Dùng Module Tsp.Volo.Abp.OpenIddict.Pro để tạo token đăng nhập

Last updated by | Quang Hợp Đặng | Aug 1, 2024 at 4:05 PM GMT+7

1. Cài package

- Cài các package sau vào Project Unified:

```
<PackageReference Include="Tsp.Volo.Abp.Account.Pro.Public.Web.OpenIddict" Version="1.0.10" />
    <PackageReference Include="Tsp.Volo.Abp.OpenIddict.Pro.Application" Version="1.0.2" />
    <PackageReference Include="Tsp.Volo.Abp.OpenIddict.Pro.EntityFrameworkCore" Version="1.0.2" />
    <PackageReference Include="Tsp.Volo.Abp.OpenIddict.Pro.HttpApi" Version="1.0.2" />
    <PackageReference Include="Tsp.Volo.Abp.OpenIddict.Pro.Web" Version="1.0.2" />
```

- Trong DependOn dùng khai báo như sau:

```
typeof(AbpOpenIddictAspNetCoreModule),
    typeof(AbpOpenIddictProHttpApiModule),
    typeof(AbpOpenIddictProEntityFrameworkCoreModule),
    typeof(AbpOpenIddictProApplicationModule),
    typeof(AbpOpenIddictProWebModule),
```

2 Config trong PreConfigureServices

- Trong PreConfigureServices sẽ thêm như sau:

```
 public override void PreConfigureServices(ServiceConfigurationContext context)
    {
        PreConfigure<OpenIddictBuilder>(builder =>
        {
            builder.AddValidation(options =>
            {
                options.AddAudiences("MeetingsManagement");
                options.UseLocalServer();
                options.UseAspNetCore();
            });
        });
    }
```

3 Config trong ConfigureServices

- Trong ConfigureServices sẽ thêm như sau:

```
context.Services.ForwardIdentityAuthenticationForBearer(OpenIddictValidationAspNetCoreDefaults.AuthenticationScheme);
```

4. OnApplicationInitializationAsync

- Đảm bảo thứ tự `app.UseAuthentication();` , xong đến `app.UseAbpOpenIddictValidation();` , xong đến `app.UseAuthorization();`

5. Sửa application.json

- Có dạng như sau:

```
{
    "ConnectionStrings": {
        // "Default": "Server=localhost;Port=3306;Database=ModuleMeeting;User=root;Password=123456;"
        "Default": "server=.;database=ModuleMeeting;trusted_connection=true;Encrypt=False"
    },
    "AuthServer": {
        "Authority": "https://localhost:44391",
        "RequireHttpsMetadata": "false"
    }
}
```

6. Config db

- Trong **UnifiedDbContext** tại `OnModelCreating` thêm `modelBuilder.ConfigureOpenIddict()` kiểu như sau:

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.ConfigurePermissionManagement();
        modelBuilder.ConfigureSettingManagement();
        modelBuilder.ConfigureAuditLogging();
        modelBuilder.ConfigureIdentity();
        modelBuilder.ConfigureFeatureManagement();
        modelBuilder.ConfigureTenantManagement();
        modelBuilder.ConfigureOpenIddict();
        modelBuilder.ConfigureMeetingsManagement();
    }
```

Note: Tạo migration và update database (nếu cần)

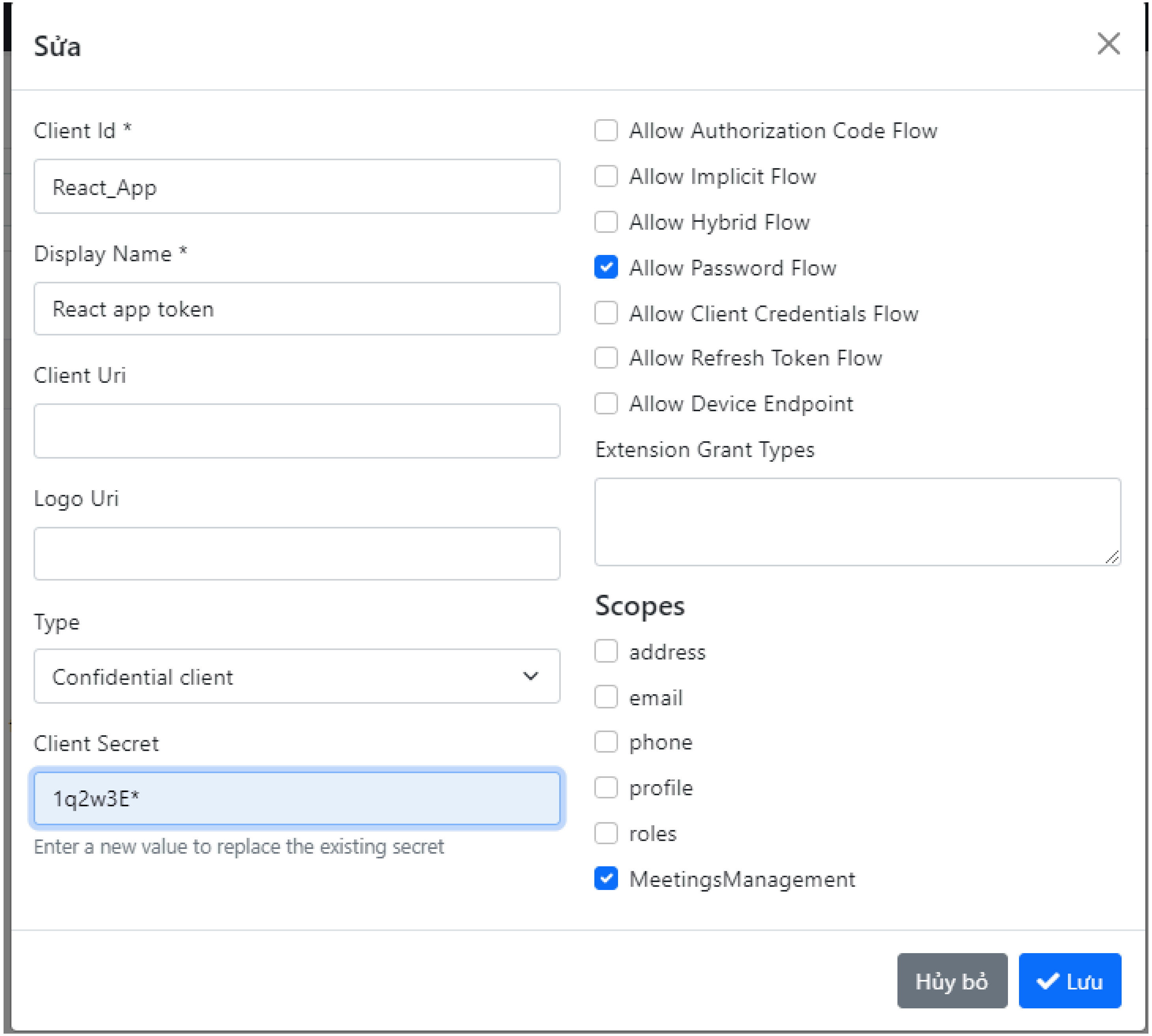
- Thêm OpenIddictDataSeedContributor để seed scope custom:

```
namespace Module.MeetingsManagement.OpenIddict;
public class OpenIddictDataSeedContributor : IDataSeedContributor, ITransientDependency
{
    private readonly IOpenIddictScopeManager _scopeManager;
    private readonly IStringLocalizer<OpenIddictResponse> L;

    public OpenIddictDataSeedContributor(
        IOpenIddictScopeManager scopeManager,
        IStringLocalizer<OpenIddictResponse> l)
    {
        _scopeManager = scopeManager;
        L = l;
    }

    [UnitOfWork]
    public virtual async Task SeedAsync(DataSeedContext context)
    {
        await CreateScopesAsync();
    }

    private async Task CreateScopesAsync()
    {
        if (await _scopeManager.FindByNameAsync("MeetingsManagement") == null)
        {
            await _scopeManager.CreateAsync(new OpenIddictScopeDescriptor
            {
                Name = "MeetingsManagement",
                DisplayName = "MeetingsManagement API",
                Resources =
                {
                    "MeetingsManagement"
                }
            });
            }
        }

}
```

Note: Có thể tham khảo thêm phần seed data của AuthServer

7. Tạo mới client_token

- vào trang `/openiddict/applications` thêm mới client như sau:



Sửa

Client Id *

React_App

Display Name *

React app token

Client Uri

Logo Uri

Type

Confidential client

Client Secret

1q2w3E*

Enter a new value to replace the existing secret

□ Allow Authorization Code Flow

□ Allow Implicit Flow

□ Allow Hybrid Flow

☑ Allow Password Flow

□ Allow Client Credentials Flow

□ Allow Refresh Token Flow

□ Allow Device Endpoint

Extension Grant Types

Scopes

□ address

□ email

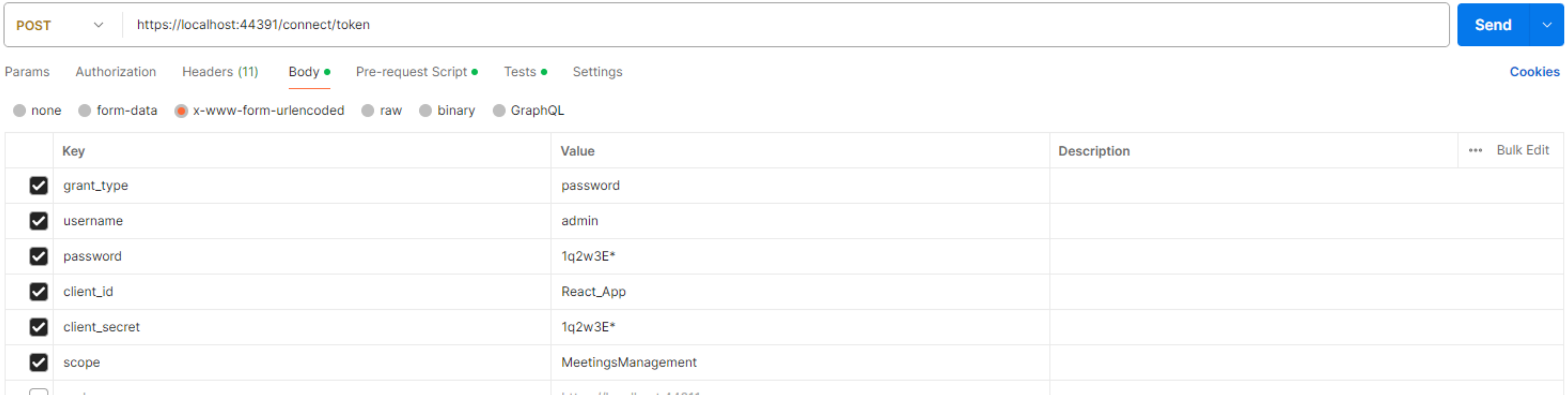
□ phone

□ profile

□ roles

☑ MeetingsManagement

- **Lấy token bằng api** `/connect/token`



POST https://localhost:44391/connect/token

○ none ○ form-data ● x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

	Key	Value	Description
☑	grant_type	password	
☑	username	admin	
☑	password	1q2w3E*	
☑	client_id	React_App	
☑	client_secret	1q2w3E*	
☑	scope	MeetingsManagement	

8. Cấu hình server với nginx

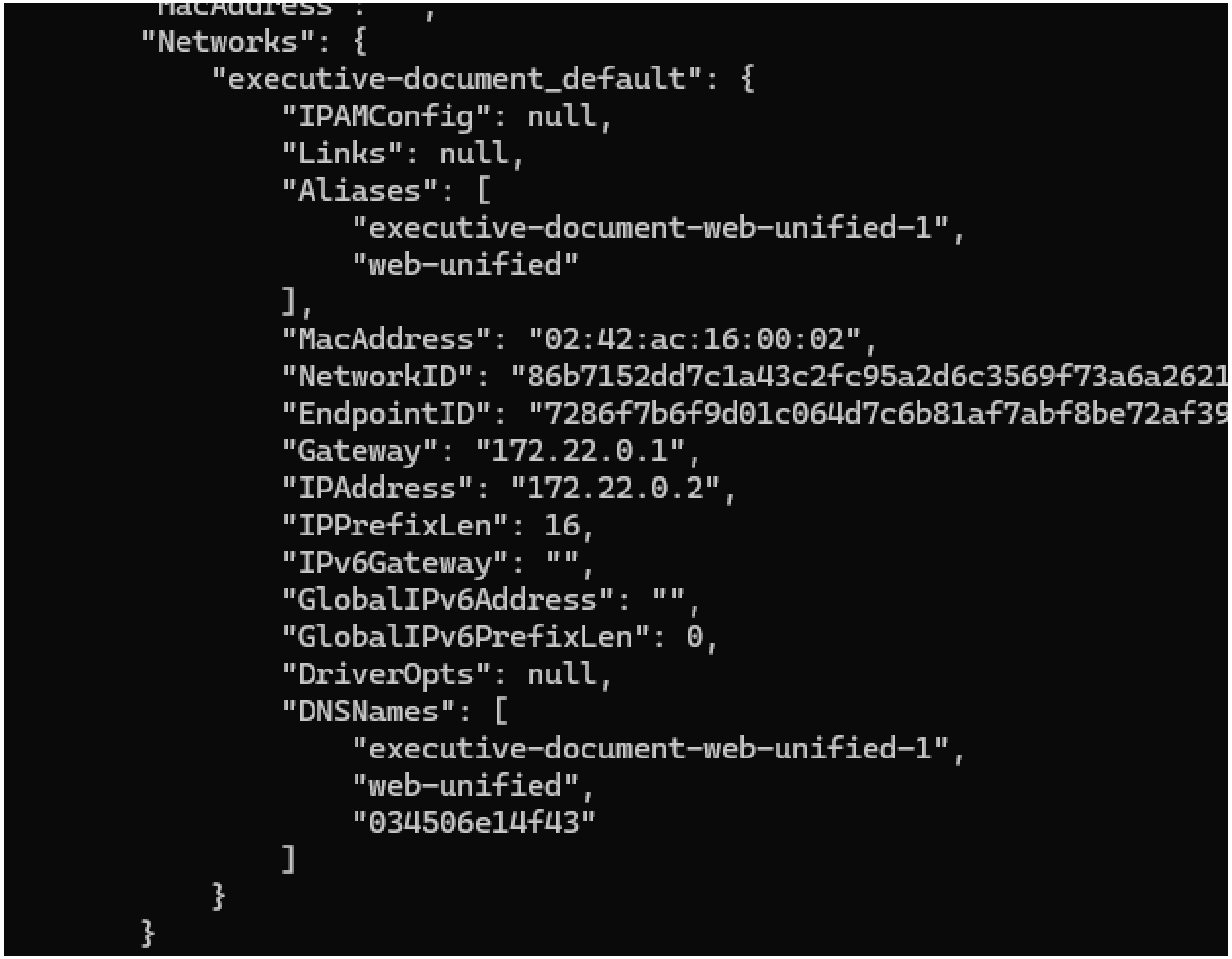
1. Thêm code dưới đây vào module của web app

```
private void ConfigureProxy(IConfiguration configuration)
{
    Configure<ForwardedHeadersOptions>(options =>
    {
        var knownProxies = configuration.GetValue<string>("ForwardedHeadersOptions:KnownProxies");
        var knownNetworks = configuration.GetValue<string>("ForwardedHeadersOptions:KnownNetworks");
        options.ForwardedHeaders =
            ForwardedHeaders.XForwardedFor | ForwardedHeaders.XForwardedProto;
        if (!string.IsNullOrEmpty(knownProxies))
        {
            options.KnownProxies.Clear();
            foreach (var proxy in knownProxies.Split(","))
            {
                options.KnownProxies.Add(IPAddress.Parse(proxy));
            }
        }
        if (!string.IsNullOrEmpty(knownNetworks))
        {
            options.KnownNetworks.Clear();
            foreach (var network in knownNetworks.Split(","))
            {
                options.KnownNetworks.Add(new Microsoft.AspNetCore.HttpOverrides.IPNetwork(IPAddress.Parse(network), 24));
            }
        }
    });
}
```

2. Thêm vào environment của file docker compose cho container của web app

- ForwardedHeadersOptions__KnownProxies=[ip của gateway]
- ForwardedHeadersOptions__KnownNetworks=

Để lấy được gateway ip từ docker sử dụng lệnh sau:
docker container inspect [container id]



```
        "MacAddress": "",
        "Networks": {
            "executive-document_default": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": [
                    "executive-document-web-unified-1",
                    "web-unified"
                ],
                "MacAddress": "02:42:ac:16:00:02",
                "NetworkID": "86b7152dd7c1a43c2fc95a2d6c3569f73a6a2621
                "EndpointID": "7286f7b6f9d01c064d7c6b81af7abf8be72af39
                "Gateway": "172.22.0.1",
                "IPAddress": "172.22.0.2",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "DriverOpts": null,
                "DNSNames": [
                    "executive-document-web-unified-1",
                    "web-unified",
                    "034506e14f43"
                ]
            }
        }
```

IP của gateway ở Networks -> Gateway

3. Thêm cấu hình sau vào file nginx config:

`underscores_in_headers on;` trong block `server`

`proxy_pass_request_headers on;` trong block `server -> location`

4. Khi get token cần thêm header

`__tenant` với giá trị tenant id của user đang muốn lấy token