Testing WEB APIs is a challenge, and it has a dependency on various third-party tools for proper and efficient testing. But swagger can make it easy. [Swagger](https://swagger.io/) is an UI representation of your RESTful API. Swagger UI allows anyone — be it your development team or your end consumers — to visualize and interact with the API’s resources having none of the implementation logic in place. It’s automatically generated from your Swagger specification, with the visual documentation, making it easy for back-end implementation and client-side consumption. In this post, we’ll see how to add Swagger to ASP.NET Core 2.0 Web API.

## **Add Swagger to ASP.NET Core 2.0 Web API**

Create a ASP.NET Core 2.0 WEB API project and install **Swashbuckle.AspNetCore** nuget package. Swashbuckle comprises three packages – a Swagger generator, middleware to expose the generated Swagger as JSON endpoints and middleware to expose a swagger-ui that’s powered by those endpoints. They can be installed together, via the **Swashbuckle.AspNetCore** meta-package. Like:

**Swashbuckle.AspNetCore** meta-package. Like:

|  |  |
| --- | --- |
|  | PM > Install-Package Swashbuckle.AspNetCore |

The default template has ValuesContoller with API action method for all HTTP verbs. We’ll use the same for this post for showing Swagger UI.

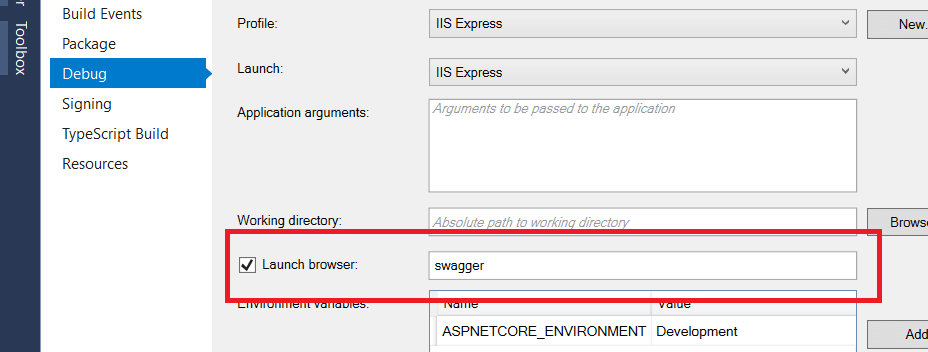
Open Startup.cs file to add swagger service to middleware. Like:

|  |  |
| --- | --- |
|  | public void ConfigureServices(IServiceCollection services) |
|  | { |
|  | services.AddMvc(); |
|  | services.AddSwaggerGen(c => |
|  | { |
|  | c.SwaggerDoc("v1", new Info |
|  | { |
|  | Version = "v1", |
|  | Title = "My API", |
|  | Description = "My First ASP.NET Core Web API", |
|  | TermsOfService = "None", |
|  | Contact = new Contact() { Name = "Talking Dotnet", Email = "contact@talkingdotnet.com", Url = "www.talkingdotnet.com" } |
|  | }); |
|  | }); |
|  | } |

And enable the Swagger UI in Configure() method.

|  |  |
| --- | --- |
|  | public void Configure(IApplicationBuilder app, IHostingEnvironment env) |
|  | { |
|  | if (env.IsDevelopment()) |
|  | { |
|  | app.UseDeveloperExceptionPage(); |
|  | } |
|  |  |
|  | app.UseMvc(); |
|  | app.UseSwagger(); |
|  | app.UseSwaggerUI(c => |
|  | { |
|  | c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1"); |
|  | }); |
|  | } |

This is all required to set up Swagger. The one last thing to do is to change the application launch URL so that Swagger UI loads on launch. To set Swagger UI as launch URL, right-click on Project -> select properties -> navigate to debug tab. On debug tab, change Launch Browser value to “**swagger**”.



Run the APP and you should see Swagger UI for ValuesContoller.

