Identity Service

OIDC测试用户： jxw01 000000

安装QuickstartUI

在服务端项目文件夹下，运行 updateUI.ps1 文件里的命令

针对不同场景的配置

> 客户端授权模式

- 资源配置 config.cs

```
        public static IEnumerable<ApiResource> GetApiResources()
        {
            var resources = new List<ApiResource>();

            resources.Add(new ApiResource("mall", "对商城开放的API"));

            return resources;
        }

        public static IEnumerable<Client> GetClients()
        {
            var clients = new List<Client>();

            clients.Add(new Client
            {
                ClientId = "client",

                AllowedGrantTypes = GrantTypes.ClientCredentials,

                ClientSecrets = { new Secret("secret".Sha256()) },

                AllowedScopes = { "mall" }
            });

            return clients;
        }
```

- 配置IServiceCollection

```
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddIdentityServer()
                .AddDeveloperSigningCredential(true, "tempkey.rsa")
                .AddInMemoryApiResources(Config.GetApiResources())
```

```
                .AddInMemoryClients(Config.GetClients());

            services.AddMvc();
        }
```

- 获取token ： http://localhost:5000/connect/token

```
{
    "scope":"mall",
    "client_id":"client",
    "client_secret":"secret",
    "grant_type":"client_credentials"
}
```

- 客户端配置

```
        public void ConfigureServices(IServiceCollection services)
        {

services.AddAuthentication(IdentityServerAuthenticationDefaults.AuthenticationSche
me)
                .AddIdentityServerAuthentication(options =>
                {
                    options.RequireHttpsMetadata = false;
                    options.Authority = "http://localhost:5000";
                    options.ApiName = "mall";
                });
        }
```

> 使用密码保护API资源

- 资源配置 config.cs

```
            public static IEnumerable<ApiResource> GetApiResources()
        {
            var resources = new List<ApiResource>();

            resources.Add(new ApiResource("app", "对app开放的API"));

            return resources;
        }

                public static IEnumerable<Client> GetClients()
        {
            var clients = new List<Client>();

            clients.Add(new Client
```

```
            {
                AccessTokenLifetime = 3600 * 24,

                ClientId = "ro.client",

                AllowedGrantTypes = GrantTypes.ResourceOwnerPassword,

                ClientSecrets = { new Secret("secret".Sha256()) },

                AllowedScopes = { "app" }
            });

            return clients;
        }

                public static IEnumerable<IdentityResource> GetIdentityResources()
        {
            return new List<IdentityResource>
            {
                new IdentityResources.OpenId(),
                new IdentityResources.Profile()
            };
        }
```

自定义允许客户端获取的Claims

```
        public class ProfileService : IProfileService
    {
        public async Task GetProfileDataAsync(ProfileDataRequestContext context)
        {
            await Task.CompletedTask;

            //depending on the scope accessing the user data.
            var claims = context.Subject.Claims;

            //set issued claims to return
            context.IssuedClaims = claims.ToList();
        }

        public async Task IsActiveAsync(IsActiveContext context)
        {
            await Task.CompletedTask;

            context.IsActive = true;
        }
    }

        // 在startup的ConfigureServices中加入:
        // services.AddProfileService<ProfileService>()
```

- 配置验证

```
    public class ResourceOwnerPasswordValidator : IResourceOwnerPasswordValidator
    {
        private readonly ILogger<ResourceOwnerPasswordValidator> _logger;

        public
ResourceOwnerPasswordValidator(ILogger<ResourceOwnerPasswordValidator> logger)
        {
            _logger = logger;
        }

        public async Task ValidateAsync(ResourceOwnerPasswordValidationContext
context)
        {
            await Task.CompletedTask;

            _logger.LogInformation("进入验证");

            //验证用户名密码

            context.Result = new GrantValidationResult("123456",
OidcConstants.AuthenticationMethods.Password, new List<Claim>()
                {
                    new Claim("tenant_id","95")
                });
        }
    }
```

- 配置IServiceCollection

```
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddIdentityServer()
                .AddDeveloperSigningCredential(true, "tempkey.rsa")
                .AddInMemoryApiResources(Config.GetApiResources())
                .AddInMemoryIdentityResources(Config.GetIdentityResources())
                .AddInMemoryClients(Config.GetClients())
                .AddResourceOwnerValidator<ResourceOwnerPasswordValidator>()
                .AddProfileService<ProfileService>();

            services.AddMvc();
        }
```

- 获取token http://localhost:5000/connect/token

```
{
    "scope":"app",
```

```
    "client_id":"ro.client",
    "client_secret":"client_secret",
    "grant_type":"password",
    "username":"username",
    "password":"password"
}
```

> 基于 OpenID Connect 的用户认证

- 添加Quickstart.UI

服务器端根目录下，打开powershell运行以下命令： iex ((New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.com/IdentityServer/IdentityServer4.Quickstart.UI/master/getmaster.ps1'))

- 资源配置 config.cs

```
        public static IEnumerable<Client> GetClients()
        {
            var clients = new List<Client>();

            clients.Add(new Client
            {
                ClientId = "mvc",
                ClientName = "MVC Client",
                AllowedGrantTypes = GrantTypes.Implicit,

                // 登录成功回调处理地址，处理回调返回的数据
                RedirectUris = { "http://localhost:5002/signin-oidc" },

                // where to redirect to after logout
                PostLogoutRedirectUris = { "http://localhost:5002/signout-
callback-oidc" },

                AllowedScopes = new List<string>
                {
                    IdentityServerConstants.StandardScopes.OpenId,
                    IdentityServerConstants.StandardScopes.Profile,
                    "tenantId",
                    "userId",
                    "userName"
                },
                // 关闭授权页面
                RequireConsent = false
            });

            return clients;
        }

                public static IEnumerable<IdentityResource> GetIdentityResources()
        {
            return new List<IdentityResource>
```

```
            {
                new IdentityResources.OpenId(),
                new IdentityResources.Profile(),
                new IdentityResource("tenantId", "tenantId", new List<string>()
{"tenantId"}),
                new IdentityResource("userId", "userId", new List<string>()
{"userId"}),
                new IdentityResource("userName", "userName", new List<string>()
{"userName"})
            };
        }

        public static List<TestUser> GetUsers()
        {
            return new List<TestUser>()
                {
                    new TestUser
                    {
                        SubjectId="1",
                        Username="jxw01",
                        Password="000000",
                        Claims = new List<Claim> {
                            new Claim("tenantId", "95"),
                            new Claim("userName", "jxw01"),
                            new Claim("userId", "95"),
                        }
                    }
                };
        }
```

- 配置IServiceCollection

```
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddIdentityServer(options => {
options.UserInteraction.LoginUrl = "/account/login"; })
                .AddDeveloperSigningCredential(true, "tempkey.rsa")
                .AddInMemoryIdentityResources(Config.GetIdentityResources())
                .AddInMemoryClients(Config.GetClients())
                .AddTestUsers(Config.GetUsers());

            services.AddMvc();
        }
```

- 客户端配置

```
        public void ConfigureServices(IServiceCollection services)
        {
            JwtSecurityTokenHandler.DefaultInboundClaimTypeMap.Clear();
```

```
            services
                .AddAuthentication(options =>
            {
                options.DefaultScheme = "Cookies";
                options.DefaultChallengeScheme = "oidc";
            })
            .AddCookie("Cookies", options=> {
                options.ExpireTimeSpan = TimeSpan.FromMinutes(20);
                options.SlidingExpiration = true;
            })
            .AddOpenIdConnect("oidc", options =>
        {
            // 添加scope，以保证可以从服务器端获取到这些信息
            options.Scope.Add("tenantId");
            options.Scope.Add("userId");
            options.Scope.Add("userName");

            options.Authority = "http://localhost:5000";
            options.RequireHttpsMetadata = false;

            options.ClientId = "mvc";
            options.SaveTokens = true;
        });

            services.AddMvc();
        }
```

For more server information,see http://localhost:5000/.well-known/openid-configuration

使用HttpClient获取Token

```
// IdentityModel 包含了一个用于 发现端点 的客户端库。这样一来你只需要知道
IdentityServer 的基础地址
var disco = await DiscoveryClient.GetAsync("http://localhost:5000");
// 接着你可以使用 TokenClient 类型来请求令牌。为了创建一个该类型的实例，你需要传入令牌端
点地址、客户端id和密码。
// 然后你可以使用 RequestClientCredentialsAsync 方法来为你的目标 API 请求一个令牌：
var tokenClient = new TokenClient(disco.TokenEndpoint, "client", "secret");
var tokenResponse = await tokenClient.RequestClientCredentialsAsync("mall");
// 使用Token访问受保护的资源
var client = new HttpClient();
client.SetBearerToken(tokenResponse.AccessToken);
```