**一、固定角色：**

即把角色与具体的Controller或Action直接关联起来，整个系统中的角色是固定的，每种角色可以访问那些Controller或Action也是固定的，这做法比较适合小型项目，角色分工非常明确的项目。

public void ConfigureServices (IServiceCollection services) {

    services.AddMvc ();

    // 添加认证Cookie信息

    services.AddAuthentication (CookieAuthenticationDefaults.AuthenticationScheme)

        .AddCookie (options => {

            options.LoginPath = new PathString ("/login");

            options.AccessDeniedPath = new PathString ("/denied");

        });

}

public void Configure (IApplicationBuilder app, IHostingEnvironment env) {

    if (env.IsDevelopment ()) {

        app.UseDeveloperExceptionPage ();

        app.UseBrowserLink ();

    } else {

        app.UseExceptionHandler ("/Home/Error");

    }

app.UseStaticFiles ();

    // 认证中间件

app.UseAuthentication ();

    app.UseMvc (routes => {

        routes.MapRoute (

            name: "default",

            template: "{controller=Home}/{action=Index}/{id?}");

    });

}

认证

[Authorize (Roles = "admin,system")]

public class HomeController : Controller {

    ....

}

登录方法

[AllowAnonymous]

[HttpPost ("login")]

public async Task<IActionResult> Login (string userName, string password, string returnUrl = null) {

    var list = new List<dynamic> {

        new { UserName = "gsw", Password = "111111", Role = "admin" },

        new { UserName = "aaa", Password = "222222", Role = "system" }

    };

    var user = list.SingleOrDefault (s => s.UserName == userName && s.Password == password);

    if (user != null) {

        //用户标识

        var identity = new ClaimsIdentity (CookieAuthenticationDefaults.AuthenticationScheme);

        identity.AddClaim (new Claim (ClaimTypes.Sid, userName));

        identity.AddClaim (new Claim (ClaimTypes.Name, user.Name));

        identity.AddClaim (new Claim (ClaimTypes.Role, user.Role));

        // 记录登录信息到cookie

        await HttpContext.SignInAsync (CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal (identity));

        if (returnUrl == null) {

            returnUrl = TempData["returnUrl"]?.ToString ();

        }

        if (returnUrl != null) {

            return Redirect (returnUrl);

        } else {

            return RedirectToAction (nameof (HomeController.Index), "Home");

        }

    }

    else {

        const string badUserNameOrPasswordMessage = "用户名或密码错误！";

        return BadRequest (badUserNameOrPasswordMessage);

    }

}

登出方法

[HttpGet("logout")]

public async Task<IActionResult> Logout()

{

    await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

    return RedirectToAction("Index", "Home");

}

1. **自定义角色**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using PrivilegeManagement.Middleware;

namespace PrivilegeManagement {

    public class Startup {

        public Startup (IConfiguration configuration) {

            Configuration = configuration;

        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices (IServiceCollection services) {

            services.AddAuthentication (CookieAuthenticationDefaults.AuthenticationScheme)

                .AddCookie (options => {

                    options.LoginPath = new PathString ("/login");

                    options.AccessDeniedPath = new PathString ("/denied");

                });

            services.AddMvc ();

        }

        public void Configure (IApplicationBuilder app, IHostingEnvironment env) {

            if (env.IsDevelopment ()) {

                app.UseDeveloperExceptionPage ();

                app.UseBrowserLink ();

            } else {

                app.UseExceptionHandler ("/Home/Error");

            }

            app.UseStaticFiles ();

            //验证中间件

            app.UseAuthentication ();

            ////添加权限中间件, 一定要放在app.UseAuthentication后

            app.UsePermission (new PermissionMiddlewareOption () {

                LoginAction = @"/login",

                    NoPermissionAction = @"/denied",

                    //这个集合从数据库中查出所有用户的全部权限

                    UserPerssions = new List<UserPermission> () {

                        new UserPermission { Url = "/", UserName = "gsw" },

                        new UserPermission { Url = "/home/contact", UserName = "gsw" },

                        new UserPermission { Url = "/home/about", UserName = "aaa" },

                        new UserPermission { Url = "/", UserName = "aaa" }

                    }

            });

            app.UseMvc (routes => {

                routes.MapRoute (

                    name: "default",

                    template: "{controller=Home}/{action=Index}/{id?}");

            });

        }

    }

}

下面看看中间件PermissionMiddleware.cs

using Microsoft.AspNetCore.Http;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Security.Claims;

using System.Threading.Tasks;

namespace PrivilegeManagement.Middleware

{

    /// <summary>

    /// 权限中间件

    /// </summary>

    public class PermissionMiddleware

    {

        /// <summary>

        /// 管道代理对象

        /// </summary>

        private readonly RequestDelegate \_next;

        /// <summary>

        /// 权限中间件的配置选项

        /// </summary>

        private readonly PermissionMiddlewareOption \_option;

        /// <summary>

        /// 用户权限集合

        /// </summary>

        internal static List<UserPermission> \_userPermissions;

        /// <summary>

        /// 权限中间件构造

        /// </summary>

        /// <param name="next">管道代理对象</param>

        /// <param name="permissionResitory">权限仓储对象</param>

        /// <param name="option">权限中间件配置选项</param>

        public PermissionMiddleware(RequestDelegate next, PermissionMiddlewareOption option)

        {

            \_option = option;

            \_next = next;

            \_userPermissions = option.UserPerssions;

        }

        /// <summary>

        /// 调用管道

        /// </summary>

        /// <param name="context">请求上下文</param>

        /// <returns></returns>

        public Task Invoke(HttpContext context)

        {

            //请求Url

            var questUrl = context.Request.Path.Value.ToLower();

            //是否经过验证

            var isAuthenticated = context.User.Identity.IsAuthenticated;

            if (isAuthenticated)

            {

                if (\_userPermissions.GroupBy(g=>g.Url).Where(w => w.Key.ToLower() == questUrl).Count() > 0)

                {

                    //用户名

                    var userName = context.User.Claims.SingleOrDefault(s => s.Type == ClaimTypes.Sid).Value;

                    if (\_userPermissions.Where(w => w.UserName == userName&&w.Url.ToLower()==questUrl).Count() > 0)

                    {

                        return this.\_next(context);

                    }

                    else

                    {

                        //无权限跳转到拒绝页面

                        context.Response.Redirect(\_option.NoPermissionAction);

                    }

                }

            }

            return this.\_next(context);

        }

    }

}

扩展中间件类PermissionMiddlewareExtensions.cs

using Microsoft.AspNetCore.Builder;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace PrivilegeManagement.Middleware

{

    /// <summary>

    /// 扩展权限中间件

    /// </summary>

    public static class PermissionMiddlewareExtensions

    {

        /// <summary>

        /// 引入权限中间件

        /// </summary>

        /// <param name="builder">扩展类型</param>

        /// <param name="option">权限中间件配置选项</param>

        /// <returns></returns>

        public static IApplicationBuilder UsePermission(

              this IApplicationBuilder builder, PermissionMiddlewareOption option)

        {

            return builder.UseMiddleware<PermissionMiddleware>(option);

        }

    }

}

中间件属性PermissionMiddlewareOption.cs