.NET 10 App Dev Hands-On Workshop

Blazor Lab 3 – 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 2.

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.Models project.

Prompt: Create a new folder named Entities in the project, and in that folder create a new folder
named Base. In that folder, create a public abstract class named BaseEntity the following
properties:
Id (int, public)
TimeStamp (long, public)

Prompt: Rename Class1.cs to GlobalUsings.cs and replace the contents with the following global
using statements (sorted alphabetically):
global using AutoLot.Blazor.Models.Entities;
global using AutoLot.Blazor.Models.Entities.Base;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Validation;
global using System.ComponentModel;
global using System.ComponentModel.DataAnnotations;
global using System.Reflection;

Prompt: Update the AutoLot.Models.csproj file to include the following:
<PropertyGroup>
  <NoWarn>$(NoWarn);ASP0029</NoWarn>
</PropertyGroup>
```

```
Prompt: Back in the AutoLot.Blazor project, in the Entities folder, create a class named Car that
inherits from BaseEntity, add the [ValidatableType] attribute to the class, and the following
properties:
Color (string, public) - Required, maximum length of 50
Price (string, public)
IsDrivable (bool, public) - Display name "Is Drivable", instantiated to true
DateBuilt (DateTime?, public)
Display (string, public)
PetName (string, public) - Required, maximum length of 50, display name "Pet Name"
MakeId (int, public) - Required, display name "Make"
MakeNavigation (Make, public)
MakeName (string, public, read-only) - returns MakeNavigation?.Name or "Unknown"
Override the ToString() method to return a string in the format:
    $"{PetName ?? "**No Name**"} is a {Color} {MakeNavigation?.Name} with ID {Id}.";
Prompt: Create another class named Make in the Entities folder that inherits BaseEntity, add the
[ValidatableType] attribute to the class, and the following properties:
Name (string, public) - Required, maximum length of 50
Cars (ICollection<Car>, public) initialized to a new List<Car>()

Prompt: Create a new folder named Validation in the project and in that folder, create a class
named MustBeGreaterThanZeroAttribute that inherits from ValidationAttribute. Update the code to
the following:
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 });
  }
}
```

Prompt: Create another class in the Validation folder named MustNotBeGreaterThanAttribute that inherits from ValidationAttribute. Update the code to the following:

```
[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;
  }
}
```

Prompt: Add a new class in the Validation folder named ServiceCollectionExtensions with a static method named AddModelValidation that extends IServiceCollection and adds validation services to the service collection.

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.Validation;
```



```
Prompt: Add a new folder named ViewModels and add a class named DealerInfo with the following
properties:
DealerName (string, public)
City (string, public)
State (string, public)

Prompt: Add a new class named AddToCartViewModel with the following properties:
Id (int, public)
StockQuantity (int, public) display name "Stock Quantity"
ItemId (int, public)
Quantity (int, public), required, MustBeGreaterThanZero, MustNotBeGreaterThan "StockQuantity"
```

Manual

Part 1: Update the AutoLot.Blazor.Models Project File

- Add the following to the `AutoLot.Blazor.Models` project file:

```
<PropertyGroup>
  <NoWarn>$(NoWarn);ASP0029</NoWarn>
</PropertyGroup>
```

Part 1: 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'. Within 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;
global using System.Reflection;
```

- 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;
[ValidatableType]
public class Car : BaseEntity
{
  [Required,MaxLength(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, MaxLength(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;
[ValidatableType]
public class Make : BaseEntity
{
  [Required, MaxLength(50)]
  public string Name { get; set; }
  public ICollection<Car> Cars { get; set; } = new List<Car>();
}
```

Part 2: Add the Custom Validation Attributes and Service Extension

- Create a new folder named `Validation` in the `AutoLot.Blazor.Models` project. Create a class named `MustBeGreaterThanZeroAttribute.cs` in that folder. Update the code to the following:

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

- In that folder, create a new class named `MustNotBeGreaterThanAttribute.cs`. Update the code 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;
  }
}
```

- In that folder, create a new class named `ServiceCollectionExtensions.cs`. Update the code to the following:

```
public static class ServiceCollectionExtensions
{
  public static IServiceCollection AddModelValidation(this IServiceCollection services)
    => services.AddValidation();
}
```

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

```
global using AutoLot.Blazor.Models.Validation;
global using Microsoft.Extensions.DependencyInjection;
```

Part 3: 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; }
  [DisplayName("Stock Quantity")]
  public int StockQuantity { get; set; }
  public int ItemId { get; set; }
  [Required]
  [MustBeGreaterThanZero]
  [MustNotBeGreaterThan(nameof(StockQuantity))]
  public int Quantity { get; set; }
}
```

Summary

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

Next Steps

The following lab will add the data services.

