LOAD JSON FILE INTO STARTUP CLASS

And

Custom StartUp class

Models for jsonfile:

namespace RSAACEPOSApi.Models

{

public class ConfigSettings

{

public string ConfigFileName { get; set; }

public string RetailerId { get; set; }

public string RetailerName { get; set; }

public string RemoveCheckDigitFromBasketUPC { get; set; }

public string SharedKey { get; set; }

public string SecretKey { get; set; }

public string MemberNumberLength { get; set; }

public string ClientBucketName { get; set; }

public string ProcessCoupons { get; set; }

public string ProcessPromotions { get; set; }

public string EnablePromotionsForMembersOnly { get; set; }

public string EnablePromotionsForAll { get; set; }

public string CostPlusStoreIds { get; set; }

public string MaxCouponValueForClient { get; set; }

public string PromotionsAPIUrl { get; set; }

public string RSACoreAPIUrl { get; set; }

public string SystemNotificationsARN { get; set; }

public string EnableRemoveNCRImpression { get; set; }

public string EnterpriseId { get; set; }

public string EnterpriseSecret { get; set; }

public string BaseUrl { get; set; }

public string UPCCouponTypeIds { get; set; }

public string MFRUPCCouponTypeIds { get; set; }

public string SetCouponMultiLineIdsToZero { get; set; }

public string CrossCellCouponsPurchaseUPCsOrderByAmount { get; set; }

public string EnableWriteLogs { get; set; }

public string EnableEmailNotificationAlerts { get; set; }

public string MustBuyPromotionTitle { get; set; }

public string GroupDiscountPromotionOverrideType { get; set; }

}

public class ConfigsValues

{

public List<ConfigSettings> ConfigSettings { get; set; }

}

}

StartUp.cs:

using Microsoft.OpenApi.Models;

using System.Net;

using System.Text;

using RSAACEPOSApi.BAL;

using RSAACEPOSApi.DAL;

using RSAACEPOSApi.Models;

using Microsoft.Extensions.Configuration;

using Microsoft.AspNetCore.Hosting;

using System.Collections.Generic;

using Microsoft.Extensions.Options;

using System.Text.Json;

namespace RSAACEPOSApi

{

public class StartUp

{

private readonly IWebHostEnvironment \_env;

public IConfiguration configRoot { get; }

public StartUp(IConfiguration configuration, IWebHostEnvironment env)

{

configRoot = configuration;

\_env = env;

}

public void ConfigureServices(IServiceCollection services)

{

// Read json data from Different Location in project

// ConfigJsonFiles

//var basepath = Path.Combine(Environment.CurrentDirectory, @"ConfigJsonFiles\");

//IConfiguration config = new ConfigurationBuilder()

//.SetBasePath(basepath)

//.AddJsonFile("ConfigSettings.json")

//.Build();

// inside wwwroot folder

//IConfiguration config = new ConfigurationBuilder()

//.SetBasePath(\_env.WebRootPath)

//.AddJsonFile("ConfigSettings.json")

//.Build();

// inside a main project RSAACEPOSApi

IConfiguration config = new ConfigurationBuilder()

.SetBasePath(\_env.ContentRootPath) //.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("ConfigSettings.json")

.Build();

services.AddMvc();

services.AddControllers();

services.AddSwaggerGen();

services.AddControllers().AddXmlSerializerFormatters();

services.AddControllers().AddXmlDataContractSerializerFormatters();

services.AddScoped<RSAACEPOSApiBAL>();

services.AddScoped<RSAACEPOSApiDALcs>();

// load jsonfile

// configure AWS Lambda service

services.AddAWSLambdaHosting(LambdaEventSource.RestApi);

// configure file

services.AddOptions();

services.Configure<ConfigsValues>(config);

services.Configure<Configs>(configRoot.GetSection("Configs"));

services.Configure<List<ConfigSettings>>(configRoot.GetSection("ConfigSettings"));

}

public void Configure(WebApplication app, IWebHostEnvironment env)

{

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Error");

app.UseStatusCodePagesWithReExecute("/Error/{0}");

}

app.UseCors(c => c.AllowAnyHeader().AllowAnyOrigin().AllowAnyMethod());

app.Use(async (context, next) =>

{

try

{

await next(context);

}

catch (Exception ex)

{

context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;

if (ex.InnerException != null)

{

await context.Response.WriteAsync(ex.Message);

}

}

});

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}

Program.cs:

using Microsoft.AspNetCore.Hosting;

using RSAACEPOSApi;

var builder = WebApplication.CreateBuilder(args);

// add builder environment

var startup = new StartUp(builder.Configuration,builder.Environment);

startup.ConfigureServices(builder.Services);

var app = builder.Build();

startup.Configure(app, builder.Environment);

//var builder = WebApplication.CreateBuilder(args);

//// Add services to the container.

//builder.Services.AddControllers();

//// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

//builder.Services.AddEndpointsApiExplorer();

//builder.Services.AddSwaggerGen();

//var app = builder.Build();

//// Configure the HTTP request pipeline.

//if (app.Environment.IsDevelopment())

//{

// app.UseSwagger();

// app.UseSwaggerUI();

//}

//app.UseHttpsRedirection();

//app.UseAuthorization();

//app.MapControllers();

//app.Run();

Controller.cs:

public ConfigsValues ConfigsValues { get; set; }

private readonly IOptions<Configs> \_options;

private readonly IOptions<List<ConfigSettings>> \_config;

private readonly IOptions<ConfigsValues> \_configValues;

public RSAACEPOSApiDALcs(

IOptions<Configs> options,

IOptions<List<ConfigSettings>> config,

IOptions<ConfigsValues> configValues

)

{

\_options = options;

\_config = config;

\_configValues = configValues;

}

[HttpGet]

public List<ConfigSettings> GetConfigs()

{

try {

var data = \_configValues.Value.ConfigSettings.ToArray();

if(data != null)

{

var configSettings = JsonConvert.SerializeObject(data);

var jsonData = JsonConvert.DeserializeObject<List<ConfigSettings>>(configSettings);

return jsonData;

}

else

{

return null;

}

}

catch (Exception ex)

{

throw;

}

}

Upload and Read JSON Files

[HttpGet]

[Route("ReadJsonFile")]

public IActionResult ReadJsonFile(int id)

{

//var uploadpath = \_webHostEnvironment.WebRootPath;

//var destinationPath = Path.Combine(uploadpath, "D:\\csharpprojects\\StudentThreeTier\\StudentThreeTier\\Content\\");

//if (!Directory.Exists(destinationPath))

//{

// Directory.CreateDirectory(destinationPath);

//}

//var sourcePath = Path.GetFileName(formFile.Name);

//var fullpath = Path.Combine(destinationPath, sourcePath);

//using(FileStream stream = new FileStream(fullpath, FileMode.Create))

//{

// formFile.CopyTo(stream);

//}

var json = System.IO.File.ReadAllText("D:\\csharpprojects\\StudentThreeTier\\StudentThreeTier\\Content\\Uploads\\client.json");

// var json = System.IO.File.ReadAllText(fullpath);

var jsonData = JsonConvert.DeserializeObject<List<RSAClient>>(json);

var data = jsonData.FirstOrDefault(j => j.RSAClientId == id);

if (data == null)

{

return NotFound($" id {id} is not found.");

}

else

{

var clientModel = new RSAClient

{

RSAClientId = data.RSAClientId,

RSAClientName = data.RSAClientName,

Stores = data.Stores,

};

return Ok(clientModel);

}

}

[HttpPost]

[Route(" UplaodAndReadJsonFile")]

public IActionResult UplaodAndReadJsonFile(IFormFile formFile,int id)

{

var uploadpath = \_webHostEnvironment.WebRootPath;

var destinationPath = Path.Combine(uploadpath, "JSONFILES");

if (!Directory.Exists(destinationPath))

{

Directory.CreateDirectory(destinationPath);

}

var sourcePath = Path.GetFileName(formFile.FileName);

var fullpath = Path.Combine(destinationPath, sourcePath);

using (FileStream stream = new FileStream(fullpath, FileMode.Create))

{

formFile.CopyTo(stream);

}

Console.WriteLine(fullpath);

// read json file

var json = System.IO.File.ReadAllText(fullpath);

var jsonData = JsonConvert.DeserializeObject<List<RSAClient>>(json);

var data = jsonData?.FirstOrDefault(item => item.RSAClientId == id);

if(data == null)

{

return NotFound($" Id {id} is invalid");

}

// return Ok(jsonData);

RSAClient clientModel = new RSAClient();

clientModel.RSAClientId = data.RSAClientId;

clientModel.RSAClientName = data.RSAClientName;

clientModel.Stores = data.Stores;

return Ok(clientModel);

}

[HttpGet]

[Route("ReadJsonFromFolder")]

public IActionResult ReadJsonFromFolder(int Id)

{

// var basepath = Path.Combine(Environment.CurrentDirectory, @"XMLFiles\");

// var basePath = Path.Combine(\_webHostEnvironment.ContentRootPath, @"JSONFiles\client.json");

// Dynamic read the folder any where in project with filename

// string fileName = @"client.json";

string fileName = @"RSAClient.json";

string currentDirectory = Directory.GetCurrentDirectory();

// get full path with folders names dynamically

string[] fullFilePath = Directory.GetFiles(currentDirectory,fileName, SearchOption.AllDirectories);

var jsonData = System.IO.File.ReadAllText(fullFilePath[0]);

var dataobj = JsonConvert.DeserializeObject<List<RSAClient>>(jsonData);

var data = dataobj?.FirstOrDefault(d => d.RSAClientId == Id);

if(data == null)

{

return NotFound("Invalid id ");

}

return Ok(data);

}

WOKING WITH DAL

Create DAL.cs:

public class Dal

{

public Dal()

{

}

// get name

public string Name(string name)

{

return name;

}

}

Controller.cs:

[HttpGet]

[Route("GetName")]

public IActionResult GetName(string name)

{

Dal dal = new Dal();

var result = dal.Name(name); return Ok(result);

}

DAL-BAL-CONTROLLER

Dal.cs:

namespace RSAACEPOSApi.DAL

{

public class Dal

{

public Dal()

{

}

// get name

public string Name(string name)

{

return name;

}

}

}

Bal.cs:

private readonly Dal \_dal;

public RSAACEPOSApiBAL(

Dal dal

)

{

\_dal = dal;

}

public string Name(string name)

{

var result = \_dal.Name(name);

return result;

}

}

Controller:

private readonly RSAACEPOSApiBAL \_rSAACEPOSApiBAL;

public RSAACEPOSApi(

RSAACEPOSApiBAL rSAACEPOSApiBAL

)

{

\_rSAACEPOSApiBAL = rSAACEPOSApiBAL;

}

[HttpGet]

[Route("GetName")]

public IActionResult GetName(string name)

{

Dal dal = new Dal();

// var result = dal.Name(name); return Ok(result);

var result = \_rSAACEPOSApiBAL.Name(name);

return Ok(result);

}

Program.cs:

services.AddScoped<RSAACEPOSApiBAL>();

services.AddScoped<RSAACEPOSApiDALcs>();

services.AddScoped<Dal>();