Cross Field or Multi Field Validation Angular

3 Comments / 3 minutes of reading / March 9, 2023

← Angular Async Validator Angular Tutorial SetValidators →

In this article, we will learn how to implement Cross Field validation or mult field Validation in Angular. We learned how to validate reactive forms & how to create a custom validator. Those articles showed how to validate a Single Form Control. But some times we also come across fields whose value depends on another field. For example, the following scenario's requires us to compare two fields.

1. Start date & end date fields. The end date must be greater than the start date.
2. Password confirmation. The new password must match the confirm password.

Table of Contents

Validation Recap

We assign a validator's to a form filed, using the second argument of the FormControl as shown below. You can also attach an Async Validator as the third argument.

```

this.contactForm = new FormGroup({
   userName: new FormControl('',[Validators.required,customValidator]),

```

The above syntax using the FormBuilder.

```

this.contactForm = this.builder.group({
     userName: ["", [Validators.required,customValidator]],

```

The Validator will run only when we change the value of `userName` and Validates only the `userName` field.

Cross Field Validation

When we validate the multiple fields, we need to ensure that our validation logic runs for each of those fields.

Hence we attach the validator to the Formgroup instead of FormControl. The Validator runs whenever we modify any of the fields in the `FormGroup`.

Example

Let us create a `matchPassword` custom validator to compare the password & confirm Password fields.

Since we attach it to a FormGroup, it gets the instance of `FormGroup` as its parameter. We can use the `get` method to get the values of both password & confirm FormControls. If they do not match then return the `ValidationErrors`. Return `null` if it values passes the Validation.

```

matchPassword(control: AbstractControl): ValidationErrors | null {

  const password = control.get("password").value;
  const confirm = control.get("confirm").value;


  if (password != confirm) { return { 'noMatch': true } }

  return null

}

```

We attach the matchPassword Validator to FormGroup using its second argument as shown below.

```

    this.mainForm = this.builder.group({
      userName: ["", [Validators.required]],
      password: ["", [Validators.required, Validators.minLength(5)]],
      confirm: ["", [Validators.required]]
    }, { validator: this.matchPassword });

```

The FormGroup also allows us the add more than one validator using the Validators.compose method.

```

  this.mainForm = this.builder.group({
    userName: ["", [Validators.required]],
    password: ["", [Validators.required, Validators.minLength(5)]],
    confirm: ["", [Validators.required]]
  }, {
    validator: Validators.compose(
      [
        this.matchPassword,
        Validators.required
      ]
    )
  });
```

Passing Parameter

You can also pass the parameter to the Multiple Field Validator.

In the following example, we pass the name of the

```

this.mainForm = this.builder.group({
    userName: ["", [Validators.required]],
    password: ["", [Validators.required, Validators.minLength(5)]],
    confirm: ["", [Validators.required]]
  }, { validator: this.matchPassword2('password', 'confirm') });

```

```

matchPassword2(firstControl, secondControl): ValidatorFn {

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

    const password = control.get(firstControl).value;
    const confirm = control.get(secondControl).value;

    if (password != confirm) { return { 'noMatch': true } }

    return null

  }
}

```

Refer to the Custom Validator with Parameters in Angular. Also refer to the tutorial on how to inject service into a Validator.

Reference

1. Cross fields Validation