Build an ASP.NET Core Service, and App with Core 2.2 Two-Day Hand-On Lab

Lab 7

This lab is the second in a series that builds the SpyStore RESTful service. This lab configures the service for Swagger documentation. Prior to starting this lab, you must have completed Lab 6.

Part 1: Configure the Application with Swagger

Step 1: Update the using statements in the Startup.cs class

1) Add the following using statements to the top of the `Statup.cs` class:

```
using System.IO;
using System.Reflection;
using Swashbuckle.AspNetCore.Swagger;
```

Step 2: Add the Required Services to the Dependency Injection Container

1) Navigate to the `ConfigureServices` method of the `Startup.cs` file and add the following code to add the `SwaggerGen` service (make sure the Url for the license uses the same port as found in launchSettings.json):

```
services.AddSwaggerGen(c =>
{
  c.SwaggerDoc("v1",
    new Info
    {
      Title = "SpyStore Service",
      Version = "v1",
      Description = "Service to support the SpyStore sample eCommerce site",
      TermsOfService = "None",
      License = new License
      {
        Name = "Freeware",
        Url = "http://localhost:38080/LICENSE.txt"
      }
    });
  var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
  var xmlPath = Path.Combine(AppContext.BaseDirectory,xmlFile);
  c.IncludeXmlComments(xmlPath);
});
```

Step 3: Configure the application to use Swagger

1) Add the following code to the Configure method, **before** the app.UseCors method call. Static File support must be added for the Swagger UI display:

```
app.UseSwagger();
app.UseSwaggerUI(c =>
{
  c.SwaggerEndpoint("/swagger/v1/swagger.json","SpyStore Service v1");
});
app.UseStaticFiles();
```

Step 4: Configure the project to generate the XML documentation

Swagger uses reflection to display the available URIs in a service. It also uses the triple-slash comments to

1) Edit the SpyStore.Service.csproj file to add the following node (shown in bold):

```
<PropertyGroup>
  <TargetFramework>netcoreapp2.2</TargetFramework>
  <GenerateDocumentationFile>true</GenerateDocumentationFile>
  <LangVersion>latest</LangVersion>
</PropertyGroup>
```

Step 5: Add the License File

1) Add a new directory named wwwroot into the SpyStore.Service project.

2) Add a new text file named License.txt to the wwwroot directory. Add any content into the file (sample content is located in the Code/Completed/Lab7 project.

Step 6: Update the launchSetting.json file

1) Open the launchSettings.json file and update the launch URLs to be the Swagger UI page (changes shown in bold – note that your port will most likely be different. **DO NOT** change the port in the IIS Express profile! DO update the port in the SpyStore.Hol.Service profile to match the port in the IIS Express profile)

```
{
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:38080",
      "sslPort": 0
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "launchUrl": "swagger",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
```

All files copyright Phil Japikse (http://www.skimedic.com/blog)

```
      }
    },
    "SpyStore.Hol.Service": {
      "commandName": "Project",
      "launchBrowser": true,
      "launchUrl": "swagger",
      "applicationUrl": "http://localhost:38080",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

Summary

This lab configured Swagger and SwaggerUI for the service.

Next steps

In the next part of this tutorial series, you will add a global exception filter into the service.