Build Your Own Validation Using the [CustomValidation] Attribute 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: Check for a Valid Workday Using the [CustomValidation] Attribute

There is no way that Microsoft can anticipate all the needs for business rule validation. As such, they have provided us with a couple of different methods to create custom validation using attributes. The **[CustomValidation]** attribute accepts two parameters; the first parameter is the type of a class in which you write a static method that returns a **ValidationResult** object. The second parameter is the name of that static method.

Right mouse-click on the **ValidationClasses** folder and add a new class to your project named **WeekdayOnlyValidator** and to this new file add the following code.

```
using System.ComponentModel.DataAnnotations;

namespace Common.Library;

public class WeekdayOnlyValidator
{
  public static ValidationResult? Validate(DateTime
date)
  {
    return date.DayOfWeek == DayOfWeek.Saturday ||
date.DayOfWeek == DayOfWeek.Sunday ? new
ValidationResult("Please enter a valid weekday from
Monday to Friday.") : ValidationResult.Success;
  }
}
```

The Validate() method checks the date passed in to ensure it does not fall on a Saturday or a Sunday. If the date does fall on a weekend, return a ValidationResult object with the error message inside. Otherwise, return a ValidationResult.Success from this method.

Lab 2: Create a Customer Class

Right mouse-click on the **EntityClasses** folder and add a class named **Customer**. Add the **[CustomValidation]** attribute to decorate an **NextBilling** property in the class as shown below.

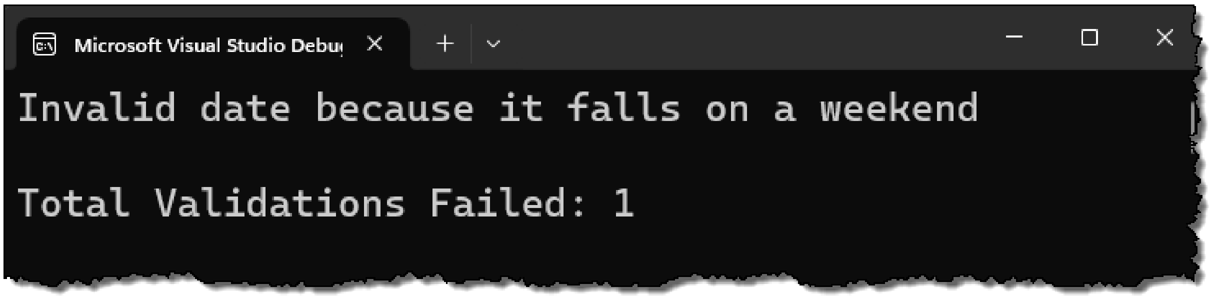
```
using Common.Library;
using System.ComponentModel.DataAnnotations;

namespace DataAnnotationsSamples;

public class Customer
{
  [Required]
  [CustomValidation(typeof(WeekdayOnlyValidator),
nameof(WeekdayOnlyValidator.Validate))]
  public DateTime NextBillingDate { get; set; }
}
```

Try it Out

Open the **Program.cs** file and add a new instantiation of the Customer class that sets the **NextBillingDate** property of the entity object to an invalid date.

```
Customer entity = new() {
  NextBillingDate = DateTime.Parse("1/14/2024")
};
```

Run this code and since the date 1/14/2024 falls on a weekend, the validation message should appear on your console window as shown below.



```
Invalid date because it falls on a weekend

Total Validations Failed: 1
```

Lab 3: Pass in a Custom Error Message to the [CustomValidation] Attribute

Let's make this WeekdayOnlyValidator class a little more flexible by being able to pass in an error message. Open the **WeekdayOnlyValidator.cs** file and modify it to look like the following.

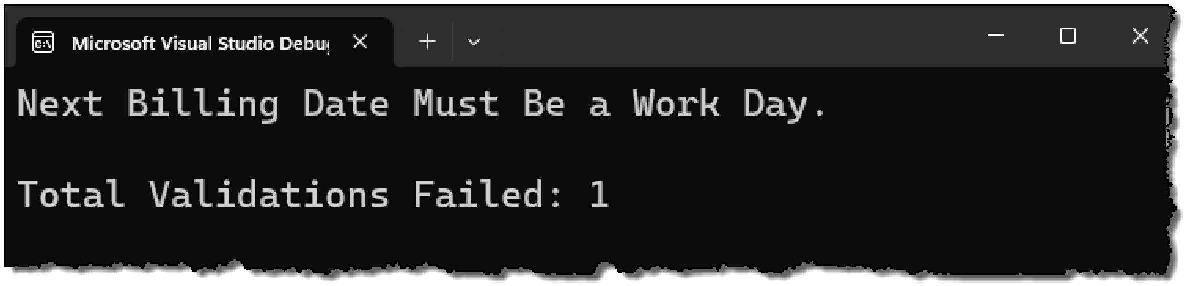
```
public class WeekdayOnlyValidator
{
  public static string ErrorMessage { get; set;  } =
string.Empty;
  public static ValidationResult? Validate(DateTime
date)
  {
    ErrorMessage ??= "Please enter a valid weekday from
Monday to Friday for {0}.";
    return date.DayOfWeek == DayOfWeek.Saturday ||
date.DayOfWeek == DayOfWeek.Sunday ? new
ValidationResult(ErrorMessage) :
ValidationResult.Success;
  }
}
```

Open the **Customer.cs** file and make it look like the following.

```
public class Customer
{
  [Required]
  [Display(Name = "Next Billing Date")]
  [CustomValidation(typeof(WeekdayOnlyValidator),
nameof(WeekdayOnlyValidator.Validate), ErrorMessage =
"{0} Must Be a Work Day.")]
  public DateTime NextBillingDate { get; set; }
}
```

Try It Out

Run the application and you should now see the following error message appear.



```
Next Billing Date Must Be a Work Day.

Total Validations Failed: 1
```