.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 AutoLot.Blazor.Models

- 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>();
}
```


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:

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

[AttributeUsage(AttributeTargets.Property, AllowMultiple = true)]
public class MustNotBeGreaterThanAttribute( string otherPropertyName, string errorMessage)
  : ValidationAttribute(errorMessage)
{
  private string _otherPropertyDisplayName;
  public MustNotBeGreaterThanAttribute(string otherPropertyName)
    : this(otherPropertyName, "{0} must not be greater than {1}") { }
  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.",
            new[] {validationContext.MemberName,otherPropertyName});
    }
    var otherPropObjectValue = otherPropertyInfo.GetValue(validationContext.ObjectInstance, null);
    if (otherPropObjectValue == null || !int.TryParse(otherPropObjectValue.ToString(),
        out var otherValue))
    {
      return new ValidationResult(FormatErrorMessage(validationContext.DisplayName),
            new[] {validationContext.MemberName,otherPropertyName});
    }
    return toValidate > otherValue
      ? new ValidationResult(FormatErrorMessage(validationContext.DisplayName),
        new[] {validationContext.MemberName,otherPropertyName})
      : ValidationResult.Success;
  }
}
```

```
//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),
        new [] {validationContext.MemberName});
    }
    return result > 0
      ? ValidationResult.Success
      : new ValidationResult(FormatErrorMessage(validationContext.DisplayName),
          new[] { validationContext.MemberName });
  }
}
```

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

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

Part 4: Add the View Models

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

```
namespace AutoLot.Blazor.Models.ViewModels;
public class DealerInfo
{
  public string DealerName { get; set; }
  public string City { get; set; }
  public string State { get; set; }
}
```

- Add a new class named `AddToCartViewModel.cs` 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 5: 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@5.3.3",
      "destination": "wwwroot/lib/bootstrap",
      "files": [
        "css/bootstrap.css",
        "css/bootstrap.min.css"
      ]
    },
    {
      "library": "font-awesome@6.5.2",
      "destination": "wwwroot/lib/font-awesome/",
      "files": [
        "css/all.min.css",
        "css/all.css",
        "sprites/regular.svg",
        "sprites/solid.svg",
        "webfonts/fa-solid-900.ttf",
        "webfonts/fa-solid-900.woff2",
        "webfonts/fa-regular-400.ttf",
        "webfonts/fa-regular-400.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` be replacing this line:

```
<link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
```

- With these lines:

```
<link href="lib/bootstrap/css/bootstrap.min.css" rel="stylesheet" />
<link href="lib/font-awesome/css/all.min.css" rel="stylesheet" />
```

Summary

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

Next Steps

The following lab will add the data services.

