

Module 7: Validating Angular Forms

Demo Document 1 – Template driven user
registration form with validation

edureka!

edureka!

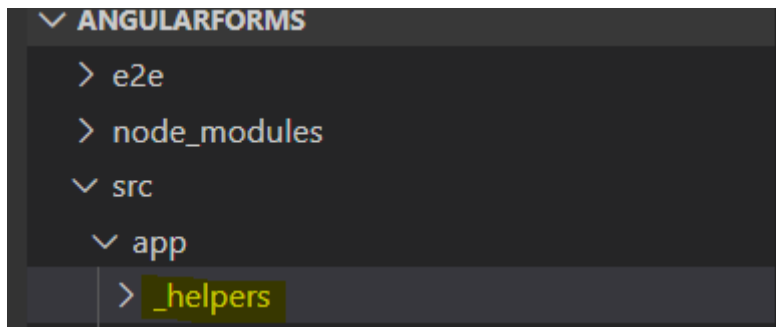
© Brain4ce Education Solutions Pvt. Ltd.

Creating Template driven user registration form and add validation to form

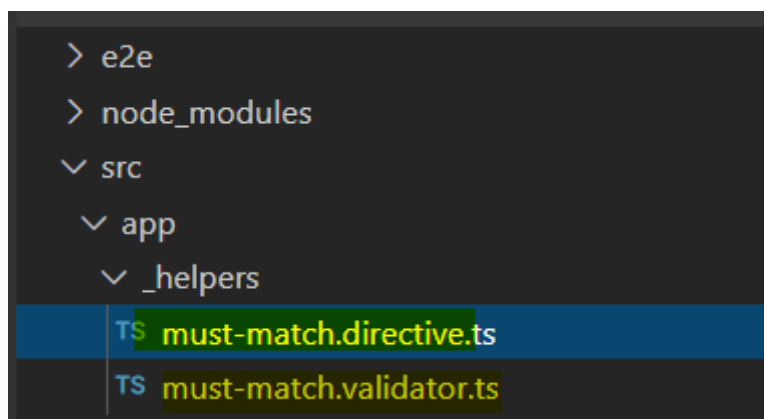
In this demo, we will see how to add validation for different fields for template driven form created in module 6 demo 1

Step 1: Open project Module6-Demo1, created in module 6 demo1

Step 2: Create “_helpers” folder under the app folder as shown below.



Step 3- Create must-match directive and must-match validator files as shown below.



Purpose of this is to validate fields in forms. It verifies if the password entered in the confirm password field matches the password entered in the password field. If you enter different password in password and confirm password fields then mustmatch directive will trigger error message.

Step 4 – The must-match.directive.ts contains the below code

```
import { Directive, Input } from '@angular/core';
import { NG_VALIDATORS, Validator, ValidationErrors, FormGroup } from '@angular/forms';

import { MustMatch } from './must-match.validator';

@Directive({
  selector: '[mustMatch]',
  providers: [{ provide: NG_VALIDATORS, useExisting: MustMatchDirective, multi: true }],
})
export class MustMatchDirective implements Validator {
  @Input('mustMatch') mustMatch: string[] = [];

  validate(formGroup: FormGroup): ValidationErrors {
    return MustMatch(this.mustMatch[0], this.mustMatch[1])(formGroup);
  }
}
```

Step 5 – The must-match.validator.ts contains the below code

```
import { FormGroup } from '@angular/forms';

// custom validator to check that two fields match
export function MustMatch(controlName: string, matchingControlName: string) {
  return (formGroup: FormGroup) => {
    const control = formGroup.controls[controlName];
    const matchingControl = formGroup.controls[matchingControlName];

    // return null if controls haven't initialised yet
    if (!control || !matchingControl) {
      return null;
    }

    // return null if another validator has already found an error on the matchingControl
    if (matchingControl.errors && !matchingControl.errors.mustMatch) {
      return null;
    }

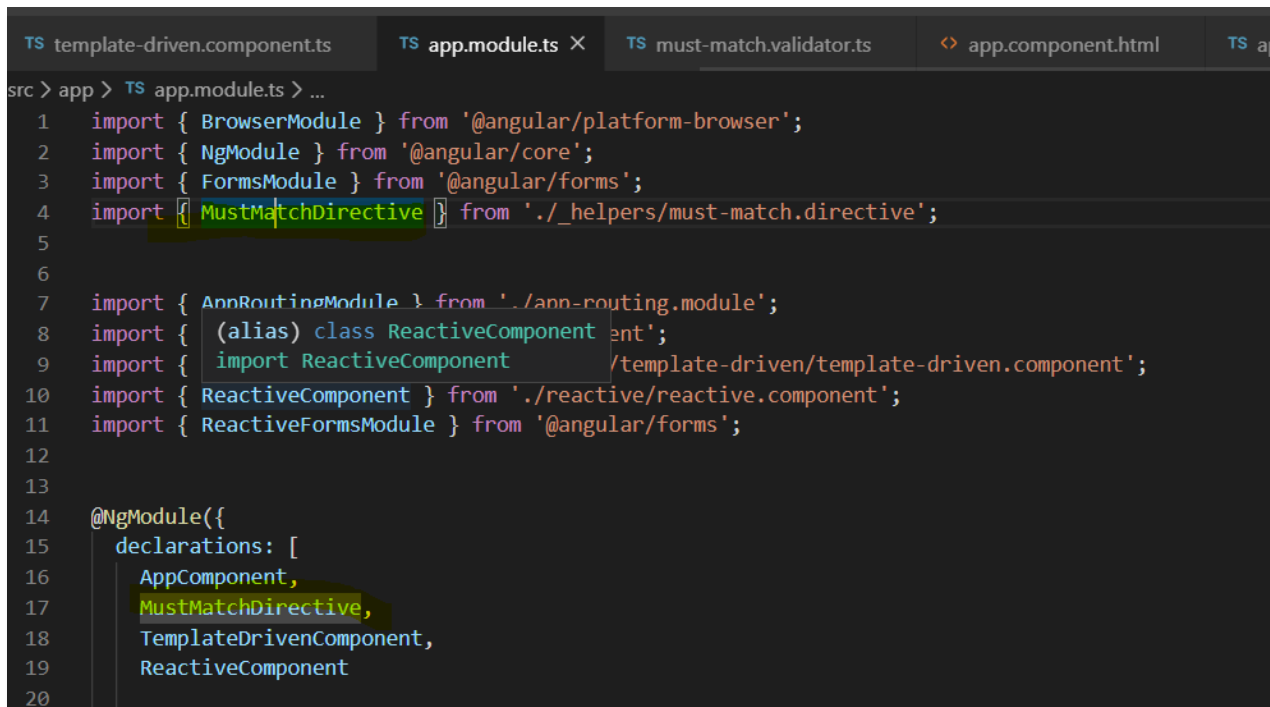
    // set error on matchingControl if validation fails
    if (control.value !== matchingControl.value) {
      matchingControl.setErrors({ mustMatch: true });
    }
  };
}
```

```

    } else {
        matchingControl.setErrors(null);
    }
}
}

```

Step 6- Open app.module.ts file and import mustmatch directive as below.



```

src > app > TS app.module.ts > ...
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { FormsModule } from '@angular/forms';
4  import { MustMatchDirective } from './_helpers/must-match.directive';
5
6
7  import { AppRoutingModule } from './app-routing.module';
8  import { (alias) class ReactiveComponent } from './reactive/reactive.component';
9  import { ReactiveComponent } from './template-driven/template-driven.component';
10 import { ReactiveComponent } from './reactive/reactive.component';
11 import { ReactiveFormsModule } from '@angular/forms';
12
13
14 @NgModule({
15   declarations: [
16     AppComponent,
17     MustMatchDirective,
18     TemplateDrivenComponent,
19     ReactiveComponent
20

```

Step 7- Open template-driven.component.ts file and create a “onSubmit()” method

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-template-driven',
  templateUrl: './template-driven.component.html',
  styleUrls: ['./template-driven.component.css']
})
export class TemplateDrivenComponent implements OnInit {

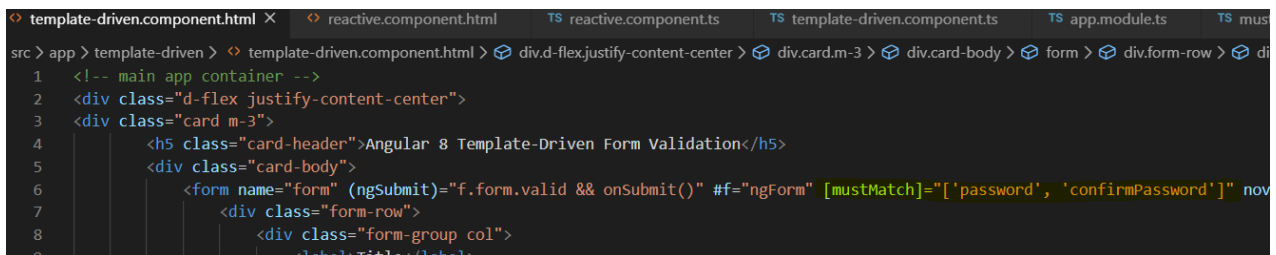
  model: any = {};
  onSubmit() {
    alert('SUCCESS!! :-)\n\n' + JSON.stringify(this.model, null, 4));
  }

  constructor() { }

  ngOnInit() {
  }
}
```

Step 8- Open template-driven.component.html and add built-in validators to validate the form fields.

Add mustmatch directive for password validation



```
src > app > template-driven > template-driven.component.html > div.d-flex.justify-content-center > div.card.m-3 > div.card-body > form > div.form-row > di
1 <!-- main app container -->
2 <div class="d-flex justify-content-center">
3 <div class="card m-3">
4   <h5 class="card-header">Angular 8 Template-Driven Form Validation</h5>
5   <div class="card-body">
6     <form name="form" (ngSubmit)="f.form.valid && onSubmit()" #f="ngForm" [mustMatch]="['password', 'confirmPassword']" nov
7     <div class="form-row">
8       <div class="form-group col">
9         <label>Title</label>
```

Note that we are adding all validation in .html file for template driven form

Step 9- Run app using ng serve

- This will open below window

The screenshot shows a registration form titled "Angular 8 Template-Driven Form Validation". It contains the following fields: "Title" (a dropdown menu), "First Name" (a text input), and "Last Name" (a text input). Below these is an "Email" text input field. Further down are "Password" and "Confirm Password" text input fields. At the bottom, there is a checkbox labeled "Accept Terms & Conditions". Two buttons, "Register" (in blue) and "Cancel" (in grey), are positioned at the bottom right of the form.

- Now click on register button, you will see that all validation messages will be triggered as below

This screenshot shows the same registration form as above, but with all validation errors triggered. Each input field is outlined in red and has a red 'x' icon in the top right corner. Below each field, a red error message is displayed: "Title is required" for the Title field, "First Name is required" for the First Name field, "Last Name is required" for the Last Name field, "Email is required" for the Email field, "Password is required" for the Password field, and "Confirm Password is required" for the Confirm Password field. The "Accept Terms & Conditions" checkbox is also unchecked, with the error message "Accept Ts & Cs is required" displayed below it. The "Register" and "Cancel" buttons remain at the bottom.

- Once you start adding values to fields validation errors will be removed

Angular 8 Template-Driven Form Validation

Title: Mr | First Name: sac | Last Name: ✖
Last Name is required

Email: ✖
Email is required

Password: ✖ Password is required

Confirm Password: ✖ Confirm Password is required

☐ Accept Terms & Conditions
Accept Ts & Cs is required

Register Cancel

- If you enter different password in password and confirm password fields then must-match directive will trigger error message

Angular 8 Template-Driven Form Validation

Title: Mr | First Name: sac | Last Name: s

Email: s@a.com

Password: | Confirm Password: ✖
Passwords must match

☒ Accept Terms & Conditions

Register Cancel

Step 10 – Once all errors are corrected and form is submitted it will show success popup as shown below

