**OpenID Connect**

IdentityServer已实现了OpenID Connect协议，但需要你实现如下的4个UI视图

登录：/Account/Login

注销：/Account/Logout

同意授权：/Consent

错误：/Error

Demo地址：https://github.com/IdentityServer/IdentityServer4.Demo

**认证服务器配置**

添加OpenID Connect Identity Scopes的支持

与OAuth 2.0类似，OpenID Connect也使用Scopes概念。 Scopes代表您想要保护的客户端希望访问的内容。 OIDC中的Scopes不代表API，而是代表用户ID，姓名或电子邮件地址等身份信息。

Config.cs中添加如下代码：

public static IEnumerable<IdentityResource> GetIdentityResources()

{

    // 身份资源

    return new List<IdentityResource>

    {

        new IdentityResources.OpenId(),

        new IdentityResources.Profile(),

    };

}

// 方便示例用的测试用户

public static List<TestUser> GetUsers()

{

    return new List<TestUser>

    {

        new TestUser

        {

            SubjectId = "1",

            Username = "alice",

            Password = "password",

            Claims = new []

            {

                new Claim("name", "Alice"),

                new Claim("website", "https://alice.com")

            }

        },

        new TestUser

        {

            SubjectId = "2",

            Username = "bob",

            Password = "password",

            Claims = new []

            {

                new Claim("name", "Bob"),

                new Claim("website", "https://bob.com")

            }

        }

    };

}

Startup.cs中的IdentityServer配置中。

public void ConfigureServices(IServiceCollection services){

    services.AddMvc();

    services.AddIdentityServer()

        .AddDeveloperSigningCredential()

        // 添加身份资源

        .AddInMemoryIdentityResources(Config.GetIdentityResources())

        .AddInMemoryApiResources(Config.GetApiResources())

        .AddInMemoryClients(Config.GetClients())

        // 添加测试用户

        .AddTestUsers(Config.GetUsers());

}

为OpenID Connect implicit flow 添加客户端

public static IEnumerable<Client> GetClients(){

    return new List<Client>

    {

        new Client

        {

            ClientId = "mvc",

            ClientName = "MVC Client",

            // 隐式认证

            AllowedGrantTypes = GrantTypes.Implicit,

            // 登录成功后回调地址

            RedirectUris = { "http://localhost:5002/signin-oidc" },

            // 退出登录后回调地址

            PostLogoutRedirectUris = { "http://localhost:5002/signout-callback-oidc" },

            // 客户端可以获取的身份资源

            AllowedScopes = new List<string>

            {

                IdentityServerConstants.StandardScopes.OpenId,

                IdentityServerConstants.StandardScopes.Profile

            }

        }

    };

}

**客户端配置**

创建一个MVC客户端

1. 配置 OpenID Connect 认证

1）添加如下 NuGet 包：

Microsoft.AspNetCore.Authentication.Cookies

Microsoft.AspNetCore.Authentication.OpenIdConnect

2）Startup配置

public void ConfigureServices(IServiceCollection services)

{

    services.AddMvc();

    // 添加认证

    services.AddAuthentication(options =>

        {

            // 默认授权使用Cookies

            options.DefaultScheme = "Cookies";

            // 未登录授权使用oidc

            options.DefaultChallengeScheme = "oidc";

        })

        // 添加名为Cookies的Scheme

        .AddCookie("Cookies")

        // 添加名未oidc的Scheme

        .AddOpenIdConnect("oidc", options =>

        {

            options.SignInScheme = "Cookies";

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

            options.RequireHttpsMetadata = false;

            options.ClientId = "mvc";           // 使用的客户端Id

            options.SaveTokens = true;

        });

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

    app.UseAuthentication();        // 启用认证

    app.UseStaticFiles();

    app.UseMvcWithDefaultRoute();

}

1. 在Action上应用 [Authorize] 特性

我们在HomeController的某个Action上添加特性[Authorize]，用于测试

1. 测试客户端

我们访问受保护的URL（如 /Home/Info），将会被重定向到认证服务器

**添加注销**

最后一步是给MVC客户端添加注销功能。

public async Task Logout()

{

    await HttpContext.SignOutAsync("Cookies");

    await HttpContext.SignOutAsync("oidc");

}

这将清除本地cookie，然后重定向到IdentityServer。 IdentityServer将清除它的cookie，然后给用户一个链接返回到MVC应用程序。