entity.SysUser;

import java.util.Map;

public class BeanCreator {

    private static final String DEFAULT_PASSWORD = "pigmax123456";

    public static SysUser createSysUserByMap(Map<String, Object> map) {
        SysUser sysUser = new SysUser();
        String username = (String) map.get("username");
        String phone = (String) map.get("mobile");
        String deptId = (String) map.get("departmentId");
        sysUser.setUsername(username);
        sysUser.setPhone(phone);
        sysUser.setDeptId(Long.parseLong(deptId));
        sysUser.setPassword(DEFAULT_PASSWORD);
        return sysUser;
    }

}
```

vue包

主要是对pig前端页面的修改

api包

新增获取token的方法

```Vue
import { validatenull } from '@/util/validate'
  import request from '@/router/axios'
  import store from '@/store'
  import qs from 'qs'
  import { getStore, setStore } from '@/util/store.js'
  import website from '@/config/website'


  const scope = 'server'

  export const loginByUsername = (username, password, code, randomStr) => {
    const grant_type = 'password'
    const dataObj = qs.stringify({ 'username': username, 'password': password })

    const basicAuth = 'Basic ' + window.btoa(website.formLoginClient)

    // 保存当前选中的 basic 认证信息
    setStore({
      name: 'basicAuth',
      content: basicAuth,
      type: 'session'
    })

    return request({
      url: '/auth/oauth2/token',
      headers: {
        isToken: false,
        Authorization: basicAuth
      },
      method: 'post',
      params: { randomStr, code, grant_type, scope },
      data: dataObj
    })
  }

  export const loginByMobile = (smsForm) => {
    const grant_type = 'app'

    const basicAuth = 'Basic ' + window.btoa(website.smsLoginClient)

    // 保存当前选中的 basic 认证信息
    setStore({
      name: 'basicAuth',
```

```
      content: basicAuth,
      type: 'session'
    })

    return request({
      url: '/auth/oauth2/token',
      headers: {
        isToken: false,
        'Authorization': basicAuth
      },
      method: 'post',
      params: { phone: smsForm.phone, code: smsForm.code, grant_type, sco
pe }
    })
  }

  export const ssoLogin = (ticket,service) => {
    return request({
      url: '/auth/token/sso_login_get_token',
      method: 'get',
      params: { ticket, service }
    })
  }

  export const refreshToken = refresh_token => {
    const grant_type = 'refresh_token'
    // 获取当前选中的 basic 认证信息
    const basicAuth = getStore({ name: 'basicAuth' })

    return request({
      url: '/auth/oauth2/token',
      headers: {
        isToken: false,
        Authorization: basicAuth
      },
      method: 'post',
      params: { refresh_token, grant_type, scope }
    })
  }

  export const getUserInfo = () => {
    return request({
      url: '/admin/user/info',
      method: 'get'
    })
  }

  export const logout = () => {
```

```
    return request({
      url: '/auth/token/logout',
      method: 'delete'
    })
  }

  /**
  * 校验令牌，若有效期小于半小时自动续期
  *
  * 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 客户端和服务器机器时钟不一致
  * @param refreshLock
  */
  export const checkToken = (refreshLock, $store) => {
    const token = store.getters.access_token
    // 获取当前选中的 basic 认证信息
    const basicAuth = getStore({ name: 'basicAuth' })

    if (validatenull(token) || validatenull(basicAuth)) {
      return
    }

    request({
      url: '/auth/token/check_token',
      headers: {
        isToken: false,
        Authorization: basicAuth
      },
      method: 'get',
      params: { token }
    }).then(response => {
      const expire = response && response.data && response.data.exp
      if (expire) {
        const expiredPeriod = expire * 1000 - new Date().getTime()
        console.log('当前token过期时间', expiredPeriod, '毫秒')
        //小于半小时自动续约
        if (expiredPeriod <= website.remainingTime) {
          if (!refreshLock) {
            refreshLock = true
            $store.dispatch('RefreshToken')
              .catch(() => {
                clearInterval(this.refreshTime)
                })
                refreshLock = false
                }
                }
                }).catch(error => {
```

```
        console.error(error)
    })
}

/**
* 注册用户
*/
export const registerUser = (userInfo) => {
return request({
url: '/admin/register/user',
method: 'post',
data: userInfo
})
}


/**
* 发送短信
*/
export const sendSmsCode = (form) => {
return request({
url: '/admin/app/sms',
method: 'post',
data: form
})
}
```

userlogin改造

Vue

```vue
<template>
  <el-form
    ref="loginForm"
    class="login-form"
    status-icon
    :rules="loginRules"
    :model="loginForm"
    label-width="0"
    >
    <el-form-item prop="username">
      <el-input
        v-model="loginForm.username"
        auto-complete="off"
        placeholder="请输入用户名"
        @keyup.enter.native="handleLogin"
        >
        <template #prefix>
          <i class="icon-yonghu"></i>
        </template>
                    </el-input>
                      </el-form-item>
                      <el-form-item prop="password">
                      <el-input
          v-model="loginForm.password"
          size="small"
          type="password"
          auto-complete="off"
          show-password
          placeholder="请输入密码"
          @keyup.enter.native="handleLogin"
            >
            <template #prefix>
            <i class="icon-mima"></i>
            </template>

            </el-input>
            </el-form-item>
            <el-form-item v-if="website.validateCode" prop="code">
            <el-input
          v-model="loginForm.code"
          :maxlength="code.len"
          auto-complete="off"
          placeholder="请输入验证码"
          @keyup.enter.native="handleLogin"
            >
```

```
            <template #prefix>
            <i class="icon-yanzhengma"></i>
            </template>
            <template #append>
            <div class="login-code">
            <span
          v-if="code.type === 'text'"
          class="login-code-img"
          @click="refreshCode"
            >{{ code.value }}</span
            >
            <img
          v-else
            :src="code.src"
          class="login-code-img"
          @click="refreshCode"
            />
            </div>
            </template>
            </el-input>
            </el-form-item>
            <el-form-item>
            <el-button
          type="primary"
          class="login-submit"
          @click.native.prevent="handleLogin"
            >登录
            </el-button
            >
            </el-form-item>

            </el-form>
            </template>

            <script>
            import { randomLenNum } from '@/util'
          import { mapGetters } from 'vuex'
          // import {ssoLogin} from '@/api/login'
          export default {
            name: 'userlogin',
            data() {
              return {
                loginForm: {
                  username: 'admin',
                  password: '123456',
                  code: '',
                  randomStr: ''
                },
```

```
                checked: false,
                code: {
                  src: '/code',
                  value: '',
                  len: 4,
                  type: 'image'
                },
                loginRules: {
                  username: [
                    { required: true, message: '请输入用户名', trigger: 'blur' },
                    { pattern: /^([a-z\u4e00-\u9fa5\d]*?)$/, message: '请输入小写字母', trigger: 'blur' }
                  ],
                  password: [
                    { required: true, message: '请输入密码', trigger: 'blur' },
                    { min: 6, message: '密码长度最少为6位', trigger: 'blur' }
                  ],
                  code: [{ required: true, message: '请输入验证码', trigger: 'blur' }]
                }
              }
            },
            created() {
              var params = new URLSearchParams(window.location.search);
              var myParam = params.get('ticket');
              var service = params.get('service');
              if(myParam!=''&&myParam!=null){
                this.ssoLogin(myParam,service)
              }
            },
            updated(){
            var params = new URLSearchParams(window.location.search);
            var myParam = params.get('ticket');
            var service = params.get('service');
            if(myParam!=''&&myParam!=null){
            this.ssoLogin(myParam,service)
            }
            },
            computed: {
            ...mapGetters(['tagWel', 'website'])
            },
            methods: {
            async ssoLogin(myParam,service){
            this.$store
            .dispatch('SSOLogin', myParam,service)
```

```
            .then(() => {
            this.$router.push({ path: this.tagWel.value })
            })
            .catch(() => {
            this.refreshCode()
            })
            },
            refreshCode() {
            this.loginForm.code = ''
            this.loginForm.randomStr = randomLenNum(this.code.len, true)
            this.code.type === 'text'
            ? (this.code.value = randomLenNum(this.code.len))
            : (this.code.src = `${this.baseUrl}/code?randomStr=${this.log
inForm.randomStr}`)
            },
            handleLogin() {
            this.$refs.loginForm.validate(valid => {
            if (valid) {
            this.$store
            .dispatch('LoginByUsername', this.loginForm)
            .then(() => {
            this.$router.push({ path: this.tagWel.value })
            })
            .catch(() => {
            this.refreshCode()
            })
            }
            })
            }
            }
            }
            </script>

            <style></style>
```

router改造

如果没权限返回登录页面

Vue

```vue
import axios from 'axios'
import { serialize } from '@/util'
import NProgress from 'nprogress' // progress bar
import errorCode from '@/const/errorCode'
import { ElMessage, ElMessageBox } from 'element-plus'
import 'nprogress/nprogress.css'
import qs from 'qs'
import store from '@/store'
import router from '@/router/index.js'
import { baseUrl } from '@/config/env' // progress bar style
axios.defaults.timeout = 30000
// 返回其他状态吗
axios.defaults.validateStatus = function(status) {
  return status >= 200 && status <= 500 // 默认的
}
// 跨域请求，允许保存cookie
axios.defaults.withCredentials = true
// NProgress Configuration
NProgress.configure({
  showSpinner: false
})

// HTTPrequest拦截
axios.defaults.baseURL = baseUrl
axios.interceptors.request.use(config => {
  NProgress.start() // start progress bar
  const isToken = (config.headers || {}).isToken === false
  const token = store.getters.access_token

  if (token && !isToken) {
    config.headers['Authorization'] = 'Bearer ' + token// token
  }

  // headers中配置serialize为true开启序列化
  if (config.method === 'post' && config.headers.serialize) {
    config.data = serialize(config.data)
    delete config.data.serialize
  }

  if (config.method === 'get') {
    config.paramsSerializer = function(params) {
      return qs.stringify(params, { arrayFormat: 'repeat' })
    }
  }
  return config
```

```
}, error => {
  return Promise.reject(error)
})

// HTTPresponse拦截
axios.interceptors.response.use(res => {
  NProgress.done()
  const status = Number(res.status) || 200
  const message = res.data.msg || errorCode[status] || errorCode['default']
  if (status == 401){
    window.open("http://localhost:3000/")
  }

  // 后台定义 424 针对令牌过去的特殊响应码
  if (status === 424) {
    ElMessageBox.confirm('令牌状态已过期，请点击重新登录', '系统提示', {
      confirmButtonText: '重新登录',
      cancelButtonText: '取消',
      type: 'warning'
    }
    ).then(() => {
      store.dispatch('LogOut').then(() => {
        // 刷新登录页面，避免多次弹框
        window.location.reload()
      })
    }).catch(() => {
    })
    return
  }

  if (status !== 200 || res.data.code === 1) {
    ElMessage({
      message: message,
      type: 'error'
    })
    return Promise.reject(new Error(message))
  }

  return res
}, error => {
  // 处理 503 网络异常
  console.log(error)
  if (error.response.status === 503) {
    ElMessage({
      message: error.response.data.msg,
      type: 'error'
    })
```

```
    }
    NProgress.done()
    return Promise.reject(new Error(error))
  })

  export default axios
```

store的user改造

Vue

```vue
import { setToken, setRefreshToken } from '@/util/auth'
import { getStore, setStore } from '@/util/store'
import { ssoLogin,loginByMobile, loginByUsername, getUserInfo, logout, refreshToken } from '@/api/login'
import { deepClone, encryption } from '@/util'
import { formatPath } from '@/router/avue-router'
import { getMenu } from '@/api/admin/menu'
const user = {
  state: {
    userInfo: getStore({
      name: 'userInfo'
    }) || {},
    permissions: getStore({
      name: 'permissions'
    }) || [],
    roles: [],
    menu: getStore({
      name: 'menu'
    }) || [],
    menuAll: getStore({ name: 'menuAll' }) || [],
    access_token: getStore({
      name: 'access_token'
    }) || '',
    refresh_token: getStore({
      name: 'refresh_token'
    }) || ''
  },
  actions: {
    // SSO单点登陆
    SSOLogin({ commit }, myParam,service) {
      return new Promise((resolve, reject) => {
        ssoLogin(myParam,service).then(response => {
          const data = response.data.data
          console.log(data)
          commit('SET_ACCESS_TOKEN', data.access_token)
          commit('SET_REFRESH_TOKEN', data.refresh_token)
          commit('CLEAR_LOCK')
          resolve()
        }).catch(error => {
          reject(error)
        })
      })
    },
    // 根据用户名登录
    LoginByUsername({ commit }, userInfo) {
```

```
      const user = encryption({
        data: userInfo,
        key: 'thanks,pig4cloud',
        param: ['password']
      })
      return new Promise((resolve, reject) => {
        loginByUsername(user.username, user.password, user.code, user.randomStr).then(response => {
          const data = response.data
          commit('SET_ACCESS_TOKEN', data.access_token)
          commit('SET_REFRESH_TOKEN', data.refresh_token)
          commit('CLEAR_LOCK')
          resolve()
        }).catch(error => {
          reject(error)
        })
      })
    },
    // 根据手机号登录
    LoginByPhone({ commit }, smsForm) {
      return new Promise((resolve, reject) => {
        loginByMobile(smsForm).then(response => {
          const data = response.data
          commit('SET_ACCESS_TOKEN', data.access_token)
          commit('SET_REFRESH_TOKEN', data.refresh_token)
          commit('CLEAR_LOCK')
          resolve()
        }).catch(error => {
          reject(error)
        })
      })
    },

    // 刷新token
    RefreshToken({ commit, state }) {
      return new Promise((resolve, reject) => {
        refreshToken(state.refresh_token).then(response => {
          const data = response.data
          commit('SET_ACCESS_TOKEN', data.access_token)
          commit('SET_REFRESH_TOKEN', data.refresh_token)
          commit('CLEAR_LOCK')
          resolve()
        }).catch(error => {
          reject(error)
        })
      })
    },
    // 查询用户信息
```

```
    GetUserInfo({ commit }) {
      return new Promise((resolve, reject) => {
        getUserInfo().then((res) => {
          const data = res.data.data || {}
          commit('SET_USER_INFO', data.sysUser)
          commit('SET_ROLES', data.roles || [])
          commit('SET_PERMISSIONS', data.permissions || [])
          resolve(data)
        }).catch(() => {
          reject()
        })
      })
    },
    // 登出
    LogOut({ commit }) {
      return new Promise((resolve, reject) => {
        logout().then(() => {
          commit('SET_MENUALL_NULL', [])
          commit('SET_MENU', [])
          commit('SET_PERMISSIONS', [])
          commit('SET_USER_INFO', {})
          commit('SET_ACCESS_TOKEN', '')
          commit('SET_REFRESH_TOKEN', '')
          commit('SET_ROLES', [])
          commit('DEL_ALL_TAG')
          commit('CLEAR_LOCK')
          resolve()
        }).catch(error => {
          reject(error)
        })
      })
    },
    // 注销session
    FedLogOut({ commit }) {
      return new Promise(resolve => {
        commit('SET_MENU', [])
        commit('SET_MENUALL_NULL', [])
        commit('SET_PERMISSIONS', [])
        commit('SET_USER_INFO', {})
        commit('SET_ACCESS_TOKEN', '')
        commit('SET_REFRESH_TOKEN', '')
        commit('SET_ROLES', [])
        commit('DEL_ALL_TAG')
        commit('CLEAR_LOCK')
        resolve()
      })
    },
    // 获取系统菜单
```

```
    GetMenu({ commit }, obj = {}) {
      // 记录用户点击顶部信息，保证刷新的时候不丢失
      commit('LIKE_TOP_MENUID', obj)
      return new Promise(resolve => {
        getMenu(obj.id).then((res) => {
          const data = res.data.data
          const menu = deepClone(data)
          menu.forEach(ele => formatPath(ele, true))
          commit('SET_MENUALL', menu)
          commit('SET_MENU', menu)
          resolve(menu)
        })
      })
    },
    //顶部菜单
    GetTopMenu() {
      return new Promise(resolve => {
        resolve([])
      })
    }
  },
  mutations: {
    SET_ACCESS_TOKEN: (state, access_token) => {
      state.access_token = access_token
      setToken(access_token)
      setStore({
        name: 'access_token',
        content: state.access_token,
        type: 'session'
      })
    },
    SET_REFRESH_TOKEN: (state, rfToken) => {
      state.refresh_token = rfToken
      setRefreshToken(rfToken)
      setStore({
        name: 'refresh_token',
        content: state.refresh_token,
        type: 'session'
      })
    },
    SET_USER_INFO: (state, userInfo) => {
      state.userInfo = userInfo
      setStore({
        name: 'userInfo',
        content: userInfo,
        type: 'session'
      })
    },
```

```
    SET_MENUALL: (state, menuAll) => {
      const menu = state.menuAll
      menuAll.forEach(ele => {
        if (!menu.find(item => item.label === ele.label && item.path === ele.path)) {
          menu.push(ele)
        }
```

Nacos配置文件新增

在application-dev.yml新增配置信息

YAML

```yaml
#cas配置
cas:
  #秘钥
  key: n0c9MTcwMjIwMjMxNzE2NDMwOTAskV
  server:
    host:
      grant_url: http://sso.maxkey.top/sign/authz/cas
      #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
      url: http://sso.maxkey.top/maxkey/authz/cas
      #cas服务端登录地址
      login_url: http://sso.maxkey.top/maxkey/#/passport/login?redirect_uri=aHR0cDovL3Nzby5tYXhrZXkudG9wL3NpZ24vYXV0aHovY2FzLzQxMDY1ZmUzLWFlNjctNDE3Mi1hNDYwLWZkMDA3OWU4ODI5NA
      #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客户端登录接口
      logout_url: ${cas.server.host.url}/logout?service=${cas.service.host.url}${cas.service.host.login_url}
  service:
    host:
      #cas客户端地址
      url: http://localhost:8080
      #cas客户端地址登录地址
      login_url: /login
      #cas客户端地址登出地址
      logout_url: /logout
```