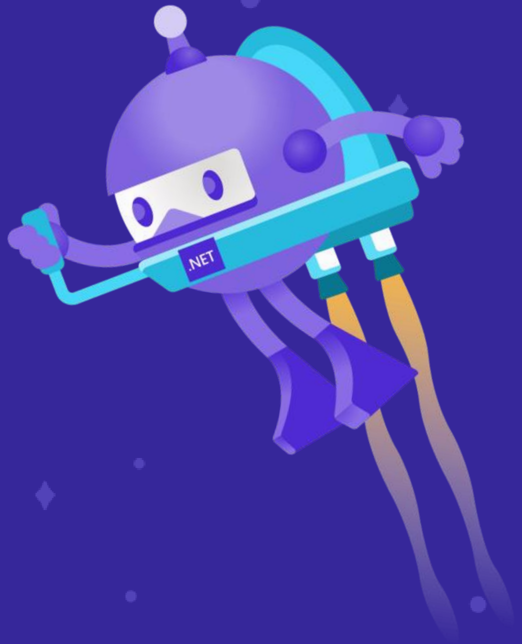
Validation Rules

Luis Matos



LUIS MATOS

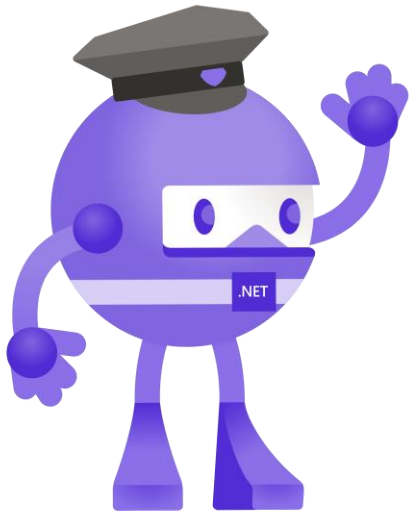
@luismatosluna

hello@luismts.com

www.luismts.com

Validation Rules

Improve the Quality of your Xamarin Forms Applications



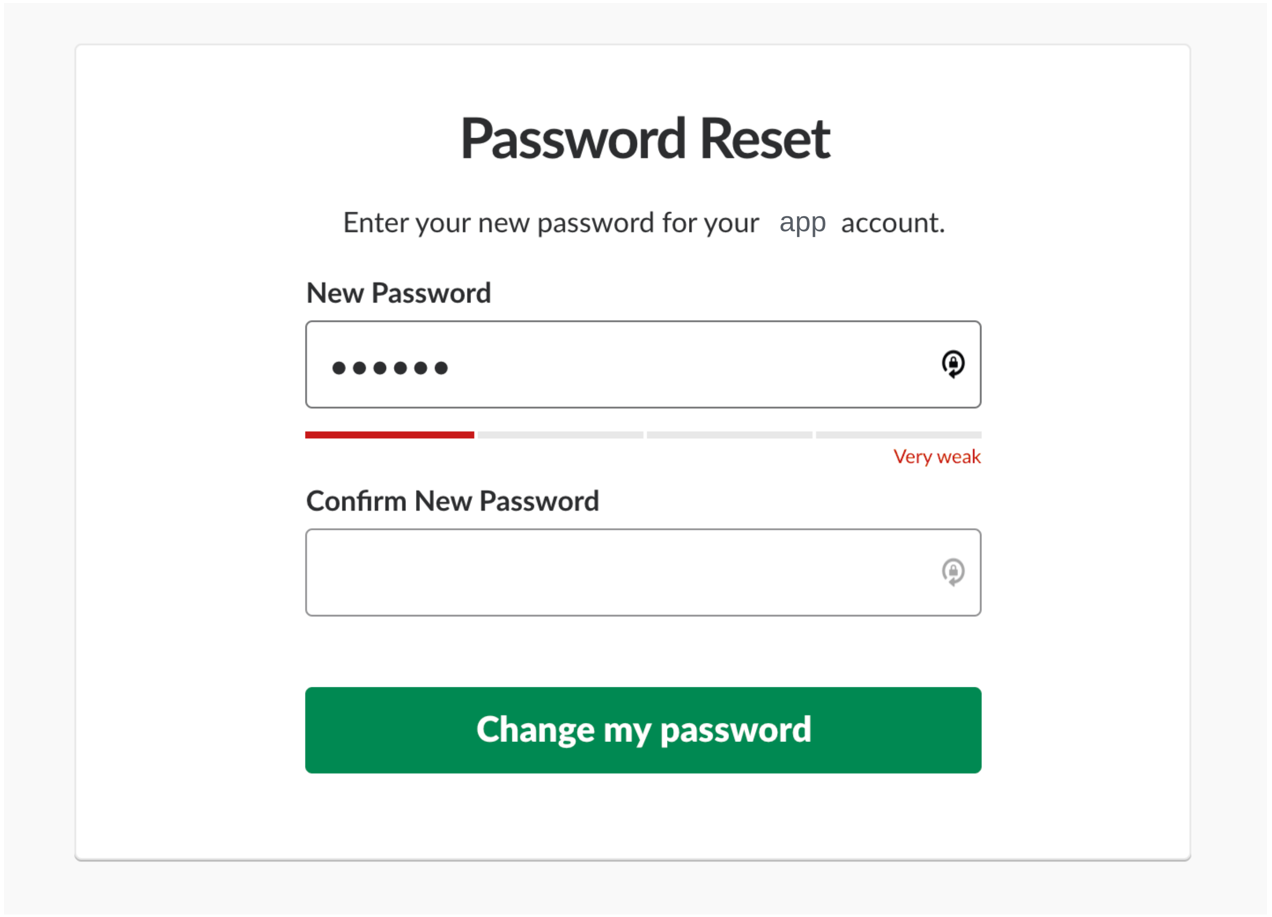
www.luismts.com

Validations

Improve the quality of your data

Any app that accepts input from users should ensure that the input is valid. An app could, for example, check for input that contains only characters in a particular range, is of a certain length, or matches a particular format.

In other words, without validation, an user can supply data that causes the app to fail. Here is where validations come to enforces business rules and prevents an attacker from injecting malicious data.



Password Reset

Enter your new password for your app account.

New Password: ••••••

Very weak

Confirm New Password:

Source: Enterprise Application Patterns eBook

Validation Libraries

For the .NET Ecosystem

ValidationRule
.NET Core

FluentValidation
.NET Library

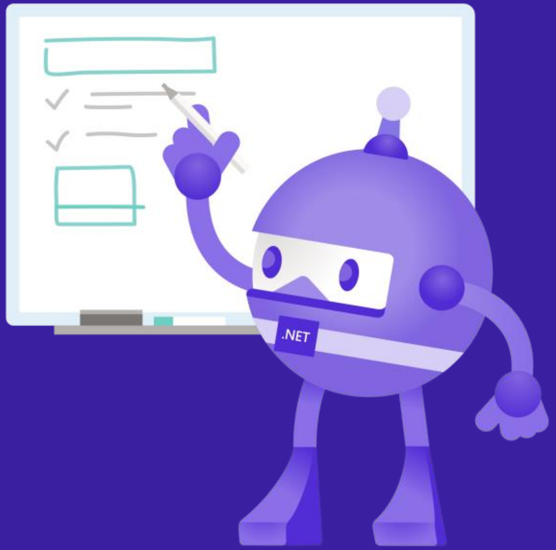
ReactiveUI.Validation
ReactiveUI

Plugin.ValidationRules
My Library ☺

LUIS MATOS

Plugin.ValidationRules

github.com/luismts/ValidationRulesPlugin



www.**luismts**.com

Rules

ValidationRule<T> Interface

```
...
public class IsNotNullOrEmptyRule: IValidationRule<string>
{
     public string ValidationMessage { get; set; }

     public bool Check(string value)
     {
          return value != null;
     }
}
...
```

Add the rules you want into your project. For that you must add a class that derives from **ValidationRule<T>** interface.

You can also use **Validation Rule Template** – Extension for Visual Studio.

Properties

ValidatableObject<T> Class

```
...
public ValidatableObject<string> Email { get; set; } = new ValidatableObject<string>();
...
```

Once you have your rules, you need to create the properties you want to validate. Those properties must be a ValidatableObject<T>. Your Properties don't need to implement INotifyPropertyChanged interface. Plugin.ValidationRules has his own implementation called ExtendedPropertyChanged. So, the plugin do the job for you.

Adding Validations

Where the magic happens

```
...
public MyViewModel()
{
   ...
   Email.Validations.Add(new IsNotNullOrEmptyRule { ValidationMessage = "An email is required." });
   Email.Validations.Add(new EmailRule());
   ...
}
...
```

Before adding a validation rule to a property, be sure that you initialize the property. After that, add as many rules as you want to a property.

Validating properties

Where the magic happens

```
...
var isValidEmail = Email.Validate();
...
```

To validate a property, just call the **myProperty.Validate()** method. You can do it manually and use it in you ViewModel or code behind. After that, you can also check is the property is valid with **myProperty.IsValid** property.

```
...
var isValidEmail = Email.IsValid;
...
```

Validating properties

Where the magic happens

```
...
private bool ValidationMethod()
{
     ...
     var isValidName      = Name.Validate();
     var isValidLastname = LastName.Validate();
     var isValidEmail     = Email.Validate();
     ...
     // Your logic here
     ...
     return isValidName && isValidLastname && isValidEmail;
}
...
```

If you don’t want to validate each property one by one; create a **method** for validating all properties at the same time for you.

```
...
var unit = new ValidationUnit(Name, LastName, Email);
var isValidUnit = unit.Validate();
...
```

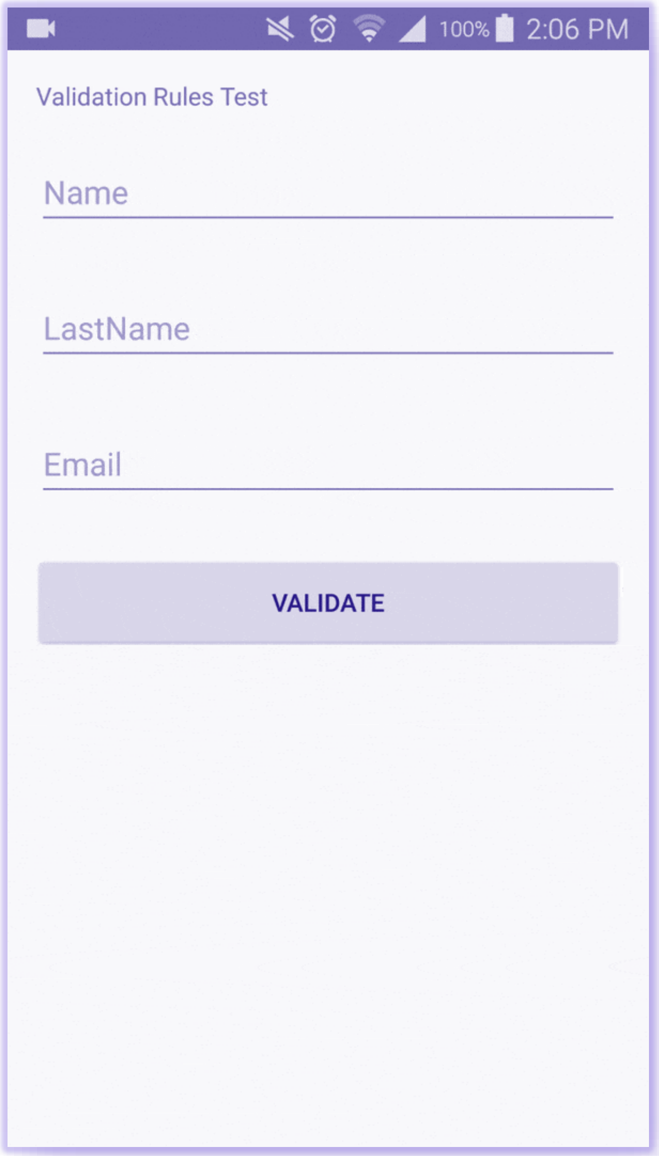
Or you can create a **ValidationUnit** property and passing all property, which you want to validate at the same time, by the constructor. After that, you can call **yourUnit.Validate()** method to validate yours properties

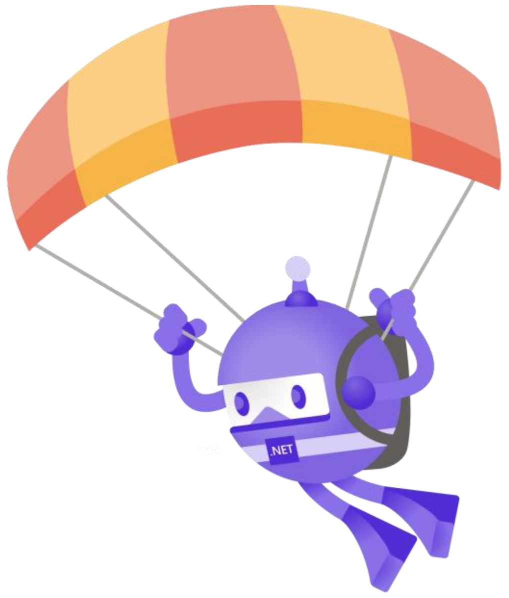
Displaying results

Where the magic happens

```
...
<Entry Text="{Binding Name.Value, Mode=TwoWay}" />
<Label Text="{Binding Name.Error}"
       TextColor="Red"
       HorizontalTextAlignment="Center" />
...
```

To bind your properties and errors to your XAML file; do it in the following way.





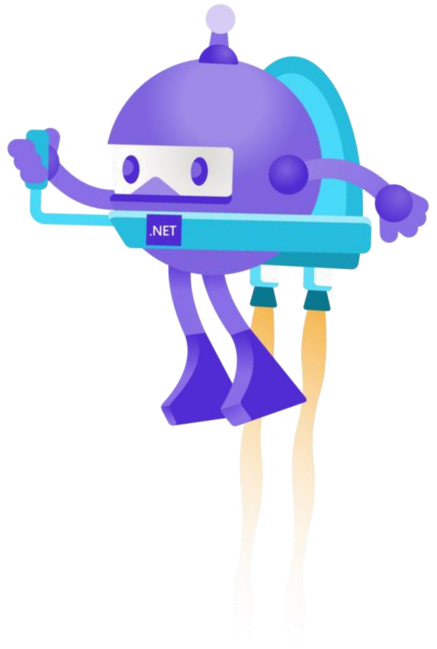
Demo

Plugin.ValidationRules

Plugins, extensions, and much more

Good libraries available for best practices

- Plugin.ValidationRules [github.com/luismts/ValidationRulesPlugin]
- Validation Rule Template Extension [github.com/luismts/ValidationRulesPlugin]
- FluentValidation [github.com/FluentValidation/FluentValidation]
- FluentValidation Plugin [github.com/mzhukovs/FluentValidationRulesPlugin]
- ReactiveUI.Validation [github.com/reactiveui/ReactiveUI.Validation]
- [your favorite goes here] 😊



Thanks for joining!

@luismatosluna