Build an EF Core and ASP.NET Core App HOL

Lab 1

This lab walks you through creating the projects and adding/updating the NuGet packages. Prior to starting this lab, you must have completed Lab 0, Installing the Prerequisites.

Part 1: Creating the Solution and Projects

Visual Studio (all versions) is capable of creating projects and solutions, but it is much more efficient to use the .NET Core command line. When creating projects using the command line, the names of solutions, projects, and directories are case sensitive.

Step 0: Create a new directory

1. Using a directory that you have been granted full permissions (e.g. documents), create a new directory for this workshop. For example, c:\users\[username]\AutolotWorkshop. Do not use windows\system or any other folder with limited permissions.
2. Open a command prompt and navigate to that directory. It does **not** have to be a Visual Studio command prompt.

Step 1: Pin the .NET Core SDK Version

When using the command line, .NET Core will use the most current version of the SDK installed. For this reason, the it is suggested that the required version of .NET Core for your project get pinned with a global.json file.

1. Check current version by typing:

```
dotnet --version
```

2. If the version is not the version you need (e.g. 3.1.400) or if you want to make sure subsequent updates to your development machine don't adversely affect your project, enter the following command to create a new file named `global.json`:

```
dotnet new globaljson --sdk-version 3.1.400
```

3. This creates the global.json file with the following content:

```
{
    "sdk": {
        "version":"3.1.400"
    }
}
```

4. Execute dotnet --version again and the version will return as 3.1.400.

Step 2: Create the Solution

The templates that are installed with .NET Core range from very simple to quite complex. Creating the global.json file is an example of a simple template, as is creating a new solution.

1. To create a new solution file named `AutoLot`, enter the following command:

```
dotnet new sln -n AutoLot
```

Step 3: Create the ASP.NET Core project

1) The `mvc` template is extremely configurable. In addition to setting the name, authentication scheme, and directory location, there are many other options. These can be explored by entering:

```
dotnet new mvc -h
```

2) From the same directory as the solution file, enter the following command to create the ASP.NET Core project using the model-view-controller pattern, set the name, turn off authentication, and set the directory for the project as `AutoLot.Web`:

```
dotnet new mvc -n AutoLot.Web -au none -o .\AutoLot.Web
```

3) Add the project to the solution with the following command:

```
dotnet sln AutoLot.sln add AutoLot.Web
```

Step 4: Create the Models and Data Access Layer projects

The `classlib` template create .NET Core class libraries or .NET Standard class libraries. For this lab, use .NET Core class libraries.

1) The template takes a name and an output directory. Create the `AutoLot.Dal` and `AutoLot.Models` projects by entering the following commands:

```
dotnet new classlib -n AutoLot.Dal -o .\AutoLot.Dal -f netcoreapp3.1
dotnet new classlib -n AutoLot.Models -o .\AutoLot.Models -f netcoreapp3.1
```

2) Add the projects to the solution with the following commands:

```
dotnet sln AutoLot.sln add AutoLot.Dal
dotnet sln AutoLot.sln add AutoLot.Models
```

3) Update the projects to enable C# 8 nullability (do this update to both AutoLot.Dal and AutoLot.Models:

```
<PropertyGroup>
  <TargetFramework>netcoreapp3.1</TargetFramework>
  <Nullable>enable</Nullable>
</PropertyGroup>
```

Step 5: Create the Unit Test project [Optional]

The xUnit template creates a new .NET Core class library with all of the required packages for unit testing. This include xUnit and the `Microsoft.NET.Test.Sdk`.

1) Create the project with the name of `AutoLot.Dal.Tests` and the directory `AutoLot.Dal.Tests` using the following command:

```
dotnet new xunit -n AutoLot.Dal.Tests -o .\AutoLot.Dal.Tests
```

2) Add the project to the solution with the following command:

```
dotnet sln AutoLot.sln add AutoLot.Dal.Tests
```

Part 2: Add the Project References

Step 1: Update the AutoLot.Web Project

The `AutoLot.Web` project references the `AutoLot.Dal` and the `AutoLot.Models` projects.

1) Enter the following commands to add the references:

```
dotnet add AutoLot.Web reference AutoLot.Models
dotnet add AutoLot.Web reference AutoLot.Dal
```

Step 2: Update the AutoLot.Dal Project

The `AutoLot.Dal` project references the `AutoLot.Models` project.

1) Enter the following command to add the reference:

```
dotnet add AutoLot.Dal reference AutoLot.Models
```

Step 3: Update the AutoLot.Dal.Tests Project

The `AutoLot.Dal.Tests` project references the `AutoLot.Dal` and the `AutoLot.Models` projects.

2) Enter the following commands to add the references:

```
dotnet add AutoLot.Dal.tests reference AutoLot.Models
dotnet add AutoLot.Dal.tests reference AutoLot.Dal
```

Part 3: Add the NuGet packages

.NET Core, ASP.NET Core, and EF Core are composed of a series of NuGet packages. Additional tools are also distributed as NuGet packages. The command line can also be used for adding NuGet packages.

Step 1: Update the AutoLot.Web Project

1) Enter the following commands to add the packages (**note** the version for EF.SqlServer):

```
dotnet add AutoLot.Web package AutoMapper
dotnet add AutoLot.Web package LigerShark.WebOptimizer.Core
dotnet add AutoLot.Web package Microsoft.Web.LibraryManager.Build
dotnet add AutoLot.Web package Microsoft.VisualStudio.Web.CodeGeneration.Design
dotnet add AutoLot.Web package System.Text.Json
dotnet add AutoLot.Web package Microsoft.EntityFrameworkCore.SqlServer -v 3.1.6
```

Step 2: Update the AutoLot.Models Project

1) Add the packages with the following commands:

```
dotnet add AutoLot.Models package Microsoft.EntityFrameworkCore.Abstractions -v 3.1.6
dotnet add AutoLot.Models package AutoMapper
dotnet add AutoLot.Models package System.Text.Json
```

Step 3: Update the AutoLot.Dal Project

1) Add the packages with the following commands:

```
dotnet add AutoLot.Dal package Microsoft.EntityFrameworkCore.SqlServer -v 3.1.6
dotnet add AutoLot.Dal package Microsoft.EntityFrameworkCore.Design -v 3.1.6
dotnet add AutoLot.Dal package System.Text.Json
dotnet add AutoLot.Dal package Microsoft.EntityFrameworkCore.Tools -v 3.1.6
```

Step 4: Update the AutoLot.Dal.Tests Project

1) Add (or update) the package with the following commands:

```
dotnet add AutoLot.Dal.Tests package Microsoft.EntityFrameworkCore -v 3.1.6
dotnet add AutoLot.Dal.Tests package Microsoft.EntityFrameworkCore.SqlServer -v 3.1.6
dotnet add AutoLot.Dal.Tests package Microsoft.Extensions.Configuration.Json
dotnet add AutoLot.Dal.Tests package Microsoft.NET.Test.Sdk
dotnet add AutoLot.Dal.Tests package xunit
dotnet add AutoLot.Dal.Tests package xunit.runner.visualstudio
dotnet add AutoLot.Dal.Tests package coverlet.collector
```

Summary

This lab created all of the projects for the HOL, added the NuGet packages, and the appropriate references.

Next steps

In the next part of this tutorial series, you will start to build the data access library using Entity Framework Core.