WEB CORE APP PAGINATION

LAUNCHSETTINGS.JSON

{

"$schema": "http://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:48379",

"sslPort": 0

}

},

"profiles": {

"http": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "http://localhost:5098",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "swagger",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

APPSETTINGS.JSON

{

"ConnectionStrings": {

"TaxiServiceDBCon": "Server=pc-05\\SQL2019;Database=TaxiServiceDB;User Id=sa;Password=sa;TrustServerCertificate=True;"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

TAXIRIDESCONTROLLER.CS

using System.Data;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Data.SqlClient;

namespace webapi.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class TaxiRidesController : ControllerBase

{

private readonly IConfiguration \_configuration;

public TaxiRidesController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpGet]

[Route("GetTaxiRides2")]

public JsonResult GetTaxiRides()

{

string query = "select \* from dbo.TaxiRides2";

DataTable table = new DataTable();

// Ensure the connection string is retrieved properly

string sqlDataSource = \_configuration.GetConnectionString("TaxiServiceDBCon");

if (string.IsNullOrEmpty(sqlDataSource))

{

throw new InvalidOperationException("Connection string 'TaxiServiceDBCon' has not been initialized.");

}

using SqlConnection TaxiServiceDBCon = new SqlConnection(sqlDataSource);

TaxiServiceDBCon.Open();

using var myCommand = new SqlCommand(query, TaxiServiceDBCon);

using var myReader = myCommand.ExecuteReader();

table.Load(myReader);

return new JsonResult(table);

}

}

}

PROGRAM.CS

using Newtonsoft.Json.Serialization;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

// Enable controllers

builder.Services.AddControllers()

// JSON serialization settings to avoid reference loop issues

.AddNewtonsoftJson(options =>

{

options.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;

options.SerializerSettings.ContractResolver = new DefaultContractResolver();

});

// Add Swagger support

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// Add CORS policy for the Blazor Server App (adjust the port if needed)

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowBlazorApp", policy =>

{

policy.WithOrigins("http://localhost:5204") // change this if your Blazor app runs on a different port

.AllowAnyHeader()

.AllowAnyMethod();

});

});

// Optional: if you plan to use IConfiguration directly elsewhere

builder.Services.AddSingleton<IConfiguration>(builder.Configuration);

var app = builder.Build();

// Enable CORS

app.UseCors("AllowBlazorApp");

// Swagger in development

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

// Authorization middleware (not mandatory unless you use [Authorize])

app.UseAuthorization();

// Map your controllers

app.MapControllers();

// Start the app

app.Run();