客户端认证为OAuth2的认证模式之一，其针对的场景是客户端不存在或不关系用户，如客户端是定时服务器或者客户端是某个工具软件

**认证服务器配置**

1. 定义API资源域

// 定义需要认证的api资源

public static IEnumerable<ApiResource> GetApiResources()

{

    return new List<ApiResource>

    {

        new ApiResource("api1", "My API")

    };

}

1. 定义客户端

// 想要访问资源的客户端

public static IEnumerable<Client> GetClients() {

    return new List<Client>

    {

        new Client

        {

            ClientId = "client",

            // 没有交互性用户，使用 clientid/secret 实现认证。

            AllowedGrantTypes = GrantTypes.ClientCredentials,

            // 用于认证的密码

            ClientSecrets =

            {

                new Secret("secret".Sha256())

            },

            // 当前客户端有权访问的api域（Scopes）

            AllowedScopes = { "api1" }

        }

    };

}

1. 在.NET中配置

public void ConfigureServices(IServiceCollection services)

{

    services.AddIdentityServer()    // 使用IdentityServer

        .AddDeveloperSigningCredential()    // 生成临时密匙

        .AddInMemoryApiResources(Config.GetApiResources())  // 添加Api资源

        .AddInMemoryClients(Config.GetClients());   // 添加客户端

}

运行此项目，打开浏览器访问http://localhost:5000/.well-known/openid-configuration你将会看到所谓的发现文档。

**资源服务器配置**

1. 添加测试API

添加一个受权限保护的测试API

[Route("identity")]

[Authorize]

public class IdentityController : ControllerBase{

    [HttpGet]

    public IActionResult Get()

    {

        return new JsonResult(from c in User.Claims select new { c.Type, c.Value });

    }

}

1. .NET配置配置

按照包IdentityServer4.AccessTokenValidation

修改Startup

public class Startup{

    public void ConfigureServices(IServiceCollection services)

    {

        ........

        // 添加 Bearer 认证

        services.AddAuthentication("Bearer")

            .AddIdentityServerAuthentication(options =>

            {

                options.Authority = "http://localhost:5000";    // 认证地址

                options.RequireHttpsMetadata = false;

                options.ApiName = "api1";   // 认证域令牌

            });

        .........

    }

    public void Configure(IApplicationBuilder app)

    {

        app.UseAuthentication();    // 使用认证

        app.UseMvc();

    }

}

**客户端配置**

1. 创建客户端

添加 IdentityModel NuGet 程序包到你的客户端项目中。

// 从端口中发现元数据

var disco = await DiscoveryClient.GetAsync("http://localhost:5000");

// 认证客户端

var tokenClient = new TokenClient(disco.TokenEndpoint, "client", "secret");

// 获取api1的访问令牌

var tokenResponse = await tokenClient.RequestClientCredentialsAsync("api1");

if (tokenResponse.IsError)

{

    Console.WriteLine(tokenResponse.Error);

    return;

}

Console.WriteLine(tokenResponse.Json);

// http客户端实例

var client = new HttpClient();

// 设置令牌

client.SetBearerToken(tokenResponse.AccessToken);

// 访问Api

var response = await client.GetAsync("http://localhost:5001/identity");

if (!response.IsSuccessStatusCode)

{

    Console.WriteLine(response.StatusCode);

}

else

{

    var content = await response.Content.ReadAsStringAsync();

    Console.WriteLine(JArray.Parse(content));

}