Today, in the software industry, sharing data is very crucial. The best way to connect and share data is through APIs. APIs are supposed to connect the developers from various platforms and share the data through them. However, understanding the API sometimes gets a bit tough for an individual. The way of developing the APIs and using them is different on different platforms. The major problem arises when the APIs are shared with a third developer. It gets tough for them to understand how exactly it will work. APIs are valuable if they can be properly implemented, accessible and for that, it should be properly documented.

Bad documentation just eats up time and the developers have worked hard to standardize the way of writing the document and its vocab. That’s where Swagger comes into the picture.

Swagger defines a set of rules and tools to semantically define APIs. It can be called as a framework for describing your APIs in a standard common language that everyone can understand.

There are many such frameworks around but Swagger comes with many benefits.

* Swagger is understandable by technical as well as non-technical users. Because of its friendly UI, it can be understandable by project managers, BA’s, & even the clients.
* Testing and debugging APIs gets easier.
* Readable by both human and machine, you can easily use it to automate the API process.

APIs that use Swagger are easy to understand, modify, and consume. Everything gets clear and that is the reason why big companies are using it in their processes.

Previously, the API development was a code-first approach which made it very difficult to document it afterwards. With Swagger, you can actually design the way you want it first, then the code which makes it clear in understanding and you can complete it with a clean code.

Configure Swagger in ASP.net Core Web API using Swashbuckle.AspNetCore

Installing the Package

The first step is to install the Swashbuckle package.

We can execute the following command in the Package Manager Console window:

**Install-Package Swashbuckle.AspNetCore**

This will install the Swashbuckle package in our application.

Configuring the Swagger Middleware

The next step is to configure the Swagger Middleware.

Let’s make the following changes in the ConfigureServices() method of the Startup.cs class:

public void ConfigureServices(IServiceCollection services)

{

// Register the Swagger generator, defining 1 or more Swagger documents

**services.AddSwaggerGen(c =>**

**{**

**c.SwaggerDoc("v1", new OpenApiInfo { Title = "My API", Version = "v1" });**

**});**

services.AddControllers();

}

This adds the Swagger generator to the services collection.

In the Configure() method, let’s enable the middleware for serving the generated JSON document and the Swagger UI:

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

// Enable middleware to serve generated Swagger as a JSON endpoint.

**app.UseSwagger();**

// Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),

// specifying the Swagger JSON endpoint.

**app.UseSwaggerUI(c =>**

**{**

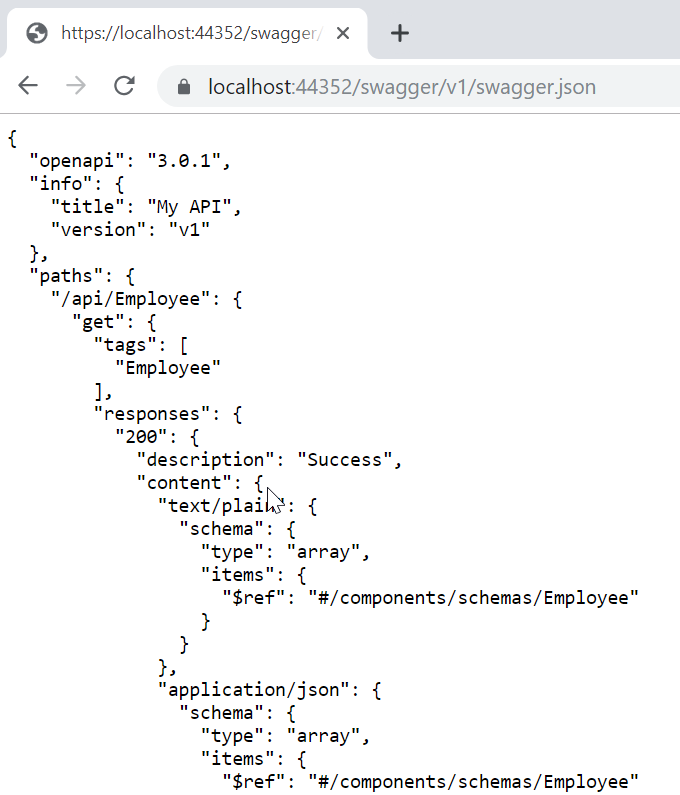
**c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");**

**});**

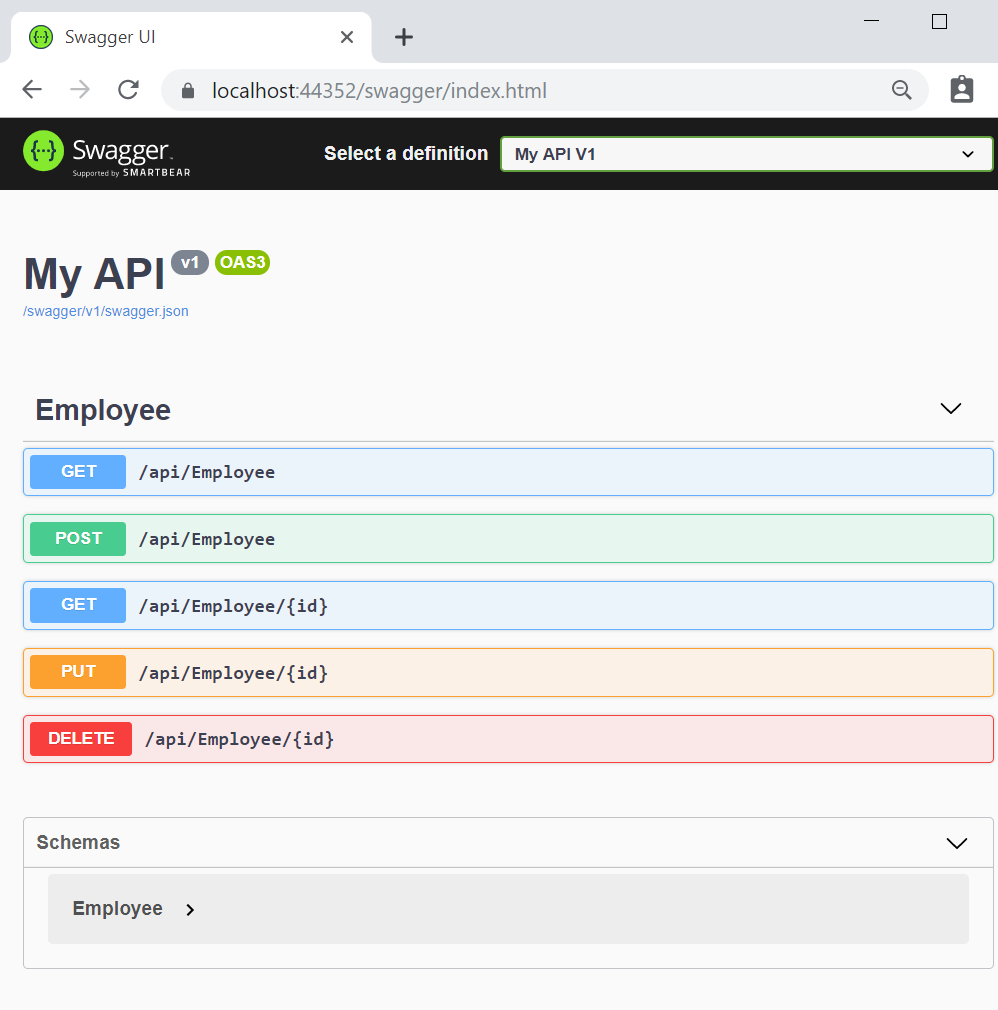
}

By executing these steps, The Swagger is configured and ready to use in our project.

Now, let’s run the app and navigate to [https://localhost](http://localhost/):<port>/swagger/v1/swagger.json. We can see that a document describing the endpoints is generated:



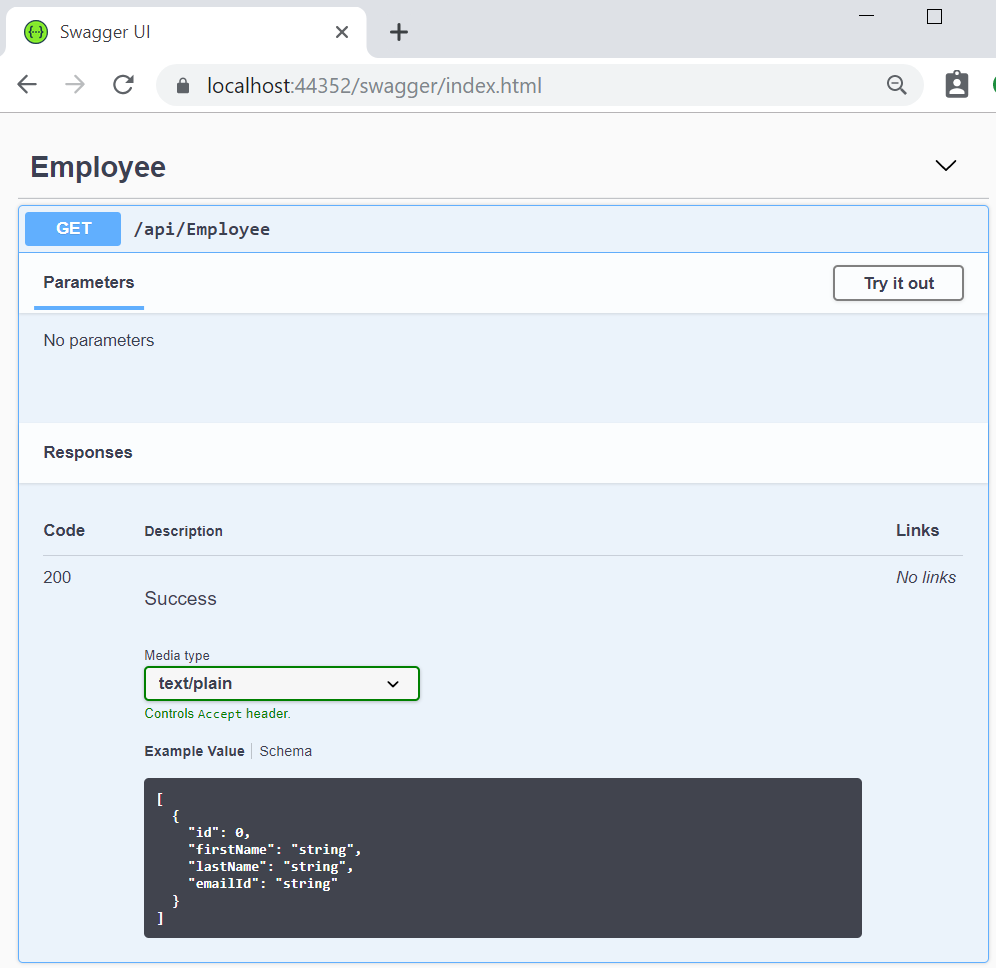
The Swagger UI can be found at [http://localhost](http://localhost/):<port>/swagger:



Now we can explore the API via the Swagger UI and it will be easier to incorporate it into other applications.

We can see each controller and its action methods listed here.

Once we click on an action method, we can see detailed information like parameters, response, and example values. There is also an option to try out each of those action methods:



By clicking on the Try it out button, we can test the endpoint and see the response:

