**Swagger with .NET Core**

1. Add the SwashBuckle.AspNetCore package

It provides 3 packages

1. Swashbucke.AspNetCore.Swagger
2. Swashbuckle.AspNetCore.SwaggerGen
3. Swashbuckle.AspNetCore.SwaggerUI
4. Add the following lines of code in ConfigureServices method

services.AddSwaggerGen(options =>

{

//Customize the Swagger Doc

options.SwaggerDoc("v1", new Info {

Title = "SynBlog API",

Version = "v1",

});

});

1. Add the Middleware for Swagger in Configure method

app.UseSwagger();

app.UseSwaggerUI(options =>

{

options.SwaggerEndpoint("/swagger/v1/swagger.json", "SynBlog API");

options.RoutePrefix = string.Empty; // to serve swagger UI in root url

});

**Customize the Swagger Doc**

1. Add the following code in Project file (csproj)

<PropertyGroup>

<GenerateDocumentationFile>true</GenerateDocumentationFile>

<NoWarn>$(NoWarn);1591</NoWarn>

</PropertyGroup>

1. Update the program.cs file and add the pragma statements

#pragma warning disable CS1591

public class Program

{

public static void Main(string[] args)

{

CreateWebHostBuilder(args).Build().Run();

}

public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>

WebHost.CreateDefaultBuilder(args)

.UseStartup<Startup>();

}

#pragma warning restore CS1591

1. Add the documentation comment in the API method. Add the <responses > element with status codes and description

/// <summary>

/// Adds a new blog to the blogs collection

/// </summary>

/// <remarks>

/// Sample request:

/// POST /api/blogs

/// {

/// "title":"Blog Title",

/// "Content":"Blog content",

/// "AddedBy":"Author email"

/// "AddedDate":"Posted Date"

/// }

/// </remarks>

/// <param name="blog"></param>

/// <returns>A newly created blog item</returns>

/// <response code="201">Returns a newly created blog item</response>

/// <response code="400">If the item is null or invalid</response>

[ProducesResponseType(201)]

[ProducesResponseType(400)]

[HttpPost("",Name ="AddBlog")]

public async Task<ActionResult<Blog>> AddBlog(Blog blog)

{

if (blog == null)

{

return BadRequest();

}

var item=await repo.AddAsync(blog);

return item;

}

1. Update the swagger service configuration in ConfigureServices method.

services.AddSwaggerGen(options =>

{

//Customize the Swagger Doc

options.SwaggerDoc("v1", new Info

{

Title = "SynBlog API",

Version = "v1",

Description = "Synergetics Blogs API",

TermsOfService = "None",

Contact = new Contact()

{

Name = "Sonu Sathyadas",

Email = "sonusathyadas@hotmail.com",

Url = "https://streamingskills.blog/"

},

License = new License

{

Name = "MIT License info",

Url = "https://streamingskills.blog/licence"

}

});

var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";

var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);

options.IncludeXmlComments(xmlPath);

});

**Customizing the Swagger UI**

1. Open the Startup class and add the StaticFiles middleware.

app.UseStaticFiles();

1. Acquire the contents of the dist folder from the [Swagger UI GitHub repository](https://github.com/swagger-api/swagger-ui/tree/master/dist). This folder contains the necessary assets for the Swagger UI page.

<https://github.com/swagger-api/swagger-ui/tree/master/dist>

1. Create a **wwwroot/swagger/ui** folder, and copy into it the contents of the dist folder.
2. Create a **custom.css** file, in **wwwroot/swagger/ui,** with the following CSS to customize the page header:

.swagger-ui .topbar {

background-color: #000;

border-bottom: 3px solid #547f00;

}

1. Reference custom.css in the index.html file, after any other CSS files
2. Browse to the index.html page at http://localhost:<port>/swagger/ui/index.html. Enter http://localhost:<port>/swagger/v1/swagger.json in the header's textbox, and click the **Explore** button.