.NET 10 App Dev Hands-On Workshop

Blazor Lab 4 – Data Services

This lab adds data services to the `AutoLot.Blazor` project. Before starting this lab, you must have completed Blazor Lab 3.

Part 1: Add the Data Service and Interfaces

Copilot Agent Mode

Setup Prompt: Always use file scoped namespaces. Always combine attributes on a single line when possible. The project does not use nullable reference types. If there is a GlobalUsings.cs file, don't include using statements in new files if they are already in the globalusings.cs file. I prefer expression bodied members when possible. Single line if statements should still use braces. Use ternary operators when appropriate. Use internal over private. All classes and methods are public unless told otherwise. Don't add a constructor unless instructed to do so. Use primary constructors when possible and don't declare a class level variable if the parameter from the constructor can be used. Don't initialize properties unless instructed to do so. The @code block in razor components should be at the bottom of the file. All work is to be done in the AutoLot.Blazor project.

Prompt: Add the following global usings to the GlobalUsings.cs file if it does not already exist (sorted alphabetically. Don't remove any existing global using statements).
global using AutoLot.Blazor.Models.Entities;
global using AutoLot.Blazor.Models.Entities.Base;
global using AutoLot.Blazor.Models.Validation;
global using AutoLot.Blazor.Models.ViewModels;

Prompt: In the Services folder, create a new folder named Interfaces, and in that folder, create a new folder named Base. In that folder, create an interface named IDataServiceBase typed to TEntity where TEntity is a class that inherits from BaseEntity. Add the following methods to the interface:
GetEntityAsync(int id) - returns a Task<TEntity>
GetAllEntitiesAsync() - returns a Task<IEnumerable<TEntity>>
AddEntityAsync(TEntity entity) - returns a Task<TEntity>
UpdateEntityAsync(TEntity entity) - returns a Task<TEntity>
DeleteEntityAsync(TEntity entity) - returns a Task

Prompt: Add the following global usings to the GlobalUsings.cs file if it does not already exist (sorted alphabetically. Don't remove any existing global using statements).
global using AutoLot.Blazor.Services.Interfaces;
global using AutoLot.Blazor.Services.Interfaces.Base;

Prompt: In the interfaces folder, create a new interface named ICarDataService that inherits from IDataServiceBase<Car>. Add the following methods to the ICarDataService interface:
GetByMakeAsync(int makeId) - returns a Task<IEnumerable<Car>>

Prompt: In the Interfaces folder, create another interface named IMakeDataService that inherits from IDataServiceBase<Make>.

Prompt: Create a folder named Base in the Services folder, and in that folder, create an abstract class named DataServiceBase. Add two protected properties:
Makes (List<Make>)
Cars (List<Car)

Add a constructor that initializes the properties with the following data:

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

Prompt: Add the following global usings to the GlobalUsings.cs file if it does not already exist (sorted alphabetically. Don't remove any existing global using statements).
global using AutoLot.Blazor.Services.Base;

Prompt: In the services folder, create a class named CarDataService that implements ICarDataService and inherits from DataServiceBase. Implement all of the methods using the Cars and Makes properties from the DataServiceBase class.

Prompt: In the services folder, create a class named MakeDataService that implements IMakeDataService and inherits from DataServiceBase.
Implement all of the methods using the Cars and Makes properties from the DataServiceBase class.

Manual

Step 1: Add the service interfaces.

- Add the following to the `GlobalUsings.cs` file in the `AutoLot.Blazor` project:

```
global using AutoLot.Blazor.Models.Entities;
global using AutoLot.Blazor.Models.Entities.Base;
global using AutoLot.Blazor.Models.Validation;
global using AutoLot.Blazor.Models.ViewModels;
```

- In the `Services` folder in the `AutoLot.Blazor` project, add a new folder named `Interfaces`. In this folder, add a new folder named Base and add a new interface named `IDataServiceBase,` and update it to the following:

```
namespace AutoLot.Blazor.Services.Interfaces.Base;
public interface IDataServiceBase<TEntity> where TEntity : BaseEntity
{
  Task<TEntity> GetEntityAsync(int id);
  Task<IEnumerable<TEntity>> GetAllEntitiesAsync();
  Task<TEntity> AddEntityAsync(TEntity entity);
  Task<TEntity> UpdateEntityAsync(TEntity entity);
  Task DeleteEntityAsync(TEntity entity);
}
```

- Add the following to the `GlobalUsings.cs` file:

```
global using AutoLot.Blazor.Services.Interfaces;
global using AutoLot.Blazor.Services.Interfaces.Base;
```

- In the `Interfaces` folder, add two interface files: `ICarDataService.cs` and `IMakeDataService.cs`. Update the code to the following listings:

```
//ICarDataService.cs
namespace AutoLot.Blazor.Services.Interfaces;
public interface ICarDataService : IDataServiceBase<Car>
{
  Task<IEnumerable<Car>> GetByMakeAsync(int makeId);
}

//IMakeDataService.cs
namespace AutoLot.Blazor.Services.Interfaces;
public interface IMakeDataService : IDataServiceBase<Make>
{
}
```

Step 2: Add the BaseDataService class

- Create a folder named `Base` in the `Services` folder. In the `Base` folder, add a new class named `DataServiceBase.cs`. Update the code to the following:

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

- Add the following to the `GlobalUsings.cs` file:

```csharp
global using AutoLot.Blazor.Services.Base;
```

Step 3: Implement the Car data service.

- Create a new class named `CarDataService` in the `Services` folder and update it to the following:

```csharp
namespace AutoLot.Blazor.Services;
public class CarDataService : BaseDataService, ICarDataService
{
  public Task<Car> GetEntityAsync(int id)
    => Task.FromResult(Cars.FirstOrDefault(c => c.Id == id));
  public Task<List<Car>> GetAllEntitiesAsync()
    => Task.FromResult(Cars.AsEnumerable());
  public Task<Car> AddEntityAsync(Car entity)
  {
    entity.Id = Cars.Max(x => x.Id) + 1;
    entity.MakeNavigation = Makes.First(m => m.Id == entity.MakeId);
    Cars.Add(entity);
    return Task.FromResult(entity);
  }
  public Task<Car> UpdateEntityAsync(Car entity)
  {
    var car = Cars.FirstOrDefault(c => c.Id == entity.Id);
    if (car != null)
    {
      car.MakeId = entity.MakeId;
      car.Color = entity.Color;
      car.PetName = entity.PetName;
      car.Price = entity.Price;
      car.IsDrivable = entity.IsDrivable;
      car.MakeNavigation = Makes.FirstOrDefault(m => m.Id == entity.MakeId);
    }
    return Task.FromResult(car);
  }
  public Task DeleteEntityAsync(TEntity entity)
  {
    var carToRemove = Cars.FirstOrDefault(c => c.Id == entity.Id);
    if (carToRemove is not null)
    {
      Cars.Remove(carToRemove);
    }
    return Task.CompletedTask;
  }
  public Task<List<Car>> GetByMakeAsync(int makeId)
    => Task.FromResult(Cars.Where(x => x.MakeId == makeId).AsEnumerable());
}
```

Step 4: Add the Make data service implementation.

- Add a new class named `MakeDataService.cs` in the Services folder and update the code to the following:

```
namespace AutoLot.Blazor.Services;
public class MakeDataService : BaseDataService, IMakeDataService
{
  public Task<Make> GetEntityAsync(int id)
    => Task.FromResult(Makes.FirstOrDefault(c => c.Id == id));
  public Task<List<Make>> GetAllEntitiesAsync()
    => Task.FromResult(Makes.AsEnumerable());
  public Task<Make> AddEntityAsync(Make entity)
  {
    entity.Id = Makes.Max(x => x.Id)+1;
    Makes.Add(entity);
    return Task.FromResult(entity);
  }
  public Task<Make> UpdateEntityAsync(int id, Make entity)
  {
    var make = Makes.FirstOrDefault(m => m.Id == entity.Id);
    if (make != null)
    {
      make.Name = entity.Name;
    }
    return Task.FromResult(make);
  }
  public Task DeleteEntityAsync(TEntity entity)
  {
    var makeToRemove = Makes.FirstOrDefault(c => c.Id == entity.Id);
    if (makeToRemove is not null)
    {
      Makes.Remove(makeToRemove);
    }
    return Task.CompletedTask;
  }
}
```

Part 2: Configure AutoLot.Blazor

Copilot Agent Mode

Prompt: Add three JSON files named appsettings.json, appsettings.Development.json , and appsettings.Staging.json to the wwwroot folder of the AutoLot.Blazor project. Update the files to the following:

```
//appsettings.json
{
  "DealerInfo": {
    "DealerName": "Skimedic's Used Cars",
    "City": "West Chester",
    "State": "Ohio"
  }
}
```

```
//appsettings.Development.json
{
  "DealerInfo": {
    "DealerName": "Skimedic's Used Cars Development Site",
    "City": "West Chester",
    "State": "Ohio"
  }
}
```

```
//appsettings.Staging.json
{
  "DealerInfo": {
    "DealerName": "Skimedic's Used Cars Staging Site",
    "City": "West Chester",
    "State": "Ohio"
  }
}
```

Prompt: Add the following to the AutoLot.Blazor.csproj file:

```
<ItemGroup>
  <Content Update="wwwroot\appsettings.json">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </Content>
  <Content Update="wwwroot\appsettings.*.json">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </Content>
</ItemGroup>
```

Prompt: Add the following three lines to the Program.cs file just before the call to RunAsync():

```
builder.Services.AddScoped<ICarDataService, CarDataService>();
builder.Services.AddScoped<IMakeDataService, MakeDataService>();
builder.Services.Configure<DealerInfo>(builder.Configuration.GetSection(nameof(DealerInfo)));
```

Manual

Step 1: Add the AppSettings files

- Add three JSON files named appsettings.json, appsettings.Development.json , and appsettings.Staging.json to the **wwwroot** folder of the AutoLot.Blazor project. Update the files to the following:

```
//appsettings.json
{
  "DealerInfo": {
    "DealerName": "Skimedic's Used Cars",
    "City": "West Chester",
    "State": "Ohio"
  }
}
```

```
//appsettings.Development.json
{
  "DealerInfo": {
    "DealerName": "Skimedic's Used Cars Development Site",
    "City": "West Chester",
    "State": "Ohio"
  }
}
```

```
//appsettings.Staging.json
{
  "DealerInfo": {
    "DealerName": "Skimedic's Used Cars Staging Site",
    "City": "West Chester",
    "State": "Ohio"
  }
}
```

- Add the following to the AutoLot.Blazor.csproj file:

```
<ItemGroup>
  <Content Update="wwwroot\appsettings.json">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </Content>
  <Content Update="wwwroot\appsettings.*.json">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </Content>
</ItemGroup>
```

Step 2: Update the Program.cs File

- Add the following three lines to the Program.cs file just before the call to RunAsync():

```
builder.Services.AddScoped<ICarDataService, CarDataService>();
builder.Services.AddScoped<IMakeDataService, MakeDataService>();
builder.Services.Configure<DealerInfo>(builder.Configuration.GetSection(nameof(DealerInfo)));

await builder.Build().RunAsync();
```

All files copyright Phil Japikse (http://www.skimedic.com/blog)   Page **8** of **9**

Summary

This lab added shared services and configured them into the application using the configuration system.

Next Steps

The next lab will focus on working with the UI, Pages, and components.