

Improving Your Validation Game

with Fluent Validation for .NET

Jonathan "J." Tower



Say goodbye to
DataAnnotations

and hello to
FluentValidation

Hi, I'm J.

Jonathan "J." Tower

Principal Consultant & Partner

Trailhead Technology Partners

🏆 Microsoft MVP in .NET

📁 Organizer of Beer City Code



TRAILHEAD
TECHNOLOGY PARTNERS

trailheadtechnology.com

✉ jtower@trailheadtechnology.com

🌐 trailheadtechnology.com/blog

🐦 jtowermi

<https://github.com/jonathantower/fluent-validation>

BUT FIRST...If You Give \$300, So Will I!

bit.ly/kcdc22-water

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 35,000 projects in 27 countries, benefiting over 9.5 million people.."

- Wikipedia

"4/4 Stars"

- CharityNavigator.org



charity: water

DataAnnotations Refresher

Refresher on DataAnnotations

```
public class Movie
{
    [Required]
    [StringLength(200)]
    public object Title { get; set; }

    [Required]
    public object DateReleased { get; set; }
}
```

Refresher on DataAnnotations

```
[HttpPost]
public ActionResult Edit(Movie std)
{
    if (ModelState.IsValid) {
        return RedirectToAction("Index");
    }
    return View(std);
}
```


Refresher on DataAnnotations

```
@model SampleApplication.Models.Movie
```

```
@Html.ValidationSummary(true, "")
```

```
@Html.ValidationMessageFor(model => model.Title)
```

Getting Started

With FluentValidation

Supported Frameworks



.NET CORE 2.0 +



.NET FRAMEWORK 4.6.1

Supported Frameworks



.NET CORE 2.0 +



.NET FRAMEWORK 4.6.1

Installing FluentValidation

Install-Package FluentValidation

```
dotnet add package FluentValidation
```

Install-Package FluentValidation.AspNetCore

```
dotnet add package FluentValidation.AspNetCore
```


FluentValidation

```
public class MovieValidator : AbstractValidator<Movie>
{
    public MovieValidator()
    {
        RuleFor(x => x.Title).NotEmpty()
            .WithMessage("Please specify a title");
        RuleFor(x => x.DateReleased).NotEmpty()
            .When(x => x.IsReleased);
        RuleFor(x => x.ShortSummary).Length(5, 1000);
    }
}
```

Razor Pages Demo

Switching from DataAnnotations to FluentValidation

Why Switch?

Compare and Contrast

	DataAnnotations	FluentValidation
Mixed Contexts	✗ Mixes models and validation logic	✓ Separates models from validation
Closed Models	✗ Can't validate	✓ Can validate
Conditional Validation	✗ Difficult	✓ Easy
Unit Testing	✗ Difficult	✓ Easy
Client-side Validation	✓ Supported	✓ Supported, but need to do it yourself

Basic Validators

With FluentValidation

Basic Validators

```
RuleFor(customer => customer.Surname).NotNull();
```

```
RuleFor(customer => customer.Surname).NotEmpty();
```

```
RuleFor(customer => customer.Surname).NotEqual("Foo", StringComparer.Ordinal);
```

```
RuleFor(customer => customer.Surname).Length(1, 250);
```

```
RuleFor(customer => customer.CreditLimit).GreaterThan(0);
```

Basic Validators

NotNull Validator	NotEmpty Validator	NotEqual Validator	Equal Validator	Length Validator	MaxLength Validator
MinLength Validator	Less Than Validator	Less Than Or Equal Validator	Greater Than Validator	Greater Than Or Equal Validator	Predicate Validator
Regular Expression Validator	Email Validator	Credit Card Validator	Enum Validator	Enum Name Validator	Empty Validator
	Null Validator	ExclusiveBetween Validator	InclusiveBetween Validator	ScalePrecision Validator	

Custom Messages

With FluentValidation

Custom Messages

```
RuleFor(c => c.Surname)  
    .NotNull()  
    .WithMessage("{PropertyName} was null");
```

```
RuleFor(c => c.Surname)  
    .NotNull()  
    .WithName("Last name");
```

Chaining

With FluentValidation

Chaining

```
RuleFor(c => c.Surname)  
    .NotNull()  
    .NotEqual("foo");
```

```
RuleFor(c => c.Surname)  
    .Cascade(CascadeMode.Stop)  
    .NotNull()  
    .NotEqual("foo");
```

Reminder...If You Give \$300, So Will I!

bit.ly/kcdc22-water

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 35,000 projects in 27 countries, benefiting over 9.5 million people.."

- Wikipedia

"4/4 Stars"

- CharityNavigator.org



charity: water

Conditions

With FluentValidation

Conditions

```
RuleFor(c => c.CustomerDiscount)
    .GreaterThan(0)
    .When(c => c.IsPreferredCustomer);
```

```
When(c => c.IsPreferredCustomer, () => {
    RuleFor(c => c.CustomerDiscount).GreaterThan(0);
    RuleFor(c => c.CreditCardNumber).NotNull();
}).Otherwise(() => {
    RuleFor(c => c.CustomerDiscount).Equal(0);
});
```

Custom Validators

With FluentValidation

Custom Validators With Must()

```
public class PersonValidator : AbstractValidator<Person> {  
    public PersonValidator() {  
        RuleFor(x => x.Pets)  
            .Must(list => list.Count < 10)  
            .WithMessage("List must contain less than 10 items");  
    }  
}
```

Custom Validators With Extension Method

```
public static class MyCustomValidators {  
    public static IRuleBuilderOptions<T, IList<TElement>> ItemsLessThan<T, TElement>(  
        this IRuleBuilder<T, IList<TElement>> ruleBuilder,  
        int num)  
    {  
        return ruleBuilder  
            .Must(list => list.Count < num)  
            .WithMessage("The list contains too many items");  
    }  
}
```

Custom Validators With Extension Method

```
public static class MyCustomValidators {  
    public static IRuleBuilderOptions<T, IList<TElement>> ItemsLessThan<T, TElement>(  
        this IRuleBuilder<T, IList<TElement>> ruleBuilder,  
        int num)  
    {  
        return ruleBuilder  
            .Must(list => list.Count < num)  
            .WithMessage("The list contains too many items");  
    }  
}
```

```
RuleFor(x => x.Pets).ItemsLessThan(10);
```

Nested Validators

With FluentValidation

Nested Validators

```
public class Customer
{
    public string Name { get; set; }
    public Address Address { get; set; }
}
```

```
public class Address
{
    public string Line1 { get; set; }
    public string Line2 { get; set; }
    public string Town { get; set; }
    public string County { get; set; }
    public string Postcode { get; set; }
}
```

Nested Validators

```
public class AddressValidator : AbstractValidator<Address>
{
    public AddressValidator()
    {
        RuleFor(address => address.Postcode).NotNull();
        //etc...
    }
}
```

Nested Validators

```
public class CustomerValidator : AbstractValidator<Customer>
{
    public CustomerValidator()
    {
        RuleFor(customer => customer.Name).NotNull();
        RuleFor(customer => customer.Address)
            .SetValidator(new AddressValidator());
    }
}
```

Include()

With FluentValidation

Include()

```
public class PersonAgeValidator : AbstractValidator<Person>
{
    public PersonAgeValidator()
    {
        RuleFor(x => x.DateOfBirth).GreaterThan(18);
    }
}
```

```
public class PersonNameValidator : AbstractValidator<Person>
{
    public PersonNameValidator()
    {
        RuleFor(x => x.Surname).NotNull().Length(0, 255);
        RuleFor(x => x.Forename).NotNull().Length(0, 255);
    }
}
```

Include()

```
public class PersonValidator : AbstractValidator<Person>
{
    public PersonValidator()
    {
        // other validations...

        Include(new PersonAgeValidator());
        Include(new PersonNameValidator());
    }
}
```

RuleSets

With FluentValidation

RuleSets

```
public class PersonValidator : AbstractValidator<Person>
{
    public PersonValidator()
    {
        RuleSet("Names", () =>
        {
            RuleFor(x => x.Surname).NotNull();
            RuleFor(x => x.Forename).NotNull();
        });

        RuleFor(x => x.Id).NotEqual(0);
    }
}
```

RuleSets

```
var validator = new PersonValidator();

// only name fields are validated
var result = validator.Validate(person, opts => opts.IncludeRuleSets("Names"));

// only Id is verified
var result = validator.Validate(person);

// all three fields are validated
var result = validator.Validate(person, opts => opts
    .IncludeRuleSets("Names")
    .IncludeRulesNotInRuleSet());
```

Advanced Topics

With FluentValidation

Unit Testing

```
[Test]
public void Should_have_error_when_Name_is_null()
{
    var model = new Person { Name = null };
    var result = validator.TestValidate(model);
    result.ShouldHaveValidationErrorFor(p => p.Name);
}
```

Asynchronous Validation

```
public class CustomerValidator : AbstractValidator<Customer>
{
    SomeExternalWebApiClient _client;

    public CustomerValidator(SomeExternalWebApiClient client)
    {
        _client = client;

        RuleFor(x => x.Id).MustAsync(async (id, cancellation) =>
        {
            return !(await _client.IdExists(id));
        }).WithMessage("ID Must be unique");
    }
}
```

```
var result = await validator.ValidateAsync(customer);
```


Localization

```
public class PersonValidator : AbstractValidator<Person>
{
    public PersonValidator(IStringLocalizer<Person> localizer)
    {
        RuleFor(x => x.Surname)
            .NotNull()
            .WithMessage(x => localizer["Surname is required"]);
    }
}
```

Client-Side Validation

Requires **FluentValidation.AspNetCore**

No named RuleSets are run

```
services.AddFluentValidationClientsideAdapters()
```

Rules Supported Client-Side

NotNull/NotEmpty

Matches (regex)

InclusiveBetween (range)

CreditCard

Email

EqualTo (cross-property equality comparison)

MaxLength

MinLength

Length

Demo

Enabling Client-Side Validation

Summing Up

1. FluentValidation improves on most aspects of validation over DataAnnotation
2. It separates your models from their validation code
3. It's easy to read, test, maintain
4. There are many powerful ways to customize your validation rules
5. It's compatible with most popular libraries and frameworks



Recap

DataAnnotations Refresher

Getting Started

Why Switch?

FluentValidation Features

- Basic and Custom Validators
- Chaining
- Conditions
- Nested Validators
- Rulesets
- Async, Unit Testing, Localization, Client-Side Validation

Q&A

Last Chance...If You Give \$300, So Will I!

bit.ly/kcdc22-water

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 35,000 projects in 27 countries, benefiting over 9.5 million people.."

- Wikipedia

"4/4 Stars"

- CharityNavigator.org



charity: water

Thank You! Questions?

Jonathan "J." Tower

Principal Consultant & Partner

Trailhead Technology Partners

✉ jtower@trailheadtechnology.com

🌐 trailheadtechnology.com/blog

🐦 [jtowermi](https://twitter.com/jtowermi)

<https://github.com/jonathantower/fluent-validation>



trailheadtechnology.com

KCDC Speaker Eval

