

第28期-实现OpenAPI帮助文档授权

2021年4月19日 21:24

固定应用端口

DeviceCenter.API: <https://localhost:6001>; <http://localhost:6000>

IdentityServer.API: <https://localhost:5001>; <http://localhost:5000>

将 OpenAPI 中间件配置到非开发平台

```
app.UseSwagger();  
app.UseSwaggerUI();
```

重新导入演示数据

修改: SampleDatas.cs 和 SampleDataSeed.cs

创建操作过滤器

```
public class SecurityRequirementsOperationFilter : IOperationFilter  
  
services.AddSwaggerGen(c =>  
{  
    c.OperationFilter<SecurityRequirementsOperationFilter>();  
});
```

配置 IdentityServer 跨域 支持

"AllowedOrigins": ["<http://localhost:8000>", "<http://localhost:6001>"]

```
services.AddCors(options =>  
{  
    string[] allowedOrigins =  
configuration.GetSection("AllowedOrigins").Get<string[]>();  
    options.AddDefaultPolicy(builder =>  
builder.WithOrigins(allowedOrigins).AllowAnyMethod().AllowAnyHeader().AllowCredentials());  
});  
  
app.UseCors();
```

配置 OpenAPI 基于 oauth 进行安全认证

```

"IdentityServer": {
    "AuthorizationUrl": "https://localhost:5001"
}

services.AddSwaggerGen(c =>
{
    string identityServer = Configuration.GetValue<string>
("IdentityServer:AuthorizationUrl")

    c.AddSecurityDefinition("oauth2", new OpenApiSecurityScheme
    {
        Type = SecuritySchemeType.OAuth2,
        Flows = new OpenApiOAuthFlows
        {
            AuthorizationCode = new OpenApiOAuthFlow
            {
                AuthorizationUrl = new
Uri($" {identityServer}/connect/authorize"),
                TokenUrl = new
Uri($" {identityServer}/connect/token"),
                Scopes = new Dictionary<string, string>
                {
                    { "openid", "Your user identifier" },
                    { "devicecenter", "Device Center API" }
                }
            }
        }
    });

    c.OperationFilter<SecurityRequirementsOperationFilter>();
});

app.UseSwaggerUI(c =>
{
    c.OAuthClientId("devicecenterswagger");
    c.OAuthClientSecret("secret");
    c.OAuthAppName("Device Center Swagger");
    c.OAuthUsePkce();
});

```

开启基于 JWT 令牌的 Bearer 身份认证方式

Install-package Microsoft.AspNetCore.Authentication.JwtBearer

```

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;
});

app.UseAuthorization();
```

在 API 上使用使用特性注解权限策略

AuthorizeAttribute: 可以使用策略和角色授权

AllowAnonymousAttribute: 允许匿名访问跳过授权

```
services.AddAuthorization(options =>options.AddPolicy("Something", policy =>
policy.RequireClaim("Permission")));
```

使用 `IAuthorizationService` 或 `[Authorize(Policy = "Something")]` 进行授权。