###ClientCredential

Oath2.0对于其的定义：
    ClientCredential授权方式用于应用程序请求access token去访问他们的资源，并不是代表用户。因此他们无法访问授权服务器里面的用户信息。
角色

    被保护的资源Api 资源服务器

    授权服务器(资源:api,resource)

    第三方Client，需要使用

###授权服务器

使用ids4empty模板
    dotnet new -i IdentityServer4.Templates 下载ids模板到文件夹

    dotnet new is4empty -n ServerName 生成模板 生成空模板是将信息写在内存中

    在Config文件中配置 和 授权服务器中的资源 受保护的资源 注册的客户端
         //授权服务器里面存储的资源 例如用户信息 openId 手机号码 地址
    public static IEnumerable<IdentityResource> IdentityResources =>
        new IdentityResource[]
        {
            new IdentityResources.OpenId()
        };
    //注册在授权服务器 受保护的资源
    public static IEnumerable<ApiScope> ApiScopes =>
        new ApiScope[]
        {
            new ApiScope("api1","My API")
        };
    //注册在授权服务其中的客户 例如一些应用程序
    public static IEnumerable<Client> Clients =>
        new Client[]
        {
            new Client
            {
                ClientId ="client",

                AllowedGrantTypes = GrantTypes.ClientCredentials,

                // 用于身份验证的密钥
                ClientSecrets =
                {
                    new Secret("secret".Sha256())

```
                },

                // 客户端有权访问的范围
                AllowedScopes = { "api1" }
            }

        };
```

###被保护的资源Api 命名空间引用 using Microsoft.AspNetCore.Authentication.JwtBearer; using Microsoft.IdentityModel.Tokens;

```
安装Microsoft.AspNetCore.Authentication.JwtBearer包

在startup配置
builder.Services.AddAuthentication("Bearer")
        .AddJwtBearer("Bearer", options =>//JWT bearer authentication performs
authentication by extracting and validating a JWT token from the Authorization
request header
        {
            options.Authority = "https://localhost:5001";

            options.TokenValidationParameters = new TokenValidationParameters
            {//包含一组参数，这些参数由 SecurityTokenHandler 在验证 SecurityToken 时
使用。
                ValidateAudience = false
            };
        });
添加app.Authentication()认证中间件

在控制器中添加一个控制器获取用户声明类型和值
[Route("[controller]")]
[Authorize]//using Microsoft.AspNetCore.Authorization;
[ApiController]
public class **IdentityController** : ControllerBase
[HttpGet]
    public IActionResult Get()
    {
        return new JsonResult(from c in User.Claims select new { c.Type, c.Value
});
    }
}
```

###注册的客户端 Client 创建一个控制台应用程序 dotnet add package IdentityModel using IdentityModel.Client;

```
通过HttpClient的GetDiscoveryDocumentAsync("授权服务器地址") 获取到一个response
var client = new HttpClient();
```

```
var disco = await client.GetDiscoveryDocumentAsync("https://localhost:5001");
if(disco.IsError)
{
    System.Console.WriteLine(disco.Error);
    return;
}

获取Token=>Token里面有用户基本信息允许暴露出来的信息

RequestClientCredentialsTokenAsync()因为用的是ClientCredential的方式
在ClientCredentialsTokenRequest的构造函数中填入 发现文档的Token端点,注册在授权服务器里面的ClientId,SecretId,Scoped所允许的资源范围
// 请求令牌
var tokenResponse = await client.RequestClientCredentialsTokenAsync(new
ClientCredentialsTokenRequest
{
    Address = disco.TokenEndpoint,

    ClientId = "client",
    ClientSecret = "secret",
    Scope = "api1"
});
if (tokenResponse.IsError)
{
    Console.WriteLine(tokenResponse.Error);
    return;
}

Console.WriteLine(tokenResponse.Json);
```

###Client调用Api

```
var apiClient = new HttpClient();
apiClient.SetBearerToken(tokenRespnse.AccessToken);//这里是请求的token中的
accessToken 这里面的内容就是JWT的内容

var response = await apiClient.GetAsync("https://localhost:6001/identity");//API地
址
if(!response.IsSuccessStatusCode)
{
    Console.WriteLine(response.StateCode);
}
else
{
    var json =  await response.Context.ReadAsStringAsync();
    Console.WriteLine(Array.Parse(json));//Newtonsoft nuget  using
Newtonsoft.Json.Linq
}
```