

# .NET App Dev Hands-On Lab

## Razor Pages\MVC Lab 7a – Custom Validation

This lab walks you through creating custom validation attributes and the related client-side scripts. Before starting this lab, you must have completed Razor Pages Lab 6 or MVC Lab 6.

### Part 1: Create the Server-Side Validation Attributes

- Add the following global using statements to the `GlobalUsings.cs` file in the `AutoLot.Services` project:

```
global using System.ComponentModel;
global using System.ComponentModel.DataAnnotations;
global using System.Reflection;
global using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
```

#### Step 1: Create the `MustBeGreaterThanZeroAttribute` attribute

- Create a new folder in the `AutoLot.Services` project named `Validation`. Add a new class named `MustBeGreaterThanZeroAttribute.cs` and update it to match the following:

```
namespace AutoLot.Services.Validation;
public class MustBeGreaterThanZeroAttribute(string errorMessage)
    : ValidationAttribute(errorMessage), IClientModelValidator
{
    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 (value is not int intValue)
        {
            return new ValidationResult(
                FormatErrorMessage(validationContext.DisplayName), new [] {validationContext.MemberName});
        }
        return intValue <= 0
            ? new ValidationResult(
                FormatErrorMessage(validationContext.DisplayName), new [] {validationContext.MemberName})
            : ValidationResult.Success;
    }
    public void AddValidation(ClientModelValidationContext context)
    {
        string propertyDisplayName =
            context.ModelMetadata.DisplayName ?? context.ModelMetadata.PropertyName;
        string errorMessage = FormatErrorMessage(propertyDisplayName);
        context.Attributes.Add("data-val-greaterthanzero", errorMessage);
        context.Attributes.Add("data-val-greaterthanzero-reevaluate", "true");
    }
}
```

- Add the following global using statements to the `GlobalUsings.cs` file in the `AutoLot.Services` project:

```
global using AutoLot.Services.Validation;
```

## Step 2: Create the MustNotBeGreaterThanAttribute Attribute

- Add a new class named `MustNotBeGreaterThanAttribute.cs`. and update the code to match the following:

```
namespace AutoLot.Services.Validation;
[AttributeUsage(AttributeTargets.Property, AllowMultiple = true)]
public class MustNotBeGreaterThanAttribute(string otherPropertyName, string errorMessage, string prefix)
    : ValidationAttribute(errorMessage), IClientModelValidator
{
    string _otherPropertyDisplayName = string.Empty;
    readonly string _prefix = prefix;
    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",
                new[] { validationContext.MemberName, otherPropertyName });
        }
        SetOtherPropertyName(otherPropertyInfo);
        if (value is not int intValue)
        {
            return new ValidationResult(FormatErrorMessage(validationContext.DisplayName),
                new[] { validationContext.MemberName, otherPropertyName });
        }
        var otherPropObjectValue = otherPropertyInfo.GetValue(validationContext.ObjectInstance, null);
        if (otherPropObjectValue is not int otherValue)
        {
            return new ValidationResult(FormatErrorMessage(validationContext.DisplayName),
                new[] { validationContext.MemberName, otherPropertyName });
        }
        return intValue > otherValue
            ? new ValidationResult(FormatErrorMessage(validationContext.DisplayName),
                new[] { validationContext.MemberName, otherPropertyName })
            : ValidationResult.Success;
    }
}
```

```

public void AddValidation(ClientModelValidationContext context)
{
    string propertyDisplayName = context.ModelMetadata.GetDisplayName();
    var propertyInfo = context.ModelMetadata.ContainerType.GetProperty(otherPropertyName);
    SetOtherPropertyName(propertyInfo);
    string errorMessage = FormatErrorMessage(propertyDisplayName);
    context.Attributes.Add("data-val-notgreaterthan", errorMessage);
    context.Attributes.Add("data-val-notgreaterthan-otherpropertyname", otherPropertyName);
    context.Attributes.Add("data-val-notgreaterthan-prefix", _prefix);
    context.Attributes.Add("data-val-notgreaterthan-reevaluate", "true");
}
}

```

### Step 3: Create the Base AddToCartViewModel

- Add a new folder named Base in the ViewModels folder of the AutoLot.Services project. Add a new class named AddToCartViewModelBase in the Base folder and update it to the following:

```
namespace AutoLot.Services.ViewModels.Base;
```

```

public class AddToCartViewModelBase
{
    public int Id { get; set; }
    [Display(Name="Stock Quantity")]
    public int StockQuantity { get; set; }
    public int ItemId { get; set; }
}

```

- Add the following global using statements to the GlobalUsings.cs file in the AutoLot.Services project:

```

global using AutoLot.Services.ViewModels;
global using AutoLot.Services.ViewModels.Base;

```

## Summary

This lab added custom validation attributes.

## Next Steps

The next lab will add the client-side validation scripts and demonstrate how to use the validation attributes.