Create a Custom [Compare] Validation Class in C# Labs

Perform these labs on your own computer using Visual Studio 2022 or later to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Create a Custom Validator Class: Check Two Date Property Values

Right mouse-click on the **ValidationClasses** folder and add a class named **CompareDateLessThanAttribute**. Replace all the code in this file with the code shown below.

```
using System.ComponentModel.DataAnnotations;
using System.Reflection;

namespace DataAnnotationsSamples;

public class CompareDateLessThanAttribute :
ValidationAttribute
{
  public CompareDateLessThanAttribute(string
propToCompare)
  {
    _propToCompare = propToCompare;
  }

  private readonly string _propToCompare;

  protected override ValidationResult? IsValid(object?
value, ValidationContext vc)
  {
    if (value != null) {
      // Get value entered
      DateTime currentValue = (DateTime)value;
      // Get PropertyInfo for comparison property
      PropertyInfo? pinfo =
vc.ObjectType.GetProperty(_propToCompare);
      // Get the value to ensure it is a valid date and
not null
      object? compare =
pinfo?.GetValue(vc.ObjectInstance, null);
      // Ensure the comparison property value is not
null
      if (compare != null) {
        // Perform the comparison
        if (currentValue > (DateTime)compare) {
          return new ValidationResult(ErrorMessage,
new[] { vc.MemberName ?? "UnknownProperty" });
        }
      }
    }

    return ValidationResult.Success;
  }
}
```

Try it Out

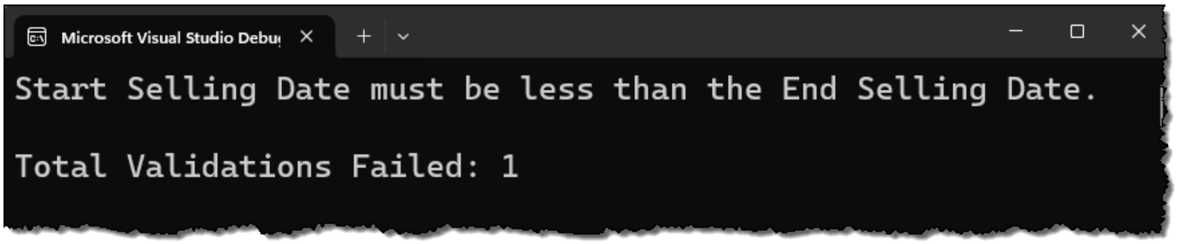
Open the **Product.cs** file and add to the **SellStartDate** property the **[CompareDateLessThan]** attribute as shown below. The first parameter to the attribute is the name of the property you want to compare it to.

```
[Display(Name = "Selling Start Date")]
[DateYearRange(-2, 5)]
[CompareDateLessThan(nameof(SellEndDate), ErrorMessage
= "Start Selling Date must be less than the End Selling
Date.")]
public DateTime SellStartDate { get; set; }
```

Open the **Program.cs** file and initialize the entity object to the following code. Notice the **SellEndDate** property is set to one day prior to the **SellStartDate**. This causes the [CompareDateLessThan] attribute to fail the validation.

```
Product entity = new() {
  ProductID = 1,
  Name = "A New Product",
  ProductNumber = "PROD001",
  ProductUrl = "http://www.advworks.com",
  Color = "Red",
  StandardCost = 5,
  ListPrice = 12,
  SellStartDate = DateTime.Today,
  SellEndDate = DateTime.Today.AddDays(-1),
  DiscontinuedDate = DateTime.Today
};
```

Run the application and view the error message you get back from the **CompareDateLessThan** class.



```
Start Selling Date must be less than the End Selling Date.

Total Validations Failed: 1
```

Lab 2: Create a Custom Validator Class: Compare Two Numeric Property Values

Right mouse-click on the **ValidationClasses** folder and add a class named **CompareDecimalLessThanAttribute.cs**. Replace all the code in this file with the code shown below.

```
using System.ComponentModel.DataAnnotations;
using System.Reflection;

namespace DataAnnotationsSamples;

public class CompareDecimalLessThanAttribute :
ValidationAttribute
{
  public CompareDecimalLessThanAttribute(string
propToCompare)
  {
    _propToCompare = propToCompare;
  }

  private readonly string _propToCompare;

  protected override ValidationResult? IsValid(object?
value, ValidationContext vc)
  {
    if (value != null) {
      // Get value entered
      decimal currentValue = (decimal)value;
      // Get PropertyInfo for comparison property
      PropertyInfo? pinfo =
vc.ObjectType.GetProperty(_propToCompare);
      // Get the value to ensure it is a valid number
and not null
      object? compare =
pinfo?.GetValue(vc.ObjectInstance, null);
      // Ensure the comparison property value is not
null
      if (compare != null) {
        // Perform the comparison
        if (currentValue > (decimal)compare) {
          return new ValidationResult(ErrorMessage,
new[] { vc.MemberName ?? "UnknownProperty" });
        }
      }
    }

    return ValidationResult.Success;
  }
}
```

Try it Out

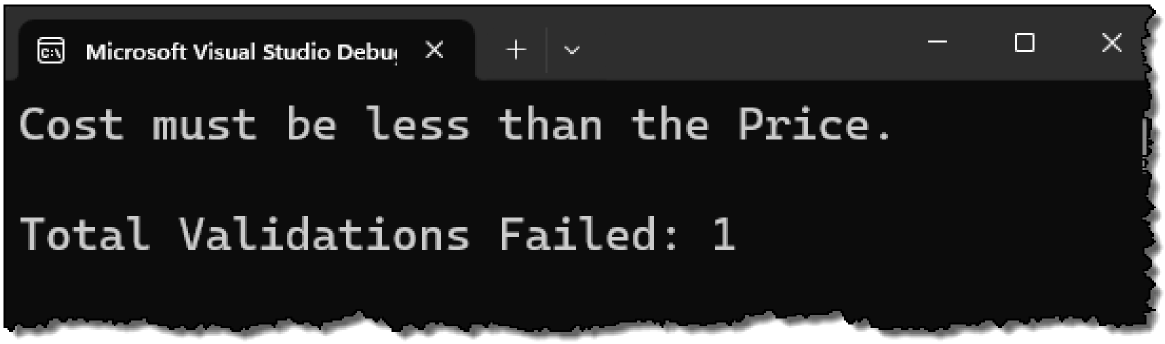
Open the **Product.cs** file and add to the **StandardCost** property the **[CompareDecimalLessThan]** attribute as shown in the code below.

```
[Display(Name = "Standard Cost")]
[Range(0.01, 9999, ErrorMessage = "{0} must be between
{1:c} and {2:c}")]
[CompareDecimalLessThan(nameof(ListPrice), ErrorMessage
= "Cost must be less than the Price.")]
public decimal? StandardCost { get; set; }
```

Open the **Program.cs** file and initialize the object to the following code. Notice the **ListPrice** property is set to a value less than the value in the **StandardCost** property. This causes the **[CompareDecimalLessThan]** attribute to fail the validation.

```
Product entity = new() {
  ProductID = 1,
  Name = "A New Product",
  ProductNumber = "PROD001",
  ProductUrl = "http://www.advworks.com",
  Color = "Red",
  StandardCost = 5,
  ListPrice = 1,
  SellStartDate = DateTime.Today,
  SellEndDate = DateTime.Today.AddDays(1),
  DiscontinuedDate = DateTime.Today
};
```

Run the application and view the error message you get back from the **CompareDecimalLessThan** class.



```
Cost must be less than the Price.

Total Validations Failed: 1
```