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

Lab 10

This lab is the fifth (and last) in a series that builds the SpyStore RESTful service. This lab configures the application to run entirely in Docker. Prior to starting this lab, you must have completed Lab 9.

Part 1: Add a Dockerfile to the SpyStore.Service Project

To configure the service to run in Docker, start with a `DockerFile`. This will be referenced by the Docker-Compose orchestration later in this lab.

Visual Studio

1) Right click on the SpyStore.Hol.Service and select Add -> Docker support. This will create a Dockerfile in the project and add a new profile in `launchSettings.json` for running in Docker.
   NOTE: Be sure to select Linux containers and not Windows containers

Create the Dockerfile Manually

1) Create a file named Dockerfile in the root of the SpyStore.Hol.Service project. Update the file to the following:

```
FROM mcr.microsoft.com/dotnet/core/aspnet:2.2-stretch-slim AS base
WORKDIR /app
EXPOSE 80

FROM mcr.microsoft.com/dotnet/core/sdk:2.2-stretch AS build
WORKDIR /src
COPY ["SpyStore.Hol.Service/SpyStore.Hol.Service.csproj", "SpyStore.Hol.Service/"]
COPY ["SpyStore.Hol.Models/SpyStore.Hol.Models.csproj", "SpyStore.Hol.Models/"]
COPY ["SpyStore.Hol.Dal/SpyStore.Hol.Dal.csproj", "SpyStore.Hol.Dal/"]
RUN dotnet restore "SpyStore.Hol.Service/SpyStore.Hol.Service.csproj"
COPY . .
WORKDIR "/src/SpyStore.Hol.Service"
RUN dotnet build "SpyStore.Hol.Service.csproj" -c Release -o /app

FROM build AS publish
RUN dotnet publish "SpyStore.Hol.Service.csproj" -c Release -o /app

FROM base AS final
WORKDIR /app
COPY --from=publish /app .
ENTRYPOINT ["dotnet", "SpyStore.Hol.Service.dll"]
```

2) Update the SpyStore.Hol.Service.csproj file for Docker (update in bold):

```
<PropertyGroup>
```

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

3) Update the launchSettings.json file to the following making sure the all of the ports are consistent (Changes in bold):

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

Part 2: Create the Docker-Compose Orchestration

Visual Studio

1) Right click on the `SpyStore.Service.Hol` project and select Add -> Container Orchestration Support. Select Docker-Compose if there is an orchestration option. This adds a new project named docker-compose into the solution.
   NOTE: Be sure to select Linux containers and not Windows containers
2) Update the docker-compose.yml file to the following code. This creates an image for the database (named db) and then pulls in the image for the service. The depends on makes sure the database container is up and running before the service starts:

```
version: '3.4'

services:
  db:
    image: "microsoft/mssql-server-linux:2017-latest"
    ports:
      - "7433:1433"
    environment:
       SA_PASSWORD: "P@ssw0rd"
       ACCEPT_EULA: "Y"
       MSSQL_PID: "Express"
  spystore.hol.service:
    image: ${DOCKER_REGISTRY-}spystoreholservice
    ports:
      - "38080:80"
    build:
      context: .
      dockerfile: SpyStore.Hol.Service/Dockerfile
    depends_on:
       - db
```

3) Update the connection string in the appsettings.Development.json file:

```
"ConnectionStrings": {
    //Docker-Compose
    "SpyStore": "Server=db;Database=SpyStoreHol;User
Id=sa;Password=P@ssw0rd;MultipleActiveResultSets=true"
  }
```

Manual

1) Create the above listed `docker-compose.yml` file in the root of the solution.

Part 3: Run the project in Docker

Visual Studio

1) Select docker-compose from the drop down and click F5.

Manual

1) Open a command prompt in the location of the `docker-compose.yml` file. To run the project, enter:

```
Docker-compose up
```

Summary

This lab configured Docker and the Docker-Compose orchestration.

Next steps

In the next part of this tutorial series, you will start building the ASP.NET Core Web Application.