File

`src/app/admin/components/add-hero/add-hero.component.ts`

Description

Component to add a hero

Implements

`OnInit`

Metadata

selector	`app-add-hero`
styleUrls	`./add-hero.component.scss`
templateUrl	`./add-hero.component.html`

Index

Constructor

```
constructor(_snackBar: MatSnackBar, formBuilder: FormBuilder, _heroService: HeroService)
```
Defined in src/app/admin/components/add-hero/add-hero.component.ts:35

Constructor in which we inject our services and different elements

Parameters :

Name	Type	Optional
_snackBar	`MatSnackBar`	No
formBuilder	`FormBuilder`	No
_heroService	`HeroService`	No

Methods

getErrorMessage

`getErrorMessage(dato)`

Defined in src/app/admin/components/add-hero/add-hero.component.ts:155

function to control error messages

Parameters :

Name	Optional
dato	No

Returns : `"This information is required" | "The maximum of characters is 30" | "" | "You must enter at leas...`

message

ngOnInit

`ngOnInit()`

Defined in src/app/admin/components/add-hero/add-hero.component.ts:49

Start when the component inits

Returns : `void`

openSnackBar

`openSnackBar(message: string, action: string)`

Defined in src/app/admin/components/add-hero/add-hero.component.ts:201

function to open snackBars

Parameters :

Name	Type	Optional
message	`string`	No
action	`string`	No

Returns : `void`

saveHero

`saveHero()`

Defined in src/app/admin/components/add-hero/add-hero.component.ts:178

Function to save a hero

Returns : `void`

Properties

Public correctData

Type : `boolean`

Defined in src/app/admin/components/add-hero/add-hero.component.ts:35

variable to check if the function was ok

Public hero

Type : `Hero`

Defined in src/app/admin/components/add-hero/add-hero.component.ts:27

variable to store the hero to add

Public message

Type : `string`

Defined in src/app/admin/components/add-hero/add-hero.component.ts:31

variable to save message info

Public newHeroForm

Type : `FormGroup`

Defined in src/app/admin/components/add-hero/add-hero.component.ts:23

to add FormGroup

```
import { Component, OnInit } from '@angular/core'
import { FormBuilder } from '@angular/forms'
import { Validators } from '@angular/forms'
import { FormGroup, FormControl, AbstractControl } from '@angular/forms'
import { Hero } from 'src/app/models/hero'
import { HeroService } from 'src/app/services/hero.service'
import { DateAdapter } from '@angular/material/core'
import { MatSnackBar } from '@angular/material';

/**
 * Component to add a hero
 */
@Component({
  selector: 'app-add-hero',
  templateUrl: './add-hero.component.html',
  styleUrls: ['./add-hero.component.scss']
})

export class AddHeroComponent implements OnInit {
  /**
   * to add FormGroup
   */
  public newHeroForm: FormGroup
  /**
   * variable to store the hero to add
   */
  public hero: Hero
  /**
   * variable to save message info
   */
  public message: string
  /**
   * variable to check if the function was ok
   */
  public correctData: boolean

  /**
   * Constructor in which we inject our services and different elements
   */
  constructor(
    private _snackBar: MatSnackBar,
    private formBuilder: FormBuilder,
    private _heroService: HeroService
  ) { }

  /**
   * Start when the component inits
   */
  ngOnInit() {
    this.correctData = true;
    this.hero = new Hero(0, '', '', 0, 0, 0, 0, 0, 0, '', '', '', '', '', '', '', '', '', '', '', '', null)

    this.newHeroForm = this.formBuilder.group({
      heroName: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      image: [
        '',
        [
          Validators.required,
          Validators.minLength(1),
          Validators.maxLength(300)
        ]
      ],
      intelligence: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      strength: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      speed: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      durability: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      power: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      combat: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      fullName: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      placeOfBirth: ['', [Validators.required, Validators.minLength(1), Validators.maxLength(30)]],
      publisher: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      alignment: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      firstApperance: [
        '',
        [
          Validators.required,
          Validators.minLength(1),
          Validators.maxLength(300)
        ]
      ],
      gender: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      race: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      height: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      weight: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      eyeColor: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      hairColor: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(30)]
      ],
      work: [
        '',
        [Validators.required, Validators.minLength(1), Validators.maxLength(300)]
      ],
      biography: [
        '',
        [
          Validators.required,
          Validators.minLength(1),
          Validators.maxLength(300)
        ]
      ]
    })
  }

  /**
```

```
   * function to control error messages
   * @param {string} dato
   * @returns message
   */
  getErrorMessage(dato) {
    var result: string
    if (this.newHeroForm.controls[dato].hasError('required')) {
      return (result = 'This information is required')
    } else if (this.newHeroForm.controls[dato].hasError('minlength')) {

      return (result = 'You must enter at least 1 characters')

    } else if (this.newHeroForm.controls[dato].hasError('maxlength')) {
      if (dato === 'image' || dato === 'firstApperance' || dato === 'biography' || dato === 'work') {
        return (result = 'The maximum of characters is 300')
      } else {

        return (result = 'The maximum of characters is 30')
      }
    } else {
      return (result = '')
    }
  }

  /**
   * Function to save a hero
   */
  saveHero() {
    this._heroService.newHero(this.hero).subscribe(
      response => {
        this.message = 'Create correctly';
        this.correctData = true;
        this.openSnackBar('CREATE CORRECTLY', 'Close')

      },
      error => {
        this.correctData = false;
        console.log(error.status);
        this.message = 'create hero failed';
        console.log(this.message);
        this.openSnackBar('CREATE HERO FAILED', 'Close')
      }
    );
  }

  /**
   * function to open snackBars
   *  @param {string} message
   *  @param {string} action
   */
  openSnackBar(message: string, action: string) {
    this._snackBar.open(message, action, {
      duration: 8000,
      panelClass: ['blue-snackbar']
    })
  }

}

<div class="adminDetail">
  <div id="form">
    <div class="title">
      <h1>Add New Hero</h1>
    </div>

    <form [formGroup]="newHeroForm" (ngSubmit)="saveHero()">
      <mat-form-field appearance="legacy">
        <mat-label>heroName</mat-label>
        <input
          matInput
          type="text"
          formControlName="heroName"
          placeholder="heroName"
          [(ngModel)]="hero.heroName"
        />
        <mat-error *ngIf="!newHeroForm.controls['heroName'].valid">{{
          getErrorMessage("heroName")
        }}</mat-error>
      </mat-form-field>

      <mat-form-field appearance="legacy">
        <mat-label>image</mat-label>
        <input
          matInput
          type="text"
          formControlName="image"
          placeholder="image"
    [(ngModel)]="hero.image"
  />
  <mat-error *ngIf="!newHeroForm.controls['image'].valid">{{
    getErrorMessage("image")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>fullName</mat-label>
  <input
    matInput
    type="text"
    formControlName="fullName"
    placeholder="fullName"
    [(ngModel)]="hero.fullName"
  />
  <mat-error *ngIf="!newHeroForm.controls['fullName'].valid">{{
    getErrorMessage("fullName")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>placeOfBirth</mat-label>
  <input
    matInput
    type="text"
    formControlName="placeOfBirth"
    placeholder="placeOfBirth"
    [(ngModel)]="hero.placeOfBirth"
  />
  <mat-error *ngIf="!newHeroForm.controls['placeOfBirth'].valid">{{
    getErrorMessage("placeOfBirth")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>publisher</mat-label>
  <input
    matInput
    type="text"
    formControlName="publisher"
    placeholder="publisher"
    [(ngModel)]="hero.publisher"
  />
  <mat-error *ngIf="!newHeroForm.controls['publisher'].valid">{{
    getErrorMessage("publisher")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>alignment</mat-label>
  <input
    matInput
    type="text"
    formControlName="alignment"
    placeholder="alignment"
    [(ngModel)]="hero.alignment"
  />
  <mat-error *ngIf="!newHeroForm.controls['alignment'].valid">{{
    getErrorMessage("alignment")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>firstApperance</mat-label>
  <input
    matInput
    type="text"
    formControlName="firstApperance"
    placeholder="firstApperance"
    [(ngModel)]="hero.firstApperance"
  />
  <mat-error *ngIf="!newHeroForm.controls['firstApperance'].valid">{{
    getErrorMessage("firstApperance")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>gender</mat-label>
  <input
    matInput
    type="text"
    formControlName="gender"
    placeholder="gender"
    [(ngModel)]="hero.gender"
  />
  <mat-error *ngIf="!newHeroForm.controls['gender'].valid">{{
    getErrorMessage("gender")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>race</mat-label>
  <input
    matInput
    type="text"
    formControlName="race"
    placeholder="race"
    [(ngModel)]="hero.race"
  />
  <mat-error *ngIf="!newHeroForm.controls['race'].valid">{{
    getErrorMessage("race")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>height</mat-label>
  <input
    matInput
    type="text"
    formControlName="height"
    placeholder="height"
    [(ngModel)]="hero.height"
  />
  <mat-error *ngIf="!newHeroForm.controls['height'].valid">{{
    getErrorMessage("height")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>weight</mat-label>
  <input
    matInput
    type="text"
    formControlName="weight"
    placeholder="weight"
    [(ngModel)]="hero.weight"
  />
  <mat-error *ngIf="!newHeroForm.controls['weight'].valid">{{
    getErrorMessage("weight")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label> eyeColor</mat-label>
  <input
    matInput
    type="text"
    formControlName="eyeColor"
    placeholder="eyeColor"
    [(ngModel)]="hero.eyeColor"
  />
  <mat-error *ngIf="!newHeroForm.controls['eyeColor'].valid">{{
    getErrorMessage("eyeColor")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>hairColor</mat-label>
  <input
    matInput
    type="text"
    formControlName="hairColor"
    placeholder="hairColor"
    [(ngModel)]="hero.hairColor"
  />
  <mat-error *ngIf="!newHeroForm.controls['hairColor'].valid">{{
    getErrorMessage("hairColor")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>work</mat-label>
  <input
    matInput
    type="text"
    formControlName="work"
    placeholder="work"
    [(ngModel)]="hero.work"
  />
  <mat-error *ngIf="!newHeroForm.controls['work'].valid">{{
    getErrorMessage("work")
  }}</mat-error>
</mat-form-field>
```

```
<mat-form-field appearance="legacy">
  <mat-label>biography</mat-label>
  <input
    matInput
    type="text"
    formControlName="biography"
    placeholder="biography"
    [(ngModel)]="hero.biography"
  />
  <mat-error *ngIf="!newHeroForm.controls['biography'].valid">{{
    getErrorMessage("biography")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>intelligence</mat-label>
  <input
    matInput
    type="number"
    placeholder="intelligence"
    [(ngModel)]="hero.intelligence"
    formControlName="intelligence"
    min="1"
    max="100"
  />
  <span matSuffix></span>
  <mat-error *ngIf="!newHeroForm.controls['intelligence'].valid">{{
    getErrorMessage("intelligence")
  }}</mat-error>
</mat-form-field>
<mat-form-field appearance="legacy">
  <mat-label>strength</mat-label>
  <input
    matInput
    type="number"
    placeholder="strength"
    [(ngModel)]="hero.strength"
    formControlName="strength"
    min="1"
    max="100"
  />
  <span matSuffix></span>
  <mat-error *ngIf="!newHeroForm.controls['strength'].valid">{{
    getErrorMessage("strength")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>speed</mat-label>
  <input
    matInput
    type="number"
    placeholder="speed"
    [(ngModel)]="hero.speed"
    formControlName="speed"
    min="1"
    max="100"
  />
  <span matSuffix></span>
  <mat-error *ngIf="!newHeroForm.controls['speed'].valid">{{
    getErrorMessage("speed")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>durability</mat-label>
  <input
    matInput
    type="number"
    placeholder="durability"
    [(ngModel)]="hero.durability"
    formControlName="durability"
    min="1"
    max="100"
  />
  <span matSuffix></span>
  <mat-error *ngIf="!newHeroForm.controls['durability'].valid">{{
    getErrorMessage("durability")
  }}</mat-error>
</mat-form-field>

<mat-form-field appearance="legacy">
  <mat-label>power</mat-label>
  <input
    matInput
    type="number"
```

```
          placeholder="power"
          [(ngModel)]="hero.power"
          formControlName="power"
          min="1"
          max="100"
        />
        <span matSuffix></span>
        <mat-error *ngIf="!newHeroForm.controls['power'].valid">{{
          getErrorMessage("power")
        }}</mat-error>
      </mat-form-field>

      <mat-form-field appearance="legacy">
        <mat-label>combat</mat-label>
        <input
          matInput
          type="number"
          placeholder="combat"
          [(ngModel)]="hero.combat"
          formControlName="combat"
          min="1"
          max="100"
        />
        <span matSuffix></span>
        <mat-error *ngIf="!newHeroForm.controls['combat'].valid">{{
          getErrorMessage("combat")
        }}</mat-error>
      </mat-form-field>

      <div *ngIf="!correctData">
        <p class="mensajeError">{{ this.message }}</p>
      </div>
      <br />

      <button type="submit" class="modifyHero" [disabled]="!newHeroForm.valid">
        ADD NEW HERO
      </button>
    </form>
  </div>
</div>
```

./add-hero.component.scss

```
.adminDetail {
  border: 2.5px solid #00a23d;
  box-shadow: 0 0 21px 9px rgba(8, 17, 10, 0.63);
  padding: 0.5em;
  margin: 0.5em;
  background-color: #272727;
  color: white;
  margin-bottom: 3em;
}

#form .mat-form-field {
  margin-left: 12px;
}

#form ::ng-deep .mat-form-field-appearance-legacy .mat-form-field-label {
  color: #709e82;
}

.modifyHero {
  outline: none;
  background-color: #f3d403;
  border: none;
  border-radius: 10px;
  border: 2.5px solid #00a23d;
  padding: 9px;
  font-size: 0.8rem;
  color: black;
}

.modifyHero:hover {
  background-color: #cc4224;
  color: rgb(223, 221, 221);
}

.title {
  display: flex;
  flex-direction: column;
}

@media (min-width: 992px) {
  .title {
    display: flex;
    flex-direction: row;
  }

  .adminDetail {
    margin: 3em;
    padding: 2em;
  }
}
```

Legend
Html element
Component
Html element with directive

result-matching ""

No results matching ""