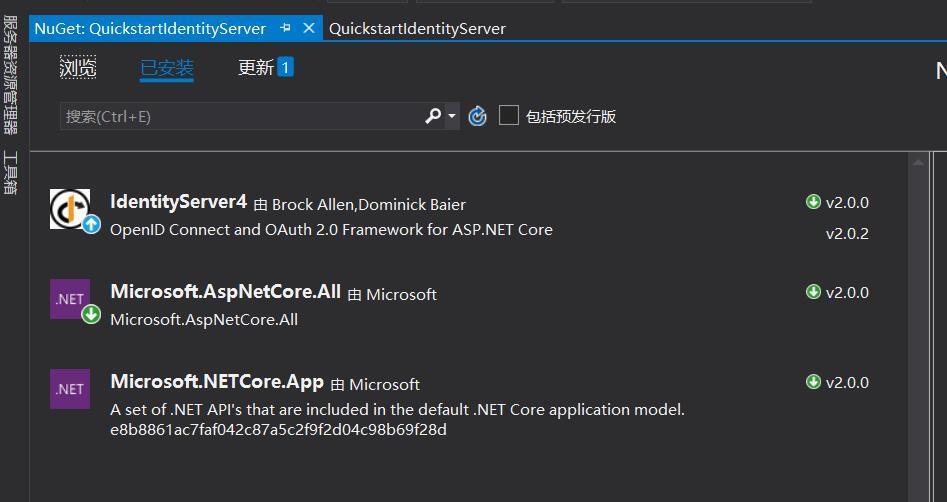
### 创建IdentityServer4 服务

创建一个名为QuickstartIdentityServer的ASP.NET Core Web 空项目（asp.net core 2.0），端口5000

NuGet包：



修改Startup.cs 设置使用IdentityServer：

public class Startup

{

public void ConfigureServices(IServiceCollection services)

{

// configure identity server with in-memory stores, keys, clients and scopes

services.AddIdentityServer()

.AddDeveloperSigningCredential()

.AddInMemoryIdentityResources(Config.GetIdentityResourceResources())

.AddInMemoryApiResources(Config.GetApiResources())

.AddInMemoryClients(Config.GetClients())

.AddResourceOwnerValidator<ResourceOwnerPasswordValidator>()

.AddProfileService<ProfileService>();

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseIdentityServer();

}

}

添加Config.cs配置IdentityResource，ApiResource以及Client：

public class Config

{

public static IEnumerable<IdentityResource> GetIdentityResourceResources()

{

return new List<IdentityResource>

{

new IdentityResources.OpenId(), //必须要添加，否则报无效的scope错误

new IdentityResources.Profile()

};

}

// scopes define the API resources in your system

public static IEnumerable<ApiResource> GetApiResources()

{

return new List<ApiResource>

{

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

};

}

// clients want to access resources (aka scopes)

public static IEnumerable<Client> GetClients()

{

// client credentials client

return new List<Client>

{

new Client

{

ClientId = "client1",

AllowedGrantTypes = GrantTypes.ClientCredentials,

ClientSecrets =

{

new Secret("secret".Sha256())

},

AllowedScopes = { "api1",IdentityServerConstants.StandardScopes.OpenId, //必须要添加，否则报forbidden错误

IdentityServerConstants.StandardScopes.Profile},

},

// resource owner password grant client

new Client

{

ClientId = "client2",

AllowedGrantTypes = GrantTypes.ResourceOwnerPassword,

ClientSecrets =

{

new Secret("secret".Sha256())

},

AllowedScopes = { "api1",IdentityServerConstants.StandardScopes.OpenId, //必须要添加，否则报forbidden错误

IdentityServerConstants.StandardScopes.Profile }

}

};

}

}

因为要使用登录的时候要使用数据中保存的用户进行验证，要实IResourceOwnerPasswordValidator接口：

public class ResourceOwnerPasswordValidator : IResourceOwnerPasswordValidator

{

public ResourceOwnerPasswordValidator()

{

}

public async Task ValidateAsync(ResourceOwnerPasswordValidationContext context)

{

//根据context.UserName和context.Password与数据库的数据做校验，判断是否合法

if (context.UserName=="wjk"&&context.Password=="123")

{

context.Result = new GrantValidationResult(

subject: context.UserName,

authenticationMethod: "custom",

claims: GetUserClaims());

}

else

{

//验证失败

context.Result = new GrantValidationResult(TokenRequestErrors.InvalidGrant, "invalid custom credential");

}

}

//可以根据需要设置相应的Claim

private Claim[] GetUserClaims()

{

return new Claim[]

{

new Claim("UserId", 1.ToString()),

new Claim(JwtClaimTypes.Name,"wjk"),

new Claim(JwtClaimTypes.GivenName, "jaycewu"),

new Claim(JwtClaimTypes.FamilyName, "yyy"),

new Claim(JwtClaimTypes.Email, "977865769@qq.com"),

new Claim(JwtClaimTypes.Role,"admin")

};

}

}

IdentityServer提供了接口访问用户信息，但是默认返回的数据只有sub，就是上面设置的subject: context.UserName，要返回更多的信息，需要实现IProfileService接口：

public class ProfileService : IProfileService

{

public async Task GetProfileDataAsync(ProfileDataRequestContext context)

{

try

{

//depending on the scope accessing the user data.

var claims = context.Subject.Claims.ToList();

//set issued claims to return

context.IssuedClaims = claims.ToList();

}

catch (Exception ex)

{

//log your error

}

}

public async Task IsActiveAsync(IsActiveContext context)

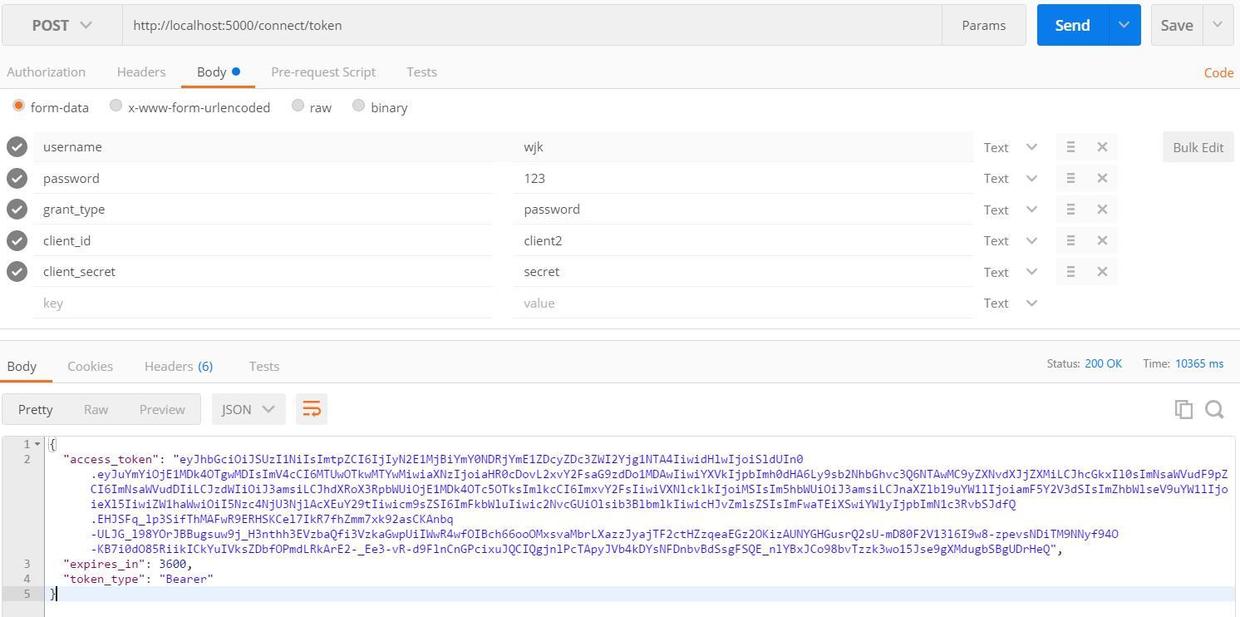
{

context.IsActive = true;

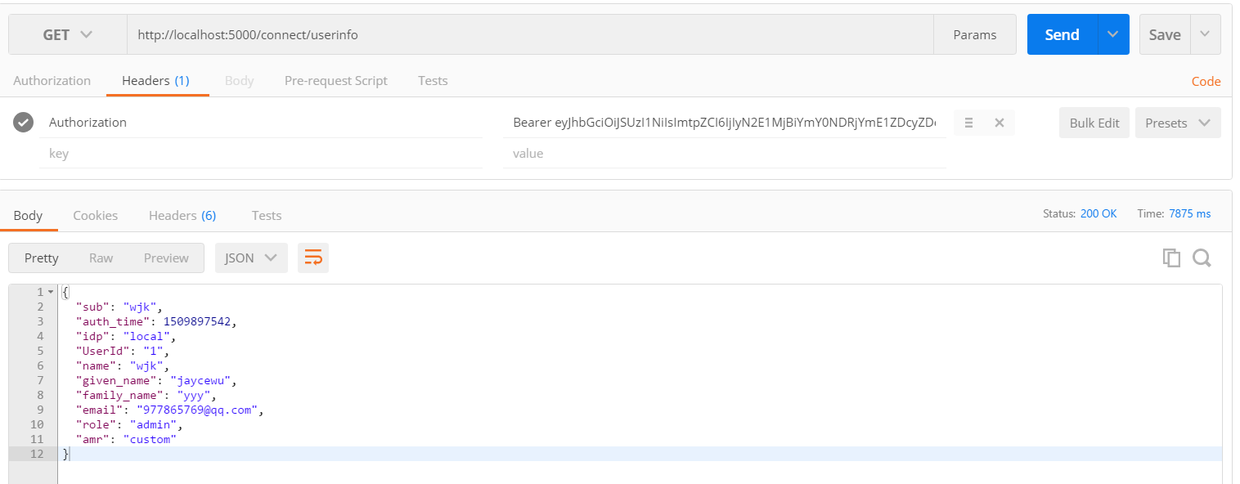
}

context.Subject.Claims就是之前实现IResourceOwnerPasswordValidator接口时claims: GetUserClaims()给到的数据。  
另外，经过调试发现，显示执行ResourceOwnerPasswordValidator 里的ValidateAsync，然后执行ProfileService 里的IsActiveAsync，GetProfileDataAsync。

启动项目，使用postman进行请求就可以获取到token：



再用token获取相应的用户信息：



token认证服务一般是与web程序分开的，上面创建的QuickstartIdentityServer项目就相当于服务端，我们需要写业务逻辑的web程序就相当于客户端。当用户请求web程序的时候，web程序拿着用户已经登录取得的token去IdentityServer服务端校验。