ASP.NET Core - Custom model validation

10

3

In MVC when we post a model to an action we do the following in order to validate the model against the data annotation of that model:

```
if (ModelState.IsValid)
```

If we mark a property as [Required], the ModelState.IsValid will validate that property if contains a value or not.

My question: How can I manually build and run custom validator?

P.S. I am talking about backend validator only.

c# asp.net-core asp.net-core-mvc

asked Oct 11 '17 at 18:01

user2818430

2,003 10 47 104

2 Answers

16

In .NET Core, you can simply create a class that inherits from `ValidationAttribute`. You can see the full details in the ASP.NET Core MVC Docs.

Here's the example taken straight from the docs:

```
public class ClassicMovieAttribute : ValidationAttribute
{
    private int _year;

    public ClassicMovieAttribute(int Year)
    {
         year = Year;
```

```
validationContext)
    {
        Movie movie = (Movie)validationContext.ObjectInstance;

        if (movie.Genre == Genre.Classic && movie.ReleaseDate.Year > _year)
        {
            return new ValidationResult(GetErrorMessage());
        }

        return ValidationResult.Success;
    }
}
```

I've adapted the example to exclude client-side validation, as requested in your question.

In order to use this new attribute (again, taken from the docs), you need to add it to the relevant field:

```
[ClassicMovie(1960)]
[DataType(DataType.Date)]
public DateTime ReleaseDate { get; set; }
```

Here's another, simpler example for ensuring that a value is `true`:

```
public class EnforceTrueAttribute : ValidationAttribute
{
    public EnforceTrueAttribute()
        : base("The {0} field must be true.") { }

    public override bool IsValid(object value) =>
        value is bool valueAsBool && valueAsBool;
}
```

This is applied in the same way:

```
[EnforceTrue]
public bool ThisShouldBeTrue { get; set; }
```

Edit: Front-End Code as requested:

The options are All, ModelOnly or None.

edited Apr 9 at 3:14

Jeremy Thompson
41.3k 13 113 209

answered Oct 11 '17 at 19:17

Kirk Larkin
27.2k 9 52 77

1 Could you add the front end code as well? It would help me and other readers I suspect. – w0051977 Apr 8 at 18:24

The full example with the client-side implementation is available in the docs, here. – Kirk Larkin Apr 9 at 9:14

7

To create a custom validation attribute in `.Net Core` , you need to inherit from `IModelValidator` and implement `Validate` method.

Custom validator

```
public class ValidUrlAttribute : Attribute, IModelValidator
{
    public string ErrorMessage { get; set; }

    public IEnumerable<ModelValidationResult> Validate(ModelValidationContext context)
    {
        var url = context.Model as string;
        if (url != null && Uri.IsWellFormedUriString(url, UriKind.Absolute))
        {
            return Enumerable.Empty<ModelValidationResult>();
        }

        return new List<ModelValidationResult>
        {
            new ModelValidationResult(context.ModelMetadata.PropertyName, ErrorMessage)
        };
    }
}
```

The model

```
public class Product
```

```
    [Required]
    public string ProductName { get; set; }

    [Required]
    [ValidUrl]
    public string ProductThumbnailUrl { get; set; }
}
```

> Will this approach give opportunity to work with "ModelState.IsValid" property in controller action method?

Yes! The `ModelState` object will correctly reflect the errors.

> Can this approach be applied to the model class? Or it can be used with model class properties only?

I don't know if that could be applied onto class level. I know you can get the information about the class from `ModelValidationContext` though:

- `context.Model` : returns the property value that is to be validated
- `context.Container` : returns the object that contains the property
- `context.ActionContext` : provides context data and describes the action method that processes the request
- `context.ModelMetadata` : describes the model class that is being validated in detail

Notes:

This validation attribute doesn't work with Client Validation, as requested in OP.

edited Apr 9 at 1:00

Jeremy Thompson
41.3k 13 113 209

answered Oct 11 '17 at 19:10

David Liang
7,031 2 20 34

Will this approach give opportunity to work with "ModelState.IsValid" property in controller action method? – Evgeniy Miroshnichenko Oct 10 '18 at 13:28

Can this approach be applied to the model class? Or it can be used with model class properties only? – Evgeniy Miroshnichenko Oct 10 '18 at 14:50

@EvgeniyMiroshnichenko: please see my updates. – David Liang Oct 10 '18 at 19:40