# **OpenID Connect**

OpenID Connect所需的所有协议支持已内置于IdentityServer中。

您需要为登录，注销，同意和错误提供必要的UI部件。

IdentityServer已提供示例项目，包含了必要的UI部件，便于学习可以使用IdentityServer的示例：

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

**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()

{

**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()

.AddInMemoryIdentityResources(Config.GetIdentityResources())

.AddInMemoryApiResources(Config.GetApiResources())

.AddInMemoryClients(Config.GetClients())

.AddTestUsers(Config.GetUsers());

}

# **为OpenID Connect implicit flow 添加客户端**

最后一步是将MVC客户端的新配置条目添加到IdentityServer。

将以下内容添加到客户端配置中：

**public** **static** IEnumerable<Client> GetClients(){

**return** **new** List<Client>

{

**new** Client

{

ClientId = "mvc",

ClientName = "MVC Client",

AllowedGrantTypes = GrantTypes.Implicit,

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

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

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**

为了能向 MVC 应用程序添加 OpenID Connect 认证支持，请添加如下 NuGet 程序包：

* Microsoft.AspNetCore.Authentication.Cookies
* Microsoft.AspNetCore.Authentication.OpenIdConnect

Startup的 ConfigureServices方法中添加以下代码：

**public** **void** ConfigureServices(IServiceCollection services)

{

services.AddMvc();

// 添加认证

services.AddAuthentication(options =>

{

options.DefaultScheme = "Cookies";

options.DefaultChallengeScheme = "oidc";

})

.AddCookie("Cookies")

.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**;

});

}

Startup中的Configure方法添加UseAuthentication：

**public** **void** Configure(IApplicationBuilder app, IHostingEnvironment env)

{

app.UseAuthentication(); // 启用认证

app.UseStaticFiles();

app.UseMvcWithDefaultRoute();

}

验证中间件应该在MVC之前添加。

最后一步是触发认证。为了进入HomeController，并在其中一个Action上添加特性[Authorize]

# **测试客户端**

通过访问受保护的Action来触发身份验证握手。 你应该看到重定向到IdentityServer的登录页面。

成功登录后，用户将看到同意画面。

最后浏览器重定向到客户端应用程序，该应用程序显示了用户的声明。

# **添加注销**

最后一步是给MVC客户端添加注销功能。

只需将以下代码添加到某个控制器即可触发注销：

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。