How to create Custom Data Annotation Validators

Asked 9 years, 6 months ago Active 2 years, 7 months ago Viewed 55k times

45

15

Wanting to create custom data annotation validation. Are there any useful guides / samples on how to create them?

Firstly:
StringLength with minimum and maximum length. I'm aware .NET 4 can do this, but want to do the same in .NET 3.5, if possible being able to define minimum length only (at least x chars), maximum length only (up to x chars), or both (between x and y chars).

Secondly:
Validation using modulus arithmetic - if the number is a valid length, I wish to validate using the Modulus 11 algorithm (I have already implemented it in JavaScript, so I guess it would just be a simple porting?)

Update:
Solved second problem, was just a case of copying over the JavaScript implementation and making a few tweaks, so don't need a solution for that.

asp.net asp.net-mvc-2 data-annotations

edited Aug 16 '12 at 18:26
Rap
5,850 1 38 77

asked Aug 5 '10 at 10:13
SamWM
4,496 10 51 81

2 Answers

To create a custom data annotation validator follow these gudelines:

1. Your class has to inherit from `System.ComponentModel.DataAnnotations.ValidationAttribute` class.

That's it.

IMPORTANT Caution

Sometimes developers check that value is not null/empty and return false. This is *usually* incorrect behaviour, because that's on `Required` validator to check which means that your custom validators should only validate non-null data but return `true` otherwise (see example). This will make them usable on mandatory (required) and non-mandatory fields.

Example

```
public class StringLengthRangeAttribute : ValidationAttribute
{
    public int Minimum { get; set; }
    public int Maximum { get; set; }

    public StringLengthRangeAttribute()
    {
        this.Minimum = 0;
        this.Maximum = int.MaxValue;
    }

    public override bool IsValid(object value)
    {
        string strValue = value as string;
        if (!string.IsNullOrEmpty(strValue))
        {
            int len = strValue.Length;
            return len >= this.Minimum && len <= this.Maximum;
        }
        return true;
    }
}
```

All properties can be set in attribute as you wish to set them.
Some examples:

```
[Required]
[StringLengthRange(Minimum = 10, ErrorMessage = "Must be >10 characters.")]

[StringLengthRange(Maximum = 20)]
```

When a particular property isn't set, its value is set in the constructor, so it always has a value. In above usage examples I deliberately added the `Required` validator as well, so it's in sync with the above **caution** I've written.

Important

So this validator will still work on your model value that's not required, but when it's present it validates (think of a text field in a web form, that's not required, but if a user enters a value in, it has to be valid).

edited Feb 24 '14 at 16:59
demoncodemonkey
10.4k 7 50 91

answered Aug 5 '10 at 10:39
Robert Koritnik
92.5k 45 250 374

1 so, in case of null value should isValid return true? – onof Aug 5 '10 at 10:53

So how do you add extra options, e.g. `[MyValidator(Min=1, Max=20)]` , Where both Min/Max (or just one) are optional? – SamWM Aug 5 '10 at 11:07

2 @onof: Yes. In that case it should return `true` . – Robert Koritnik Aug 5 '10 at 11:58

@Sam: I edited my answer and included an example of exactly what you're trying to do in the first scenario. – Robert Koritnik Aug 5 '10 at 12:08

@Rap it works for me in MVC 5. msdn.microsoft.com/en-us/library/cc679289(v=vs.110).aspx – demoncodemonkey Feb 25 '14 at 8:26

7

Use the `CustomValidationAttribute` together with a validate function with signature

```
public static ValidationResult Validate(MyType x, ValidationContext context)
```

Example (for a string property)

```
using System.ComponentModel.DataAnnotations;

public class MyClass
{
    [CustomValidation(typeof(MyClass), "Validate")]
    public string MyProperty { get; set; }
```

```
                ? new ValidationResult(null)
                : ValidationResult.Success;
        }
    }
```

answered Jul 13 '17 at 9:07
Micha Wiedenmann
15.6k 17 70 111

What is the ValidationContext context argument used for? – slasky Feb 22 '18 at 21:45

@petryuno1 You could use it to get the model if you happen to need to look at other properties in your vlaidation. e.g., var model = (MyClass)context.ObjectInstance; – Jason Butera Jun 8 '18 at 15:22