.NET App Dev Hands-On Workshop

Blazor Lab 8A – AutoLot API Short

This lab adds an API to the `AutoLot` solution. Before starting this lab, you must have completed Blazor Lab 8.

Part 1: Create the Project and Add it to the Solution

- Create the ASP.NET Core service application project (all on one line):

```
[Windows]
dotnet new webapi -lang c# -n AutoLot.ShortApi -au none -o .\AutoLot.ShortApi --use-minimal-apis -
f net8.0
```

- Add the project to the solution and project references:

```
dotnet sln AutoLot.sln add AutoLot.ShortApi
dotnet add AutoLot.ShortApi reference AutoLot.Blazor.Models
```

- Open the project file (*.csproj) for the new projects and disable nullable types:

```
<PropertyGroup>
  <TargetFramework>net8.0</TargetFramework>
  <Nullable>disable</Nullable>
  <ImplicitUsings>enable</ImplicitUsings>
</PropertyGroup>
```

Part 2: Adjust the launchsettings.json file

- Move the https profile to the top in `AutoLot.Api`

If a profile isn't selected, the first profile in the list will be selected by default. Move the `https` profile to the top so it gets selected, not the `http` profile.

- Update the ports in the `launchSettings.json` file to 5011 for https and 5010 for http (only relevant parts shown here):

```
"iisExpress": {
  "applicationUrl": "http://localhost:5010",
  "sslPort": 5011
}
"https": {
  //omited for brevity
  "applicationUrl": "https://localhost:5011;http://localhost:5010",
},
```

Part 3: Create the BaseDataService Class

- Create a new class named `BaseDataService` in the root of the `AutoLot.ShortApi` project. Update the code to the following:

```
using AutoLot.Blazor.Models.Entities;
namespace AutoLot.ShortApi;

public static class BaseDataService
{
  public static List<Make> Makes =
  [
    new() { Id = 1, Name = "VW" }, new() { Id = 2, Name = "Ford" },
    new() { Id = 3, Name = "Saab" }, new() { Id = 4, Name = "Yugo" },
    new() { Id = 5, Name = "BMW" }, new() { Id = 6, Name = "Pinto" }
  ];
  public static List<Car> CarList =
  [
    new() { Id = 1, MakeId = 1, Color = "Black", PetName = "Zippy", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 1) },
    new() { Id = 2, MakeId = 2, Color = "Rust", PetName = "Rusty", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 2) },
    new() { Id = 3, MakeId = 3, Color = "Black", PetName = "Mel", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 3) },
    new() { Id = 4, MakeId = 4, Color = "Yellow", PetName = "Clunker", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 4) },
    new() { Id = 5, MakeId = 5, Color = "Black", PetName = "Bimmer", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 5) },
    new() { Id = 6, MakeId = 5, Color = "Green", PetName = "Hank", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 5) },
    new() { Id = 7, MakeId = 5, Color = "Pink", PetName = "Pinky", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 5) },
    new() { Id = 8, MakeId = 6, Color = "Black", PetName = "Pete", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 6) },
    new() { Id = 9, MakeId = 4, Color = "Brown", PetName = "Brownie", Price = "$45,000.00",
      MakeNavigation = Makes.First(m => m.Id == 4) },
    new() { Id = 10, MakeId = 1, Color = "Rust", PetName = "Lemon", IsDrivable = false,
      Price = "$45,000.00", MakeNavigation = Makes.First(m => m.Id == 1) }
  ];
}
```

Part 4: Update the Program.cs File

- Clear out the `Program.cs` file and update it to the following:

```
using AutoLot.ShortApi;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddCors(options =>
options.AddDefaultPolicy(builder=>builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader()));

var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCors();
app.UseHttpsRedirection();

app.MapGet("/api/cars/{id}", (int id)
  => Results.Ok(BaseDataService.CarList.FirstOrDefault(c => c.Id == id))).WithOpenApi();
app.MapGet("/api/cars/bymake/{makeId}", (int makeId)
  => Results.Ok(BaseDataService.CarList.Where(c => c.MakeId == makeId))).WithOpenApi();
app.MapGet("/api/cars", () => Results.Ok(BaseDataService.CarList)).WithOpenApi();
app.MapPost("/api/cars", () => Results.Ok()).WithOpenApi();
app.MapPut("/api/cars/{id}", (int id) => Results.Ok()).WithOpenApi();
app.MapDelete("/api/cars/{id}", (int id) => Results.Ok()).WithOpenApi();
app.MapGet("/api/makes/{id}", (int id)
  => Results.Ok(BaseDataService.Makes.FirstOrDefault(c => c.Id == id))).WithOpenApi();
app.MapGet("/api/makes", () => Results.Ok(BaseDataService.Makes)).WithOpenApi();
app.MapPost("/api/makes", () => Results.Ok()).WithOpenApi();
app.MapPut("/api/makes/{id}", (int id) => Results.Ok()).WithOpenApi();
app.MapDelete("/api/makes/{id}", (int id) => Results.Ok()).WithOpenApi();

app.Run();
```

Summary

This lab built the API to be used in the next lab.

Next steps

In the next part of this tutorial series, you will use the API from `AutoLot.Blazor`.