Documenting an ASP.NET Core Web API
Using Swagger

Mohamed ELshafei

Help Doc Using Swagger

Swagger (OpenAPI) is a language-agnostic specification for describing REST APIs.

It allows both computers and humans to understand the capabilities of a REST API without direct access to the source code. Its main goals are to:

- Minimize the amount of work needed to connect decoupled services.
- Reduce the amount of time needed to accurately document a service.

The two main OpenAPI implementations for .NET are Swashbuckle and NSwag

Help Doc Using NSwag

- **NSwag**, third-party APIs that incorporate Swagger and generate a client implementation.
- **NSwag** allows you to expedite the development cycle and easily adapt to API changes.

```
Install-Package NSwag.AspNetCore
```

- **Add and configure Swagger middleware**
- register the required Swagger services

```
services.AddSwaggerDocument();
```

- enable the middleware for Swagger and Swagger UI

```
app.UseOpenApi();
app.UseSwaggerUi3();
```

http://localhost:<port>/swagger

Help Doc Using Swashbuckle

- **Swagger** tooling for APIs built with ASP.NET Core. Generate beautiful API documentation, including a UI to ***explore*** and ***test*** operations, directly from your routes, controllers and models.
- Install the standard Nuget package into your ASP.NET Core application.

```
Install-Package Swashbuckle.AspNetCore
```

- **Add and configure Swagger middleware**
- register the required Swagger services
- enable the middleware for Swagger and Swagger UI

```
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
```

```
app.UseSwagger();
app.UseSwaggerUI();
```

http://localhost:<port>/swagger

Provide Global API Metadata

- you can provide a full description for your API, terms of service or even contact and licensing information:

```
services.AddSwaggerGen(c =>
{
   c.SwaggerDoc("v1",
      new OpenApiInfo
      {
         Title = "My API - V1",
         Version = "v1“,
         Description = "A sample API to demo Swashbuckle",
         TermsOfService = new Uri("http://tempuri.org/terms"),
          Contact = new OpenApiContact
            {
                Name = “MElShafie",
                Email = " MElShafie @tempuri.org"
             },
       }
    );
});
```

Enrich OpenAPI documentation with XML comments

- To enhance the generated docs with human-friendly descriptions, you can annotate controller actions and models with Xml Comments and configure Swashbuckle to incorporate those comments into the outputted Swagger JSON
- Open the Properties dialog for your project, click the "Build" tab and ensure that "XML documentation file" is checked.
- Configure Swashbuckle to incorporate the XML comments on file into the generated Swagger JSON:

```
services.AddSwaggerGen(c =>
{
    var filePath = Path.Combine(System.AppContext.BaseDirectory, "MyApi.xml");
    c.IncludeXmlComments(filePath);
});
```

Enrich OpenAPI documentation with annotations

- Install the following Nuget package into your ASP.NET Core application.

```
Install-Package Swashbuckle.AspNetCore.Annotations
```

- In the ConfigureServices method of Startup.cs, enable annotations within in the Swagger config block:

```
services.AddSwaggerGen(c =>
{
  ...

  c.EnableAnnotations();
});
```

Enrich Operation Metadata

- Once annotations have been enabled, you can enrich the generated Operation metadata by decorating actions with a *SwaggerOperationAttribute*.

```
[HttpPost]
[SwaggerOperation(
    Summary = "Creates a new product",
    Description = "Requires admin privileges",
    OperationId = "CreateProduct",
    Tags = new[] { "Purchase", "Products" }
)]
 public IActionResult Create([FromBody]Product product)
```

Enrich Response Metadata

- ASP.NET Core provides the ProducesResponseTypeAttribute for listing the different responses that can be returned by an action. to include human friendly descriptions with each response in the generated Swagger

```
[HttpPost]
[SwaggerResponse(201, "The product was created", typeof(Product))]
[SwaggerResponse(400, "The product data is invalid")]
public IActionResult Create([FromBody]Product product)
```

Enrich Response Metadata

- Identify which content-types your API handles on requests and responses. The following attributes specify that the API should only use the application/json content type in both directions

```
[HttpPost]
[Produces("application/json")]
[Consumes("application/json")]
public IActionResult Create([FromBody]Product product)
```

Securing Swagger UI endpoints

- Call MapSwagger().RequireAuthorization to secure the Swagger UI endpoints. The following example secures the swagger endpoints in middleware :

```
app.MapSwagger().RequireAuthorization();
```