Custom Validator with Parameters in Angular

1 Comment / 3 minutes of reading / March 9, 2023

← Custom Validator in Reactive Forms
Angular Tutorial
Inject Service to Validator →

Learn how to create a custom validator with parameters in Angular Reactive Forms. This is the continuation of our previous tutorial, where we learned how to build a custom validator in Reactive forms.

Table of Contents

Custom Validator with Parameter

Here is the code of greater than validator (`gte`) from the Custom Validators in Angular Reactive Form tutorial. The validator checks if the given value is greater than 10 and if not return `ValidationErrors` .

```
export function gte(control: AbstractControl): ValidationErrors | null {

  const v:number=+control.value;

  if (isNaN(v)) {
    return { 'gte': true, 'requiredValue': 10 }
  }

  if (v <= 10) {
    return { 'gte': true, 'requiredValue': 10 }
  }

  return null

}

```

The problem with the above validator is that the value 10 is hardcoded. Hence, we will be not able to reuse it. If we want to resue it, we need to pass the number as the parameter.

Let us add the parameter `val:number` to the validator as shown below.

```

export function gte(control: AbstractControl,val:number): ValidationErrors | null {

```

The compiler immediately throws an error as shown below.

```

error TS2345: Argument of type '((control: AbstractControl, val: number) => ValidationError

```

That is because, the `Validator` must implement ValidatorFn Interface. It can have only one parameter i.e `control: AbstractControl`

```

interface ValidatorFn {
  (control: AbstractControl): ValidationErrors | null
}

```

Passing Parameters to a Custom Validator

To pass a parameter, we need to create a factory function or a function that returns a function. The example code is as shown below

```

export function gte(val: number): ValidatorFn {

  return (control: AbstractControl): ValidationErrors | null => {

    let v: number = +control.value;

    if (isNaN(v)) {
      return { 'gte': true, 'requiredValue': val }
    }

    if (v <= +val) {
      return { 'gte': true, 'requiredValue': val }
    }

    return null;

  }

}

```

First, we create a factory function. It receives the val as the argument. It must return the function of the type ValidatorFn

```

export function gte(val: number): ValidatorFn {

```

The get must return a function ValidatorFn

```

return (control: AbstractControl): ValidationErrors | null => {
   //Validaton code here
}

```

Using the Validator

Now, add the validator to the Validator collection of the FormControl as shown below

```

  myForm = new FormGroup({
    numVal: new FormControl('', [gte(10)]),
  })

```

Accessing the Errors in Template

```html

<h1>Custom Validator with Parameter in Angular</h1>

<h2>Reactive Form</h2>

<form [formGroup]="myForm" (ngSubmit)="onSubmit()" novalidate>

  <div>
    <label for="numVal">Number :</label>
    <input type="text" id="numVal" name="numVal" formControlName="numVal">
    <div *ngIf="!numVal.valid && (numVal.dirty ||numVal.touched)">
      <div *ngIf="numVal.errors.gte">
       The number should be greater than {{numVal.errors.requiredValue}}
      </div>
    </div>

  </div>


  <p>Is Form Valid : {{myForm.valid}} </p>

  <p>
    <button type="submit" [disabled]="!myForm.valid">Submit</button>
  </p>


</form>

```

Summary

We learned how to pass a parameter to a custom validator. First, we create a factory function, which accepts the parameter. The factory function returns the validator function. Using this technique we can pass as many as parameters to a custom validator in Angular.

← Custom Validator in Reactive Forms

Angular Tutorial

Inject Service to Validator →