Log4net

Step1:

Install dependcies for project: log4net

Program.cs:

using log4net;

using log4net.Config;

namespace RSAMobileAPI

{

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)

{

// add this logRepository

var logRepository = LogManager.GetRepository(Assembly.GetEntryAssembly());

XmlConfigurator.Configure(logRepository, new FileInfo("log4net.config"));

services.AddMvc();

services.AddControllers();

services.AddSwaggerGen();

services.AddControllers().AddXmlSerializerFormatters();

services.AddControllers().AddXmlDataContractSerializerFormatters();

services.AddEndpointsApiExplorer();

// appsettings

services.AddOptions();

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

services.AddScoped<IAuthManager, AuthManager>();

services.AddScoped<IEncryptMgr, EncryptMgr>();

services.AddScoped<ICustomerMgr, CustomerMgr>();

services.AddScoped<ICommonMgr, CommonMgr>();

services.AddScoped<IServicesMgr, ServicesMgr>();

services.AddScoped<ICouponsMgr,CouponsMgr>();

services.AddScoped<IBarcodeMgr,BarcodeMgr>();

services.AddScoped<IStoreMgr, StoreMgr>();

services.AddScoped<ConstantMgr>();

services.AddScoped<IUserMgr, UserMgr>();

services.AddScoped<ImageMgr, ImageMgr>();

services.AddScoped<IAmazonMgr, AmazonMgr>();

services.AddScoped<UsersDA>();

services.AddScoped<ServicesDA>();

services.AddScoped<CustomerDA>();

services.AddScoped<CommonDA>();

services.AddScoped<CouponsDA>();

services.AddScoped<StoreDA>();

services.AddScoped<ConfigurationHelper>();

services.AddScoped<MultipleResultSets>();

services.AddScoped<Helpers>();

services.AddScoped<NCRHelper>();

services.AddScoped<APIMethods>();

services.AddScoped<RSAServices.AWSMailHelper>();

services.AddDbContext<RSA\_QAEntities>(options => options.UseSqlServer(configRoot.GetConnectionString("RSADBCon")));

// provide identityuser

// For Identity

//services.AddIdentity<IdentityUser, IdentityRole>()

// .AddEntityFrameworkStores<RSA\_QAEntities>()

// .AddDefaultTokenProviders();

services.AddIdentity<ApplicationUser, IdentityRole>

(

options => {

options.Password.RequiredUniqueChars = 0;

options.Password.RequireUppercase = false;

options.Password.RequireLowercase = false;

options.Password.RequireNonAlphanumeric = false;

//options.Password.RequiredLength = 8;

}

).AddEntityFrameworkStores<RSA\_QAEntities>().AddDefaultTokenProviders();

// Add Logging builder

services.AddLogging(builder =>

{

builder.AddConsole();

});

}

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();

}

}

}

RSAMobileAPI.csproj:

<ItemGroup>  
<None Update="log4net.config">

<CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>

</None>

</ItemGroup>

Application.json:

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

}

Log4net.config:

<?xml version="1.0" encoding="utf-8" ?>

<log4net>

<root>

<level value="DEBUG" />

<appender-ref ref="ConsoleAppender" />

<appender-ref ref="FileAppender" />

</root>

<appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%level %logger - %message%newline" />

</layout>

</appender>

<appender name="FileAppender" type="log4net.Appender.RollingFileAppender">

<file value="C:\Users\Veritra\RSAWebServices\Logs\log.txt" />

<appendToFile value="true" />

<rollingStyle value="Size" />

<maxSizeRollBackups value="10" />

<maximumFileSize value="10MB" />

<staticLogFileName value="true" />

<layout type="log4net.Layout.PatternLayout">

//<conversionPattern value="%date %-5level - %message%newline" />

// this one includes datetime and log info message like.

2024-05-30 14:20:37,000 INFO [1] - Error occured in the ChangeUserPhoneNumber method. Error details : Please enter a valid 10 phone number.

// <conversionPattern value="%level %logger - %message%newline" />

//This one includes info message. Like

INFO RSAMobileAPI.Controllers.RSAMobileAPI - 5/30/2024 2:20:37 PM - Error occured in the ChangeUserPhoneNumber method. Error details : Please enter a valid 10 phone number.

<conversionPattern value="%message%newline" /> // currently using this

// This one includes only log message. Like

5/30/2024 2:30:02 PM - Error occured in the ChangeUserPhoneNumber method. Error details : Please enter a valid 10 phone number.

</layout>

</appender>

</log4net>

<!-- C:\Users\Veritra\RSAWebServices\Logs\log.txt --> // local folder for logs message

RSAMobileAPIController.cs:

namespace RSAMobileAPI.Controllers

{

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

[ApiController]

public class RSAMobileAPI : ControllerBase

{

private readonly ILogger<RSAMobileAPI> \_logger;

private readonly ILog log;

// Log4Net

// private readonly ILog log = LogManager.GetLogger(typeof(RSAMobileAPI));

public RSAMobileAPI(

// log4net

log = LogManager.GetLogger(typeof(RSAMobileAPI));

)

public IActionResult Index()

{

log.Info("ChangeUserPhoneNumber method start");

// Simulate an error

log.Info("Error occurred in the ChangeUserPhoneNumber method. Error details: Please enter a valid 10-digit phone number.");

return View();

}

}