后台权限管理解决方案

修订日期	版本号	描述	修改人
2020-07-16	V0.1	初稿	拉勾教育研究院

一、权限管理的意义

后台管理系统中，通常需要控制不同的登录用户可以操作的内容。权限管理用于管理系统资源，分配用户菜单、资源权限，以及验证用户是否有访问资源权限。

二、权限管理相关概念

2.1 资源管理

1. 资源是权限控制的对象。资源可以是页面跳转路由，也可以是后台管理功能的接口 url。资源管理就是将页面路由、后台功能接口录入到权限数据库，并实现增删改查功能。方便新增模块或下线模块的资源管理。
2. 为方便资源分类，设计了资源分类功能，可以将一个模块下的资源归到一个分类中。
3. 当用户拥有资源权限时，用户访问资源不受限制。否则访问未授权资源时网关进行拦截，并返回访问受限错误信息。

2.2 菜单管理

1. 菜单是一组资源或页面的操作入口，也可以理解为模块或分类。支持多级菜单结构。可以设置排序和是否显示。
2. 菜单和资源没有直接关系。
3. 在前后端分离的架构下，菜单的 url 主要指页面跳转路由。
4. 菜单也作为权限控制的一个维度，粒度较粗，可以动态控制用户的菜单展示。用户登录到后台管理系统后会获取该用户关联的菜单列表并展示，没有授权的菜单不展示。

2.3 角色管理

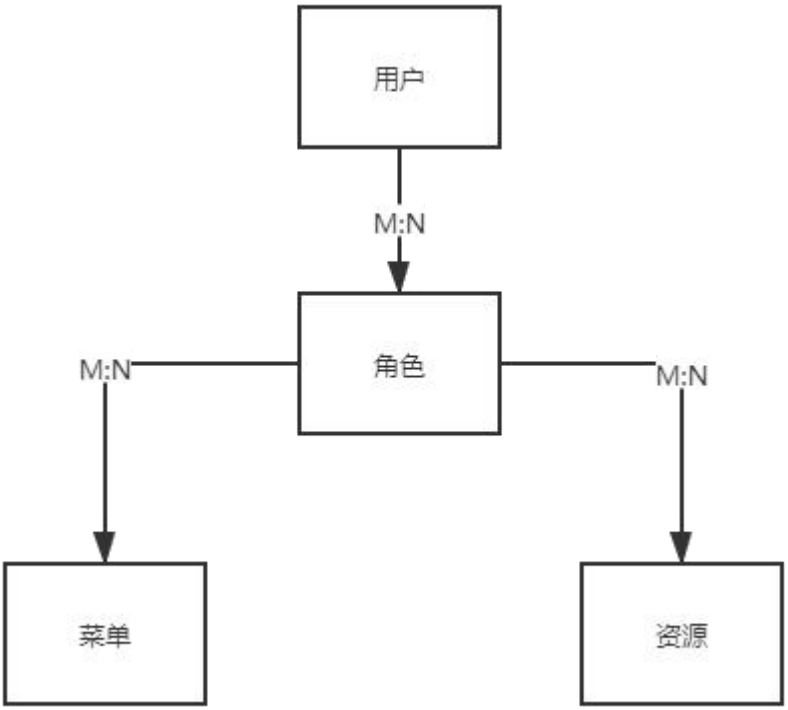
1. 角色是资源或菜单权限的集合。通过角色对不同管理员分配不同的资源、菜单权限。拥有相同权限的用户可以访问相同的菜单和资源。可以理解为权限分组。
2. 通过对用户分配角色，来最终实现用户的访问权限控制。一个用户可分配多个角色，这

些角色的资源、菜单的并集即是用户可访问的全部资源。

三、权限管理的设计

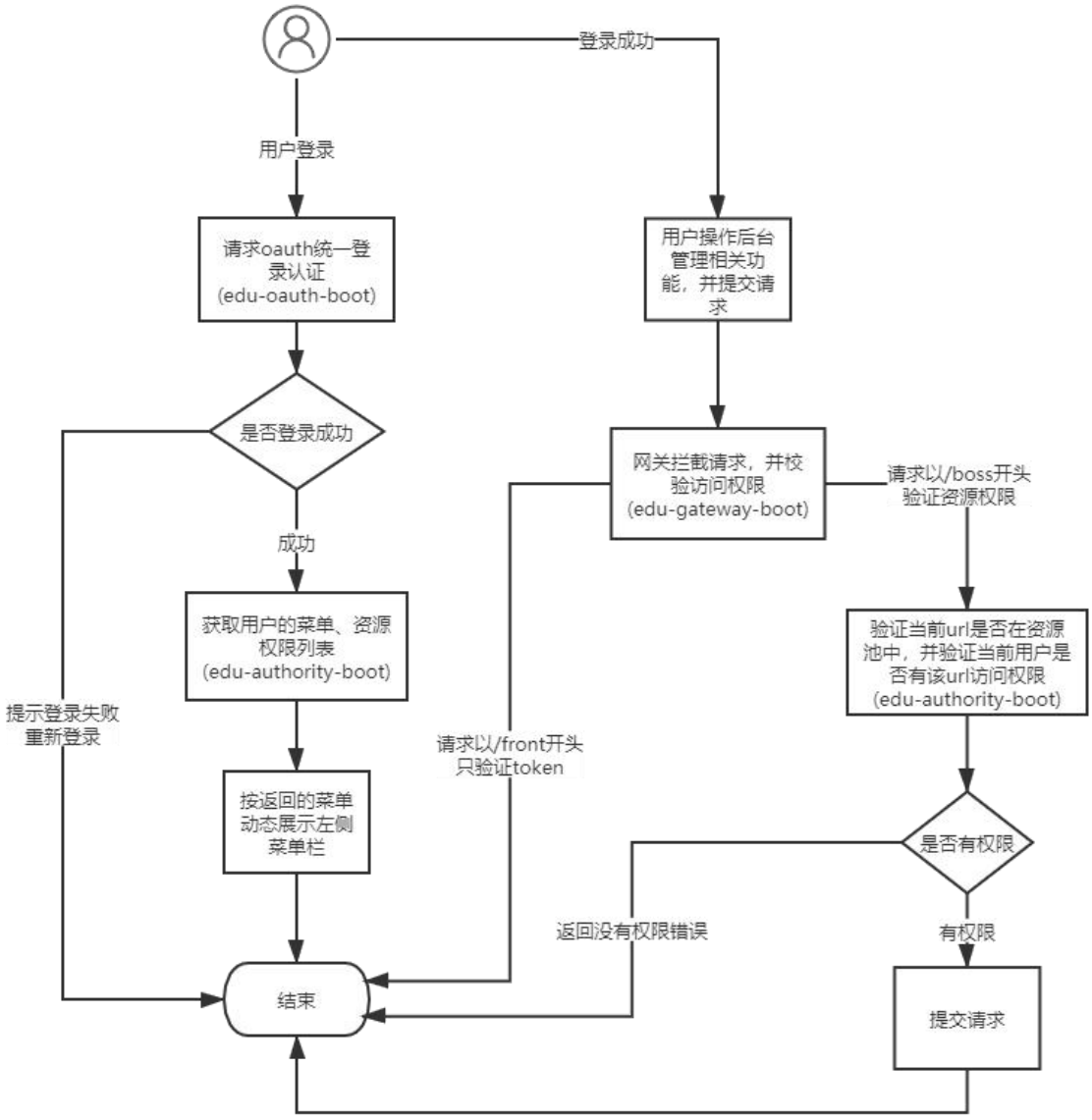
3.1 权限管理的核心控制组件

1. 权限管理包含用户、角色、菜单、资源四个核心组件。通过建立这些组件之间的关联关系，来实现用户到菜单、资源的权限控制。
2. 项目中将前后端用户统一起来，后台管理用户不再单独设置用户表。用户都保存在 edu_user.user。这样方便前后端登录统一使用 oauth。
3. 权限管理组件的关系图：



3.2 权限验证的流程

权限验证包括两方面内容：
1. 后台用户登录获取授权菜单、资源，前端根据返回的菜单列表动态展示左侧菜单列表。同时，前端在跳转页面路由时，也可通过返回的菜单/资源列表中查找当前路由是否存在，如果不存在则不允许跳转。
2. 所有请求通过网关进行权限拦截和过滤。/front 前缀的请求只验证用户是否登录，验证 token 是否正常并解析 token 中的用户信息。然后直接转发请求。/boss 前缀的默认是后台管理功能接口，会验证当前请求 url 是否有权限，即查询当前用户的角色关联的资源中有无该 url，如果没有就拦截并返回无权限访问错误。
3. 整体的权限验证流程图：



四、权限管理实现

4.1 权限数据库设计

4.2 权限数据初始化

权限管理数据需要做一次初始化，以便拥有基本的权限管理功能。并且需要分配一个超级管理员角色给一个用户。该用户登录进来后再给其他用户分配权限。后续就可直接在管理后台进行权限相关的增删改查了。

注：初始化的最后一步，是插入一条数据到 user_role_relation 表，给一个用户赋超级管理员角色。

```
1. insert into user_role_relation(user_id, role_id, created_by, updated_by)values
({user_id},'system','system');
```

角色、菜单、资源初始化

```
1. -- 清除表数据
2. -- truncate table roles;
3. -- truncate table menu;
4. -- truncate table resource_category;
5. -- truncate table resource;
6. -- truncate table user_role_relation;
7. -- truncate table role_menu_relation;
8. -- truncate table role_resource_relation;
9.
10. -- 初始化系统管理员角色
11. insert into roles(code,name,description,created_by,updated_by)
12. values
13. ('ADMIN','超级管理员','后台管理员，初始拥有权限管理功能','system','system');
14.
15.
16. -- 初始化菜单
```

```
insert into menu(parent_id,href,icon,name,description,order_num,shown,level,created_by,updated_by)
values
(-1,'','lock','权限管理','管理系统角色、菜单、资源',1,1,0,'system','system'),
(1,'Role','setting','角色列表','管理系统角色',1,1,1,'system','system'),
(1,'Menu','setting','菜单列表','管理系统菜单',2,1,1,'system','system'),
(1,'Resource','setting','资源列表','管理系统资源',3,1,1,'system','system'),
(-1,'Courses','film','课程管理','课程的新增、修改、查看、发布、上下架',2,1,0,'system','system'),
(-1,'Users','user','用户管理','用户的查询、禁用、启用',3,1,0,'system','system'),
(-1,'','setting','广告管理','广告、广告位管理',4,1,0,'system','system'),
(7,'Advertise','setting','广告列表','广告管理',1,1,1,'system','system'),
(7,'AdvertiseSpace','setting','广告位列表','广告位管理',2,1,1,'system','system'),
(1,'AllocMenu','setting','给角色分配菜单页面','给角色分配菜单页面路由',4,0,1,'system','system'),
(1,'AllocResource','setting','给角色分配资源页面','给角色分配资源页面路由',5,0,1,'system','system'),
(1,'AddMenu','setting','添加菜单页面','添加菜单页路由',6,0,1,'system','system'),
(1,'UpdateMenu','setting','更新菜单页面','更新菜单页路由',7,0,1,'system','system'),
(1,'ResourceCategory','setting','资源分类列表页面','资源分类列表页面路由',8,0,1,'system','system'),
(7,'AddAdvertise','setting','添加广告页面','添加广告页面路由',3,0,1,'system','system'),
(7,'UpdateAdvertise','setting','编辑广告页面','编辑广告页面路由',4,0,1,'system','system'),
(7,'AddAdvertiseSpace','setting','添加广告位页面','添加广告位页面路由',5,0,1,'system','system'),
(7,'UpdateAdvertiseSpace','setting','更新广告位页面','更新广告位页面路由',6,0,1,'system','system'),
(5,'CourseItem','setting','课程详情页面','课程详情页面路由',1,0,1,'system','system'),
(5,'CourseSections','setting','课时信息页面','课时信息页面路由',2,0,1,'system','system'),
(5,'VideoOptions','setting','课时上传视频','课时上传视频页面路由',3,0,1,'system','system');


-- 初始化资源分类
insert into resource_category(id,name,sort,created_by,updated_by)
values
(1,'角色管理',1,'system','system'),
(2,'菜单管理',2,'system','system'),
(3,'资源管理',3,'system','system'),
(4,'课程管理',4,'system','system'),
(5,'用户管理',5,'system','system'),
(6,'阿里上传',6,'system','system');


-- 初始化资源
insert into resource(name,url,category_id,description,created_by,updated_by)
values
('获取所有角色','/boss/role/all',1,'获取所有角色','system','system'),
```

('给用户分配角色','/boss/role/allocateUserRoles',1,'给用户分配角色','system','system'),
('按条件查询角色','/boss/role/getRolePages',1,'按条件查询角色','system','system'),
('列出所有角色并标记用户是否拥有','/boss/role/getRolesWithUserPermission',1,'列出所有角色并标记用户是否拥有','system','system'),
('保存或者更新角色','/boss/role/saveOrUpdate',1,'保存或者更新角色','system','system'),
('查询用户角色','/boss/role/user/{userId}',1,'查询用户角色','system','system'),
('获取角色','/boss/role/{id}',1,'获取角色','system','system'),
('删除角色','/boss/role/{id}',1,'删除角色','system','system'),
('给角色分配菜单','/boss/menu/allocateRoleMenus',2,'给角色分配菜单','system','system'),
('获取所有菜单','/boss/menu/getAll',2,'获取所有菜单','system','system'),
('获取编辑菜单页面信息','/boss/menu/getEditMenuInfo',2,'获取编辑菜单页面信息','system','system'),
('获取所有菜单并按层级展示','/boss/menu/getMenuNodeList',2,'获取所有菜单并按层级展示','system','system'),
('按条件分页查询菜单','/boss/menu/getMenuPages',2,'按条件分页查询菜单','system','system'),
('获取角色拥有的菜单列表','/boss/menu/getRoleMenus',2,'获取角色拥有的菜单列表','system','system'),
('保存或新增菜单','/boss/menu/saveOrUpdate',2,'保存或新增菜单','system','system'),
('是否显示开关','/boss/menu/switchShown',2,'是否显示开关','system','system'),
('根据 ID 查询菜单','/boss/menu/{id}',2,'根据 ID 查询菜单','system','system'),
('删除菜单','/boss/menu/{id}',2,'删除菜单','system','system'),
('给角色分配资源','/boss/resource/allocateRoleResources',3,'给角色分配资源','system','system'),
('查询资源分类列表','/boss/resource/category/getAll',3,'查询资源分类列表','system','system'),
('保存或更新资源分类','/boss/resource/category/saveOrderUpdate',3,'保存或更新资源分类','system','system'),
('删除资源分类','/boss/resource/category/{id}',3,'删除资源分类','system','system'),
('获取所有资源','/boss/resource/getAll',3,'获取所有资源','system','system'),
('按条件分页查询资源','/boss/resource/getResourcePages',3,'按条件分页查询资源','system','system'),
('获取角色拥有的资源列表','/boss/resource/getRoleResources',3,'获取角色拥有的资源列表','system','system'),
('保存或者更新资源','/boss/resource/saveOrUpdate',3,'保存或者更新资源','system','system'),
('获取资源','/boss/resource/{id}',3,'获取资源','system','system'),
('删除资源','/boss/resource/{id}',3,'删除资源','system','system'),
('封禁用户','/boss/user/forbidUser',5,'封禁用户','system','system'),
('分页查询用户信息','/boss/user/getUserPages',5,'分页查询用户信息','system','system'),
('获取用户菜单和资源权限列表','/boss/permission/getUserPermissions',5,'获取用户菜单和资源权限列表','system','system'),
('查询用户角色','/boss/role/user/{userId}',1,'查询用户角色','system','system'),
('课程上下架','/boss/course/changeState',4,'课程上下架','system','system'),
('新建课程页面路由','/#/courses/new',4,'新建课程页面路由','system','system'),
('通过课程 Id 获取课程信息','/boss/course/getCourseById',4,'通过课程 Id 获取课程信息','system','system'),

```
('分页查询课程信息','/boss/course/getQueryCourses',4,'分页查询课程信息','system','system'),
('保存或者更新课程信息','/boss/course/saveOrUpdateCourse',4,'保存或者更新课程信息','system','system'),
('上传图片','/boss/course/upload',4,'上传图片','system','system'),
('保存活动商品','/boss/activityCourse/save',4,'保存活动商品','system','system'),
('更新活动商品状态','/boss/activityCourse/updateStatus',4,'更新活动商品状态','system','system'),
('获取章节','/boss/course/section/getBySectionId',4,'获取章节','system','system'),
('获取章节和课时','/boss/course/section/getSectionAndLesson',4,'获取章节和课时','system','system'),
('保存或更新章节','/boss/course/section/saveOrUpdateSection',4,'保存或更新章节','system','system'),
('获取课时内容','/boss/course/lesson/getById',4,'获取课时内容','system','system'),
('保存或更新课时','/boss/course/lesson/saveOrUpdate',4,'保存或更新课时','system','system'),
('获取阿里云图片上传凭证','/boss/course/upload/aliyunImagUploadAddressAdnAuth.json',6,'获取阿里云图片上传凭证','system','system'),
('阿里云转码请求','/boss/course/upload/aliyunTransCode.json',6,'阿里云转码请求','system','system'),
('阿里云转码进度','/boss/course/upload/aliyunTransCodePercent.json',6,'阿里云转码进度','system','system'),
('获取阿里云视频上传凭证','/boss/course/upload/aliyunVideoUploadAddressAdnAuth.json',6,'获取阿里云视频上传凭证','system','system'),
('获取媒体信息','/boss/course/upload/getMediaByLessonId.json',6,'获取媒体信息','system','system'),
('刷新阿里云视频上传凭证','/boss/course/upload/refreshAliyunVideoUploadAddressAdnAuth.json',6,'刷新阿里云视频上传凭证','system','system');

-- 初始化角色-菜单关系
insert into role_menu_relation
(menu_id, role_id, created_by, updated_by)
values
(1,1,'system','system'),
(2,1,'system','system'),
(3,1,'system','system'),
(4,1,'system','system'),
(5,1,'system','system'),
(6,1,'system','system'),
(7,1,'system','system'),
(8,1,'system','system'),
(9,1,'system','system'),
(10,1,'system','system'),
(11,1,'system','system'),
(12,1,'system','system'),
```

```
(13,1,'system','system'),
(14,1,'system','system'),
(15,1,'system','system'),
(16,1,'system','system'),
(17,1,'system','system'),
(18,1,'system','system'),
(19,1,'system','system'),
(20,1,'system','system'),
(21,1,'system','system');


-- 初始化角色-资源关系
insert into role_resource_relation(resource_id, role_id, created_by, updated_by)
values
(1,1,'system','system'),
(2,1,'system','system'),
(3,1,'system','system'),
(4,1,'system','system'),
(5,1,'system','system'),
(6,1,'system','system'),
(7,1,'system','system'),
(8,1,'system','system'),
(9,1,'system','system'),
(10,1,'system','system'),
(11,1,'system','system'),
(12,1,'system','system'),
(13,1,'system','system'),
(14,1,'system','system'),
(15,1,'system','system'),
(16,1,'system','system'),
(17,1,'system','system'),
(18,1,'system','system'),
(19,1,'system','system'),
(20,1,'system','system'),
(21,1,'system','system'),
(22,1,'system','system'),
(23,1,'system','system'),
(24,1,'system','system'),
(25,1,'system','system'),
(26,1,'system','system'),
(27,1,'system','system'),
(28,1,'system','system'),
(29,1,'system','system'),
(30,1,'system','system'),
(31,1,'system','system'),
(32,1,'system','system'),
(33,1,'system','system'),
(34,1,'system','system'),
(35,1,'system','system'),
(36,1,'system','system'),
(37,1,'system','system'),
```

```
(38,1,'system','system'),
(39,1,'system','system'),
(40,1,'system','system'),
(41,1,'system','system'),
(42,1,'system','system'),
(43,1,'system','system'),
(44,1,'system','system'),
(45,1,'system','system'),
(46,1,'system','system'),
(47,1,'system','system'),
(48,1,'system','system'),
(49,1,'system','system'),
(50,1,'system','system'),
(51,1,'system','system');


-- 初始化用户-角色关系
insert into user_role_relation(user_id, role_id, created_by, updated_by)
values
({user_id},1,'system','system');
```

4.3 权限拦截验证实现

1. 后台页面跟后台服务采用前后端分离的架构，前端所有请求都会经过网关转发到后台服务。请参考架构图。
2. edu-boss-boot 项目是管理后台的服务，它主要组合其它服务接口，所以不连接数据库。权限管理的相关接口由 edu-authority-boot 进行封装，提供角色、菜单、资源的增删改查接口，以及给用户分配角色、给角色分配菜单、给角色分配资源的功能接口。
3. 后台请求拦截通过 edu-gateway-boot 网关的权限过滤器实现。代码实现如下：

```
package com.lagou.edu.gateway.filter;

import com.lagou.edu.auth.client.service.IAuthService;
import com.lagou.edu.gateway.service.IPermissionService;
import io.jsonwebtoken.*;
import org.apache.commons.lang.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.gateway.filter.GatewayFilterChain;
import org.springframework.cloud.gateway.filter.GlobalFilter;
import org.springframework.cloud.gateway.support.ServerWebExchangeUtils;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.buffer.DataBuffer;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.server.reactive.ServerHttpRequest;
import org.springframework.web.server.ServerWebExchange;
import reactor.core.publisher.Flux;
```

```
21. import reactor.core.publisher.Mono;
22.
23. import java.net.URI;
24. import java.util.LinkedHashSet;
25.
26. /**
27.  * 请求 url 权限校验
28.  */
29. @Configuration
30. @ComponentScan(basePackages = "com.lagou.edu")
31. public class AccessGatewayFilter implements GlobalFilter {
32.     private final Logger log = LoggerFactory.getLogger(getClass());
33.
34.     private static final String X_USER_ID = "x-user-id";
35.     private static final String X_USER_NAME = "x-user-name";
36.     private static final String X_USER_IP = "x-user-ip";
37.
38.     private static final String BOSS_PATH_PREFIX = "/boss";
39.
40.     /**
41.      * 由 authentication-client 模块提供签权的 feign 客户端
42.      */
43.     @Autowired
44.     private IAuthService authService;
45.
46.     @Autowired
47.     private IPermissionService permissionService;
48.
49.     /**
50.      * 1.首先网关检查 token 是否有效，无效直接返回 401，不调用签权服务
51.      * 2.调用签权服务器看是否对该请求有权限，有权限进入下一个 filter，没有权限返
回 403
52.      * <p>
53.      * 注： 前端会根据 401 去做登录跳转或更新 token。而无权限返回 403
54.      *
55.      * @param exchange
56.      * @param chain
57.      * @return
58.      */
59.     @Override
60.     public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
61.         ServerHttpRequest request = exchange.getRequest();
62.         String url = request.getPath().value();
63.         String authentication = request.getHeaders().getFirst(HttpHeaders.AUTHORIZATION);
64.         log.info("Access filter. url:{}, access_token:{}", url, authentication);
65.         // 有 authentication 的情况下，判断登录状态及是否有权限
66.         if (StringUtils.isNotBlank(authentication)) {
67.             return validateAuthentication(exchange, chain, authentication, url);
68.         }
69.         // 不需要网关签权的 url，直接返回
```

```
70.         if (authService.ignoreAuthentication(url)) {
71.             return chain.filter(exchange);
72.         }
73.         // 没有 authentication 且不是忽略权限验证的 url，则返回 401.
74.         return unauthorized(exchange);
75.     }
76.
77.     /**
78.      * 有 authentication 字段的情况下，验证登录 token。
79.      * 1. 如果解析 token 异常，返回 401，前端重新跳转登录页。
80.      * 2. token 状态正常，判断该 url 是否有权限，如果没有就返回 403，前端根据 403 提
示用户无权限访问接口。
81.      *
82.      * @param exchange
83.      * @param chain
84.      * @param authentication
85.      * @param url
86.      * @return
87.      */
88.     private Mono<Void> validateAuthentication(ServerWebExchange exchange,
GatewayFilterChain chain, String authentication, String url) {
89.         ServerHttpRequest request = exchange.getRequest();
90.         String method = request.getMethodValue();
91.         String ip = request.getRemoteAddress().getAddress().getHostAddress();
92.         // 获取原始的 url。
93.         LinkedHashSet<URI> originUrl =
exchange.getRequiredAttribute(ServerWebExchangeUtils.GATEWAY_ORIGINAL_REQUEST_
URL_ATTR);
94.         // 处理 jwt
95.         String userId = null;
96.         String userName = null;
97.         try {
98.             Jws<Claims> jwt = authService.getJwt(authentication);
99.             if (null != jwt && null != jwt.getBody()) {
100.                 userId = (String) jwt.getBody().get("user_id");
101.                 userName = (String) jwt.getBody().get("user_name");
102.
103.                 // 拼装用户 id、用户名放到请求里面
104.                 ServerHttpRequest.Builder builder = request.mutate();
105.                 if (StringUtils.isNotBlank(userName)) {
106.                     builder.header(X_USER_NAME, userName);
107.                 }
108.                 if (StringUtils.isNotBlank(userId)) {
109.                     builder.header(X_USER_ID, userId);
110.                 }
111.                 if (StringUtils.isNotBlank(ip)) {
112.                     builder.header(X_USER_IP, ip);
113.                 }
114.                 exchange = exchange.mutate().request(builder.build()).build();
115.                 log.info("userId:{}, userName:{}, access_token:{}, url:{}", userId, userName,
```

```
authentication, url);
            }
        } catch (ExpiredJwtException | MalformedJwtException | SignatureException var4) {
            log.error("user token error :{}", var4.getMessage());
            // 如果不是忽略 url，则返回 401，需要登录
            if (!authService.ignoreAuthentication(url)) {
                return unauthorized(exchange);
            }
        }

        // 如果是忽略的 url，在填充 header 中的登录用户信息后直接返回
        if (authService.ignoreAuthentication(url)) {
            return chain.filter(exchange);
        }

        // 管理后台才需要权限，其他时候只判断 jwt 是否正常
        boolean hasPermission = true;
        if (isBossPath(originUrl, url)) {
            // 将原始 url 赋值给当前 url。
            url = BOSS_PATH_PREFIX.concat(url);
            // 调用 edu-authority-boot 服务验证当前用户是否有权限访问该 url
            hasPermission = permissionService.permission(authentication, userId, url, method);
            log.info("Check boss permission. userId:{}, have permission:{}, url:{}, method:{}",
userId, hasPermission, url, method);
        }

        if (hasPermission && StringUtils.isNotBlank(userId) &&
StringUtils.isNotBlank(userName)) {
            log.info("User can access. userId:{}, userName:{}, url:{}, method:{}", userId,
userName, url, method);
            return chain.filter(exchange);
        }
        return forbidden(exchange);
    }

    /**
     * 根据原始 url 判断是否请求的后台管理功能 url。
     * <p>filter 中如果使用了 StripPrefix，url 会被截取前一个"/"节点。无法检测到 url
是否包含/boss。这里获取原始的 url 进行判断</p>
     *
     * @param originUrl 获取的原始 url
     * @param url       通过一些 filter 处理过的 url。
     * @return
     */
    private boolean isBossPath(LinkedHashSet<URI> originUrl, String url) {
        if (url.startsWith(BOSS_PATH_PREFIX)) {
            return true;
        }
        for (URI uri : originUrl) {
```

```
            if (uri.getPath().startsWith(BOSS_PATH_PREFIX)) {
                return true;
            }
        }
        return false;
    }

    /**
     * 未通过权限验证，返回 forbidden
     *
     * @param exchange
     * @return
     */
    private Mono<Void> forbidden(ServerWebExchange exchange) {
        return rebuildExchange(exchange, HttpStatus.FORBIDDEN);
    }

    /**
     * 未登录或 token 状态异常，返回 401
     *
     * @param exchange
     */
    private Mono<Void> unauthorized(ServerWebExchange exchange) {
        return rebuildExchange(exchange, HttpStatus.UNAUTHORIZED);
    }

    private Mono<Void> rebuildExchange(ServerWebExchange exchange, HttpStatus
httpStatus) {
        exchange.getResponse().setStatusCode(httpStatus);
        DataBuffer buffer = exchange.getResponse()
                .bufferFactory().wrap(httpStatus.getReasonPhrase().getBytes());
        return exchange.getResponse().writeWith(Flux.just(buffer));
    }
}
```

4.4 权限数据接口地址

4.4.1 角色管理接口

1. 获取登录用户菜单及可访问资源列表： /boss/permission/getUserPermissions
1. 保存或者更新角色： /boss/role/saveOrUpdate
2. 按条件查询角色： /boss/role/getRolePages
3. 删除角色： /boss/role/{id} DELETE
4. 给用户分配角色：/boss/role/allocateUserRoles

4.4.2 菜单管理接口

1. 保存或新增菜单：/boss/menu/saveOrUpdate

2. 按条件分页查询菜单：/boss/menu/getMenuPages
3. 删除菜单：/boss/menu/{id} DELETE
4. 给角色分配菜单：/boss/menu/allocateRoleMenus
5. 获取角色拥有的菜单列表：/boss/menu/getRoleMenus
6. 获取所有菜单并按层级展示：/boss/menu/getMenuNodeList

4.4.3 资源分类管理接口

1. 查询资源分类列表：/boss/resource/category/getAll
2. 保存或更新资源分类：/boss/resource/category/saveOrderUpdate
3. 删除资源分类：/boss/resource/category/{id} DELETE

4.4.4 资源管理接口

1. 保存或者更新资源：/boss/resource/saveOrUpdate
2. 按条件分页查询资源：/boss/resource/getResourcePages
3. 删除资源：/boss/resource/{id} DELETE
4. 获取角色拥有的资源列表：/boss/resource/getRoleResources
5. 给角色分配资源：/boss/resource/allocateRoleResources