

# 第49期-前端访问控制列表和操作权限控制

2021年12月6日 9:34

## 基于 JwtBearer 身份认证

```
JwtSecurityTokenHandler.DefaultInboundClaimTypeMap.Add(nameof(ClaimTypes.Name).ToLower(), ClaimTypes.Name);

services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(options =>
{
    options.Authority = configuration.GetValue<string>("IdentityServer:AuthorizationUrl");
    options.RequireHttpsMetadata = false;
    options.TokenValidationParameters.ValidateAudience = false;
    options.TokenValidationParameters.NameClaimType = ClaimTypes.Name;
});
```

## 使用前端请求管道传递访问令牌

```
middlewares: [
    async function (ctx, next) {
        const user = await userManager.getUser();
        let token = user?.access_token;
        if (token) {
            const authHeader = { Authorization: `Bearer ${token}` };
            ctx.req.options.headers = { ...ctx.req.options.headers, ...authHeader };
            ctx.req = {
                url: `${ctx.req.url}`,
                options: { ...ctx.req.options },
            };
        }
        await next();
    }
],
```

## 设计前端访问控制列表

```
[Serializable]
public class ApplicationConfiguration
{
    public PermissionConfiguration? Permissions { get; set; }
}
```

```
[Serializable]
public class PermissionConfiguration
{
    public Dictionary<string, bool> Policies { get; set; }
}
```

```

public Dictionary<string, bool> GrantedPolicies { get; set; }

public PermissionConfiguration()
{
    Policies = new Dictionary<string, bool>();
    GrantedPolicies = new Dictionary<string, bool>();
}
}

```

```

[HttpGet]
[AllowAnonymous]
public async Task<ApplicationConfiguration> GetAsync()
{
    _logger.LogDebug("Executing ConfigurationApplicationService.GetAsync()...");

    var result = new ApplicationConfiguration
    {
        Permissions = await GetPermissionConfigurationAsync()
    };

    _logger.LogDebug("Executed ConfigurationApplicationService.GetAsync().");

    return result;
}

```

## 前端加载访问控制列表

```

export default function access(initialState: {
    currentUser?: User | undefined;
    appConfigs?: API.ApplicationConfiguration;
}) {
    const { currentUser, appConfigs } = initialState || {};
    let roles: string[] = currentUser?.profile.role || [];
    let policies = appConfigs?.permissions?.policies;
    let grantedPolicies = appConfigs?.permissions?.grantedPolicies;

    let permissions = {};

    for (const key in policies) {
        if (Object.prototype.hasOwnProperty.call(policies, key)) {
            permissions[key] = Object.prototype.hasOwnProperty.call(grantedPolicies, key) &&
            grantedPolicies?.[key]
        }
    }

    return {
        canTenantManager: roles.indexOf('TenantOwner') > -1,
        canIdentityManager: roles.findIndex(r => r.startsWith('TenantOwner')) > -1,
        ...permissions,
    };
}

```

## 菜单权限控制

```
routes: [
  {
    name: 'product.list',
    path: '/device/product',
    access: 'ProductManager.Products',
    component: './products/list',
  },
],
```

## 操作按钮细粒度可见性权限控制

<https://beta-pro.ant.design/docs/authority-management-cn>

## 基于授权主体生成权限演示数据

```
public class PermissionDataSeedProvider : IDataSeedProvider
```

## 重构并修复多租户数据筛选器

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());

    foreach (IMutableEntityType entityType in
        modelBuilder.Model.GetEntityTypes())
    {
        if (entityType.ClrType.IsAssignableTo(typeof(IMultiTenant)))
        {
            modelBuilder.Entity(entityType.ClrType).AddQueryFilter<IMultiTenant>(e =>
                e.TenantId == this.GetService<ICurrentTenant>().Id);
        }

        if (entityType.ClrType.IsAssignableTo(typeof(ISoftDelete)))
        {
            modelBuilder.Entity(entityType.ClrType).AddQueryFilter<ISoftDelete>(e => !
                e.IsDeleted);
        }
    }

    base.OnModelCreating(modelBuilder);
}
```

## 权限检查相关服务的依赖注入生命周期

```
private static IServiceCollection AddAuthorization(this IServiceCollection
    services)
```

```

{
    services.AddDistributedMemoryCache().AddTransient<IPermissionStore,
PermissionStore>();
    services.AddTransient<IPermissionDefinitionManager,
PermissionDefinitionManager>();

    var exportedTypes = AppDomain.CurrentDomain.GetAssemblies().SelectMany(a
=> a.ExportedTypes).Where(t => t.IsClass);

    var permissionDefinitionProviders = exportedTypes.Where(t =>
t.IsAssignableTo(typeof(IPermissionDefinitionProvider)));
    permissionDefinitionProviders.ToList().ForEach(t =>
services.AddSingleton(typeof(IPermissionDefinitionProvider), t));

    var permissionValueProviders = exportedTypes.Where(t =>
t.IsAssignableTo(typeof(IPermissionValueProvider)));
    permissionValueProviders.ToList().ForEach(t =>
services.AddTransient(typeof(IPermissionValueProvider), t));

    return services;
}

```

## 租户管道中间件执行顺序问题

介于认证和授权中间，认证之后的用户才有租户信息，授权之前也应该具有租户信息。

```

app.UseAuthentication();
app.UseTenantMiddleware();
app.UseAuthorization();

```

















## 查询用户角色设置

```

SELECT u.UserName,r.[Name] FROM [dbo].[AspNetUsers] u
INNER JOIN [AspNetUserRoles] ur ON u.Id=ur.UserId
INNER JOIN [AspNetRoles] r ON r.Id=ur.RoleId order by u.Id,r.Id

```

## 本期课程代码变更内容

- ▲  ZeroStack.DeviceCenter.API
  - ▲  Controllers
    - C# ConfigurationsController.cs
    - C# ProductsController.cs
  - ▲  Extensions\Authorization
    - C# PermissionChecker.cs
  - C# Startup.cs
- ▲  ZeroStack.DeviceCenter.Application
  - ▲  Services\Permissions
    - C# PermissionDataSeedProvider...
    - C# DependencyRegistrar.cs
  - ▲  ZeroStack.DeviceCenter.Infrastruct...
    - C# DeviceCenterDbContext.cs
- ▲  Web\ZeroStack.DeviceCenter.Web\Cli...
  - ▲  config
    - TS routes.ts
  - ▲  src
    - ▲  pages
      - ▲  authorization
        - ▲  login-callback
          - TS index.tsx
        - ▲  logout
          - TS index.tsx
        - ▲  services
          - TS user-service.ts
      - ▲  products
        - TS index.tsx
    - ▲  services\deviceCenter
      - TS Configurations.ts
      - TS index.ts
      - TS typings.d.ts
  - TS access.ts
  - TS app.tsx