Formulários - Templates forms

É uma forma declarativa de seus formulários no template do componente. Com template forms a gente usa as diretivas ngModel nos inputs que deve ser controlado pelo framework .

<form novalidate>: com o novalidate eliminamos a validação do formulário que pode ser exibida de maneiras diferentes dependendo do browser.

Trabalhando com formulário no angular:

<form #form=”ngForm” novalidate>

<iniput name=”endereço” ngModel >

</form>

É possível acompanhar o que acontece com o formulário em tempo real, colocando-o dentro de template interpolation e usando o pipe json:

{{form.value | json]}

Validação de formulários:

A diretiva ngModel, disponibiliza alguns estados que podem ser verificados para dar feedback visual ao usuário:

Valid|invalid: diz respeito se o valor do campo está de acordo com as regras de validação ou não.

Pristine|Dirty: Pristine representa o estado inicial do campo ou do form, uma vez que o usuário digita no campo ele se torna Dirty e não volta mais

Touched|Untouched: Touched indica quando o usuário entra no campo, a diferença entre dirty e touched é que pro campo ficar com o estado dirty o usuário precisa modifica-lo.

Opções de validação que podem ser atribuídas à um campo, são:

Validotors-

Require: que específica se o campo é obrigatório.

Pattern – regex: que recebe um padrão de expressão regular.

Min e maxLength: recebe um número e confere se o valor está sendo respeitado.

Em todos esses casos o formulário só se tornará true, caso seja obedecido.

Adicionando uma classe dinamicamente:

[class.has-success]="iptAddress.valid && (iptAddress.dirty || iptAddress.touched)"

Nesse exemplo a classe has-success será adicionada caso o campo seja valido ou tocado ou diferente do seu estado inicial.

**CONTENT PROJECTION**

Content projection (projeção de conteúdo) permite inserir um DOM sombra no seu componente. Simplificando, se você desejar inserir elementos HTML ou outros componentes em um componente, no angular utilizamos <ng-content></ng-content> para fazer a projeção de conteúdo.

Ng Content: é uma tag similar ao rout outlet onde você marca uma região onde através de uma navegação você entra com um componente correspondente a esse caminho, o ng content também representa um espaço que vai entrar um conteúdo, esse conteúdo fica entre as tags do componente quando for usá-lo. Essa é a técnica conhecida como contente projection.

Content child: com ele é possível dizer qual é o elemento ou a diretiva que queremos pegar a referência, utilizado para recuperar valores dentro de um componente filho.

No lugar do input, é aonde vai o ngcontent:

<div class="col-sm-6 col-xs-12">

  <div class="form-group" [class.has-success]="input.valid && (input.dirty || input.touched)"

                          [class.has-error]="!input.valid && (input.dirty || input.touched)">

    <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> {{label}}</label>

    <!-- <input type="text" class="form-control"  name="address" minlength="5" #iptAddress="ngModel"  required ngModel placeholder="Endereço" autocomplete="off"> -->

    <ng-content></ng-content>

    <span class="help-block" \*ngIf="input.valid && (input.dirty || input.touched)"><i class="fa fa-check"></i>Ok</span>

    <span class="help-block" \*ngIf="!input.valid && (input.dirty || input.touched)"><i class="fa fa-remove"></i> {{errorMessage}}</span>

  </div>

</div>

É necessário criar uma referência ao ngModel e para isso é necessário importar o tipo:

import {NgModel} from '@angular/forms';

e o contente child:

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

Dentro do contente child podemos utilizar uma referência ao mapeamento ou uma diretiva, nesse caso iremos utilizar uma diretiva, forma de se declarar isso:

@ContentChild(NgModel) model: NgModel

Repare que a variável é o ’model’ NgModel o tipo.

Porém para chamar essa diretiva existe um momento correto, e para isso usamos uma interface chamada AfterContentInit.

import { Component, OnInit, Input, ContentChild, AfterContentInit } from '@angular/core';

export class InputComponent implements OnInit, AfterContentInit {

ngAfterContentInit(){

  }