.NET App Dev Hands-On Workshop

Blazor Lab 2 – Blazor Shared Assets

This lab begins the work with ASP.NET Core Blazor WebAssembly (WASM). Before starting this lab, you must have completed Blazor Lab 1.

Part 1: Clean up Unnecessary Scaffolded Code

Step 1: Clean AutoLot.Blazor

- Delete `Pages\Counter.razor` and `Pages\Weather.razor`.
- Delete the `wwwroot\sample-data` folder and the JSON file it contains.

Part 2: Add the Entities to the Shared project under AutoLot.Blazor

- Create a new folder named `Entities` in the `AutoLot.Blazor.Models` project. In that folder, create a new folder named `Base`, and in that folder create a new class file named `BaseEntity.cs`. Update the code to the following:

```
namespace AutoLot.Blazor.Models.Entities.Base;

public abstract class BaseEntity
{
  public int Id { get; set; }
  public long TimeStamp { get; set; }
}
```

- Rename `Class1.cs` to `GlobalUsings.cs` in the `AutoLot.Blazor.Models` project and update it to the following:

```
global using AutoLot.Blazor.Models.Entities;
global using AutoLot.Blazor.Models.Entities.Base;
global using System.ComponentModel;
global using System.ComponentModel.DataAnnotations;
```

- In the Entities folder, add two new files, `Car.cs` and `Make.cs`. Update them to match the following:

```
//Car.cs
namespace AutoLot.Blazor.Models.Entities;

public class Car : BaseEntity
{
  [Required,StringLength(50)]
  public string Color { get; set; }
  public string Price { get; set; }
  [DisplayName("Is Drivable")]
  public bool IsDrivable { get; set; } = true;
  public DateTime? DateBuilt { get; set; }
  public string Display { get; set; }
  [Required, StringLength(50), DisplayName("Pet Name")]
  public string PetName { get; set; }
  [Required, DisplayName("Make")]
  public int MakeId { get; set; }
  public Make MakeNavigation { get; set; }
  public string MakeName => MakeNavigation?.Name ?? "Unknown";
  public override string ToString()
  {
    return $"{PetName ?? "**No Name**"} is a {Color} {MakeNavigation?.Name} with ID {Id}.";
  }
}

//Make.cs
namespace AutoLot.Blazor.Models.Entities;
public class Make : BaseEntity
{
  [Required, StringLength(50)]
  public string Name { get; set; }
  public IEnumerable<Car> Cars { get; set; } = new List<Car>();
}
```

- Create a new folder named `ViewModels` in the `AutoLot.Blazor.Models` project. Create two new files, `ApiServiceSettings.cs` and `DealerInfo.cs`, and update the code to match the following listings:

```
//ApiServiceSettings.cs
namespace AutoLot.Blazor.Models.ViewModels;

public class ApiServiceSettings
{
  public ApiServiceSettings() { }
  public string Uri { get; set; }
  public string CarBaseUri { get; set; }
  public string MakeBaseUri { get; set; }
  public int MajorVersion { get; set; }
  public int MinorVersion { get; set; }
  public string Status { get; set; }
  public string ApiVersion
    => $"{MajorVersion}.{MinorVersion}"
       + (!string.IsNullOrEmpty(Status) ? $"-{Status}" : string.Empty);
}
```

```
//DealerInfo.cs
namespace AutoLot.Blazor.Models.ViewModels;

public class DealerInfo
{
  public string DealerName { get; set; }
  public string City { get; set; }
  public string State { get; set; }
}
```

Part 3: Add the Custom Validation Attributes

- Create a new folder named `Validation` in the `AutoLot.Blazor.Models` project. In that folder, create two new classes named `MustBeGreaterThanZeroAttribute.cs` and `MustNotBeGreaterThanAttribute.cs`. Update the classes to the following:

```
//MustBegreaterThanZeroAttribute
namespace AutoLot.Blazor.Models.Validation;

public class MustBeGreaterThanZeroAttribute(string errorMessage)
  : ValidationAttribute(errorMessage)
{
  public MustBeGreaterThanZeroAttribute() : this("{0} must be greater than 0") { }
  public override string FormatErrorMessage(string name)
    => string.Format(ErrorMessageString, name);
  protected override ValidationResult IsValid(object value, ValidationContext validationContext)
  {
    if (!int.TryParse(value.ToString(), out int result))
    {
      return new ValidationResult(FormatErrorMessage(validationContext.DisplayName));
    }
    return result > 0
      ? ValidationResult.Success
      : new ValidationResult(FormatErrorMessage(validationContext.DisplayName));
  }
}
```



```
//MustNotBeGreaterThanAttribute
using System.Reflection;
namespace AutoLot.Blazor.Models.Validation;

[AttributeUsage(AttributeTargets.Property, AllowMultiple = true)]
public class MustNotBeGreaterThanAttribute(
    string otherPropertyName, string errorMessage, string prefix)
  : ValidationAttribute(errorMessage)
{
  private string _otherPropertyDisplayName;
  public MustNotBeGreaterThanAttribute(string otherPropertyName, string prefix = "")
    : this(otherPropertyName, "{0} must not be greater than {1}", prefix) { }
  public override string FormatErrorMessage(string name)
    => string.Format(ErrorMessageString, name, _otherPropertyDisplayName);
  internal void SetOtherPropertyName(PropertyInfo otherPropertyInfo)
  {
    _otherPropertyDisplayName =
        otherPropertyInfo.GetCustomAttributes<DisplayAttribute>()
            .FirstOrDefault()?.Name
        ?? otherPropertyInfo.GetCustomAttributes<DisplayNameAttribute>()
            .FirstOrDefault()?.DisplayName
        ?? otherPropertyName;
  }
  protected override ValidationResult IsValid(object value, ValidationContext validationContext)
  {
    var otherPropertyInfo = validationContext.ObjectType.GetProperty(otherPropertyName);
    if (otherPropertyInfo == null)
    {
      return new ValidationResult("Unable to validate property. Please contact support");
    }
    SetOtherPropertyName(otherPropertyInfo);
    if (!int.TryParse(value.ToString(), out int toValidate))
    {
      return new ValidationResult($"{validationContext.DisplayName} must be numeric.");
    }
    var otherPropObjectValue = otherPropertyInfo.GetValue(validationContext.ObjectInstance, null);
    if (otherPropObjectValue == null || !int.TryParse(otherPropObjectValue.ToString(),
      out var otherValue))
    {
      return new ValidationResult(FormatErrorMessage(validationContext.DisplayName));
    }
    return toValidate > otherValue
        ? new ValidationResult(FormatErrorMessage(validationContext.DisplayName))
        : ValidationResult.Success;
    }
}
```

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

```
global using AutoLot.Blazor.Models.Validation;
```

- Add a new class name `AddToCartViewModel` to the `ViewModels` folder, and update the code to the following:

```
namespace AutoLot.Blazor.Models.ViewModels;

public class AddToCartViewModel
{
    public int Id { get; set; }
    [Display(Name="Stock Quantity")] public int StockQuantity { get; set; }
    public int ItemId { get; set; }
    [Required, MustBeGreaterThanZero, MustNotBeGreaterThan(nameof(StockQuantity))]
    public int Quantity { get; set; }
}
```

Part 4: Manage Client-Side Libraries

- Add a `JSON` file named `libman.json` to the root of the `AutoLot.Blazor` project. Update the file to match the following:

```
{
  "version": "1.0",
  "defaultProvider": "cdnjs",
  "libraries": [
    {
      "library": "twitter-bootstrap@5.3.2",
      "destination": "wwwroot/lib/bootstrap/",
      "files": [
        "css/bootstrap.min.css", "css/bootstrap.css"
      ]
    },
    {
      "provider": "cdnjs",
      "library": "font-awesome@5.15.4",
      "destination": "wwwroot/lib/font-awesome/",
      "files": [
        "css/all.min.css", "css/all.css",
        "webfonts/fa-brands-400.eot",
        "webfonts/fa-brands-400.svg",
        "webfonts/fa-brands-400.ttf",
        "webfonts/fa-brands-400.woff",
        "webfonts/fa-brands-400.woff2",
        "webfonts/fa-regular-400.eot",
        "webfonts/fa-regular-400.svg",
        "webfonts/fa-regular-400.ttf",
        "webfonts/fa-regular-400.woff",
        "webfonts/fa-regular-400.woff2",
        "webfonts/fa-solid-900.eot",
        "webfonts/fa-solid-900.svg",
        "webfonts/fa-solid-900.ttf",
        "webfonts/fa-solid-900.woff",
        "webfonts/fa-solid-900.woff2"
      ]
    }
  ]
}
```


- Delete the `\wwwroot\css\bootstrap` folder from `AutoLot.Blazor`. Right-click on the `libman.json` file and select "Restore Client-Side Libraries". Update the `wwwroot\Index.html` file for the new location:

```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>AutoLot.Blazor</title>
    <base href="/" />
    <link href="lib/bootstrap/css/bootstrap.min.css" rel="stylesheet" />
    <link rel="stylesheet" href="css/app.css" />
    <link href="lib/font-awesome/css/all.min.css" rel="stylesheet" />
    <link rel="icon" type="image/png" href="favicon.png" />
    <link href="AutoLot.Blazor.styles.css" rel="stylesheet" />
</head>
```

Summary

This completes the `AutoLot.Blazor.Models` project.

Next Steps

The next lab will add the data and API services into the `AutoLot.Blazor` project.