COMPUTING SUBJECT:	Restful ASP.Net Core-services
TYPE:	Assignment
IDENTIFICATION:	RestService#3
COPYRIGHT:	*Peter Levinsky & Michael Claudius*
LEVEL:	Medium
TIME CONSUMPTION:	1 – 1½ hours
EXTENT:	40-60 lines
OBJECTIVE:	Restful services with helping information
PRECONDITIONS:	Rest service theory. Http-concepts Computer Networks Ch. 2.2
COMMANDS:	

IDENTIFICATION: RestService#3 / PELE with kindly respect and inspiration from MICL

Overall Purpose

The overall purpose for the group of 'RestService' assignments is to be able to provide and consume restful ASP.Net Core web services, to prepare the 'RestService' to be published in Azure, including testing the service and finally to setup the 'RestService' to be consumed from a browser (e.g. using Typescript) i.e. support CORS.

The whole group of assignments consist of 7 steps:

1. A simple REST Service with CRUD.
2. More advanced and complex URI's.
3. **Adding help-pages to the REST Service (Swagger) (this assignment)**
4. Testing a REST Service and publish in Azure.
5. Consuming a REST service from a C# Console application.
6. Adding Support for CORS to the REST Service
7. A REST Service using a database

Background Material:

The HTTP protocol: See Computer Network chap 2 pp. 111-136

Note of REST (Peter Levinsky): See NetHttpNote.pdf

Oswago Universitet: RESTful Service Best Practices: Recommendations for Creating Web Services: See http://cs.oswego.edu/~alex/teaching/csc435/RESTful.pdf

Usefull tools (Postman & Fiddler): See Tools.htm (tool #3 & tool #4)

This Assignment: RestService#3

Purpose
The purpose of this assignment is add helping information to your REST Service.

Mission
You are to improve your Restful web services based on the ASP.Net Core services. You will be able to add help-pages to your REST-Service together with simple 'try it' pages. This you shall do in four steps:

1. Add Swagger to your project

2. Configure Swagger in your project 2.a = .Net Core 2.1 2.b = Core 3.1!!

3. Try the help pages

4. More swagger features

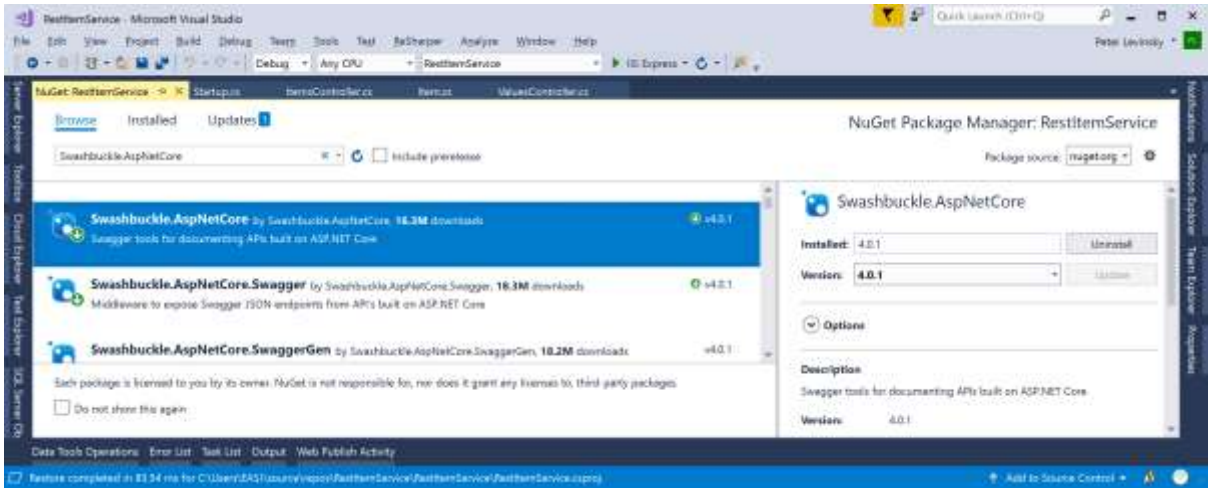
Additional reading:

- **Swaggers homepage: https://swagger.io/**
- **OpenAPI (swagger) with C# and Visual Studio: https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-2.2**
- **How to use Swagger in Visual Studio .Net Core: https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-2.2&tabs=visual-studio**

.Net Core 2.1 version

Assignment 1: Add Swagger to your project

You must use NuGet to associate swagger to your REST Service project. I.e. go to manage NuGet browse for '**Swashbuckle.AspNetCore**' - version5.6.1 and you get something like this:



Install the package '**Swashbuckle.AspNetCore**'.

Assignment 2a: Configure Swagger in your project

****** This is for .Net Core Version 2.1 !!! **** se below for 3.1
You must now configure your REST Service to use this swagger to making help-pages.

Open the '**startup.cs**' file.
In the '**ConfigureServices**' – method add below 'services.AddMvc….'

```
services.AddSwaggerGen(c =>
   {
      c.SwaggerDoc("v1", new Info { Title = "Items API", Version = "v1.0" });
   });
```

Then in the '**Configure**' – method add before 'app.UseMvc…'

```
app.UseSwagger();

app.UseSwaggerUI(c =>
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "Items API v1.0")
);
```

*Be aware the first parameter en SwaggerGen ("v1"... must be the same as in the url for the SwaggerUI ("/swagger/**v1**/swagger.json")*
********** End version 2.1 **********

.Net Core 3.0 version

Assignment 2b: Configure Swagger in your project

You must now configure your REST Service to use this swagger to making help-pages.

Open the '**startup.cs**' file.
In the '**ConfigureServices**' – method add below 'services.AddMvc….'

```
services.AddSwaggerGen(c =>
   {
      c.SwaggerDoc("v1", new OpenApiInfo{ Title = "Items API", Version = "v1.0" });
   });
```

Then in the '**Configure**' – method add before 'app.UseMvc…'

```
app.UseSwagger();

app.UseSwaggerUI(c =>
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "Items API v1.0")
);
```

*Be aware the first parameter en SwaggerGen ("v1"... must be the same as in the url for the SwaggerUI ("/swagger/**v1**/swagger.json")*

Assignment 3: Try the help pages (both 2.1 and 3.0)

Run your REST-Service and use the URL: http://localhost:{your-own-portnumber}/swagger

What do you see?

Try the 5 different URI's i.e. two Get, one Post, one Put and one Delete.

Assignment 4 (advanced): More swagger

You can do even more with Swagger. Here comes some examples:

a. Extend information: in startup->ConfigureService in the method ‘AddSwaggerGen’ give some parameters for the Info like:

```
new Info()                          // .Net Core 2.1
        {
           Title = "Items API",
           Version = "v1.0",
           Description = "Example of OpenAPI for api/localItems",
           TermsOfService = "None",
           Contact = new Contact()
           {
              Name = "{your-name}",
              Email = "{your-email}",
              Url = "{your-url}"
           },
           License = new License()
           {
              Name = "No licence required",
              Url = String.Empty
           }
        }


new OpenApiInfo()                   // .Net Core 3.0
        {
           Title = "Items API",
           Version = "v1.0",
           Description = "Example of OpenAPI for api/localItems",
           TermsOfService = new Uri({any URL}),
           Contact = new OpenApiContact()
           {
              Name = "{your-name}",
              Email = "{your-email}",
              Url = new Uri("({any URL}")
           },
           License = new OpenApiLicense()
           {
              Name = "No licence required",
              Url = new Uri("({any URL}")
           }

        }
```

What happen now?

b. Modify the URL to help in startup->Configure in the method for UseSwaggerUI change the information to:

```
{
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "Items API v1.0");
```

```
    c.RoutePrefix = "api/help";
}
```

What happen now?

c. Generate comments
   a. For all your method in the '**ItemsController**' generate comments by using the /// - notation.
   b. Refactor your REST service in startup->ConfigureService. In the AddSwaggerGen – method Add following lines after the SwaggerDoc – method following lines:

      ```
      var xmlFile =
      $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
      var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
      c.IncludeXmlComments(xmlPath);
      ```

   c. Last thing to do is to refactor your .csproj-file. I.e. right-click on your project in the solution explore – choose edit 'xxxxx.csproj' and add following to the file:

      ```
      <PropertyGroup>
        <GenerateDocumentationFile>true</GenerateDocumentationFile>
        <NoWarn>$(NoWarn);1591</NoWarn>
      </PropertyGroup>
      ```

      Save the file.

   d. Run your REST Service
      Lookup in your project-folder:
      <<projectfolder>>\RestItemService\bin\Debug\netcoreapp2.1 where you find the xml file 'RestItemService.xml' with all your documentation.
      Can be used later when implementing Client-programs

Now you have a fine REST service next step will to test the service to see if it do function correctly