Sau khi viết Service thì ta qua controller

Jwt > Odata > dependency injection > authorization

Tuy nhiên phải setup trong program.cs trước

Các bước như sau:

Bước 1: thêm cái này dưới swagger gen (Authentication)

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = configuration["JWT:Issuer"],

ValidAudience = configuration["JWT:Audience"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(configuration["JWT:SecretKey"]))

};

});

>>>>>>>>> sau đó copy cái đoạn trong DBcontext bỏ lên trên

IConfiguration configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json", true, true).Build();

>>>>>>>>>>>>>>> tiếp đến bỏ đoạn code tạo secretkey vào appsettings

"Jwt": {

"SecretKey": "UYDEV\_SECRETKEY\_SIEUCAPBAOMAT\_VAYMACHUADU16KYTU",

"Issuer": "UYDEV\_Issuer",

"Audience": "UYDEV\_Audience" }

Tạo xong rồi bây giờ tới USE

app.UseAuthentication();

Thêm giao diện ở swagger để bỏ cái token vào thay cái addSwagger thành như sau

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "FU\_NewsManagement",

Version = "v1"

});

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme()

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "JWT Authorization header using the Bearer scheme",

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement {

{

new OpenApiSecurityScheme {

Reference = new OpenApiReference {

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

Bước 2 : set up odata

Trong code bên dưới đã thêm code tránh vòng lập vô hạn khi ánh xạ qua lại giữa 2 bảng số ít và nhiều

var modelBuilder = new ODataConventionModelBuilder();

modelBuilder.EntitySet<FootballPlayer>("FootballPlayers");

builder.Services.AddControllers()

.AddJsonOptions(options =>

{

options.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.IgnoreCycles;

options.JsonSerializerOptions.DefaultIgnoreCondition = JsonIgnoreCondition.Never;

})

.AddOData(

options => options.Select().Filter().OrderBy().Expand().Count().SetMaxTop(null).AddRouteComponents(

"odata",

modelBuilder.GetEdmModel()));

ở bên dưới thay cái endpoint.mapcontroller thành

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

Và thêm

app.UseRouting();

A screen shot of a computer program

Description automatically generated

Bước 3 : khai báo dependency injection

1 cái repo, 1 cái service

A screenshot of a computer program

Description automatically generated

builder.Services.AddScoped<IFootballPlayerRepo, FootballPlayerRepo>();

builder.Services.AddScoped<IPremierLeagueAccountRepo, PremierLeagueAccountRepo>();

builder.Services.AddScoped<IFootballPlayerService, FootballPlayerService>();

builder.Services.AddScoped<IPremierLeagueAccountService, PremierLeagueAccountService>();

Bước 4: Authorization phân quyền

builder.Services.AddAuthorization(options =>

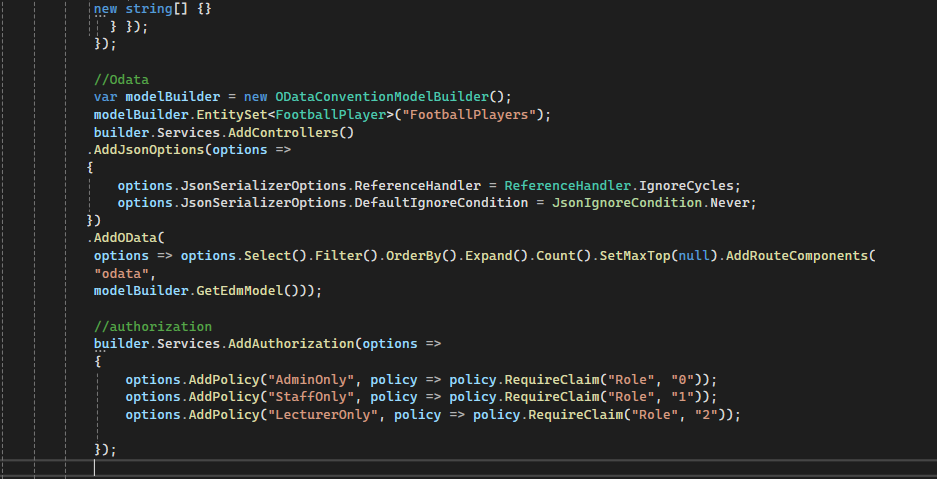
{

options.AddPolicy("AdminOnly", policy => policy.RequireClaim("Role", "0"));

options.AddPolicy("StaffOnly", policy => policy.RequireClaim("Role", "1"));

options.AddPolicy("LecturerOnly", policy => policy.RequireClaim("Role", "2"));

});



Cấu hình hoàn chỉnh file program

using BOs;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.OData;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OData.ModelBuilder;

using Microsoft.OpenApi.Models;

using Repos;

using Services;

using System.Text;

using System.Text.Json.Serialization;

namespace PE\_PRN231\_FA24\_TrialTest\_TranGiaHuy\_OdataAPI

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Configuration

IConfiguration configuration = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json", true, true).Build();

// OData Configuration

var modelBuilder = new ODataConventionModelBuilder();

modelBuilder.EntitySet<FootballPlayer>("FootballPlayers");

builder.Services.AddControllers()

.AddJsonOptions(options =>

{

options.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.IgnoreCycles;

options.JsonSerializerOptions.DefaultIgnoreCondition = JsonIgnoreCondition.Never;

})

.AddOData(options => options.Select().Filter().OrderBy().Expand().Count().SetMaxTop(null)

.AddRouteComponents("odata", modelBuilder.GetEdmModel()));

// Dependency Injection

builder.Services.AddScoped<IFootballPlayerRepo, FootballPlayerRepo>();

builder.Services.AddScoped<IPremierLeagueAccountRepo, PremierLeagueAccountRepo>();

builder.Services.AddScoped<IFootballPlayerService, FootballPlayerService>();

builder.Services.AddScoped<IPremierLeagueAccountService, PremierLeagueAccountService>();

// JWT Authentication

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = configuration["JWT:Issuer"],

ValidAudience = configuration["JWT:Audience"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(configuration["JWT:SecretKey"]))

};

});

// Swagger Configuration with JWT

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme()

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "JWT Authorization header using the Bearer scheme",

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement {

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" }

},

Array.Empty<string>()

}});

});

// Authorization Policies

builder.Services.AddAuthorization(options =>

{

options.AddPolicy("AdminOnly", policy => policy.RequireClaim("Role", "0"));

options.AddPolicy("StaffOnly", policy => policy.RequireClaim("Role", "1"));

options.AddPolicy("LecturerOnly", policy => policy.RequireClaim("Role", "2"));

});

var app = builder.Build();

// Configure the HTTP request pipeline

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

app.Run();

}

}

}