Lab 6: Services

Tot nu toe hebben we de data altijd rechtstreeks in de components gedeclareerd. Stel dat we dezelfde data nodig hebben in verschillende componenten, geeft dit problemen. Daarom dat we services aanmaken die in staat voor het aanleveren van data.

Contact service maken

Maak een nieuwe folder in **/src/app** met de naam **services**. In deze folder maak je een nieuw bestand met de naam **contact.service.ts**. Voorlopig gaat onze service de contactList array bevatten samen met een getter om de list door te geven, contacts toe te voegen en om contacts te verwijderen.

Voorzie onderstaande code in de contact.service.ts file:

```
import { Injectable } from '@angular/core';
import { Contact } from '../models/contact.model';

@Injectable()
export class ContactService {
    contactList: Contact[] = [
        new Contact('jane doe', 'jane.doe@mail.com', '0113448239',
true, 'assets/avatar.png'),
        new Contact('john doe', 'john.doe@mail.com', '011424839',
false, 'assets/avatar.png'),
        new Contact('Dries Swinnen', 'dries.swinnen@pxl.be',
'011664839', true, 'assets/avatar.png')
    ];

    constructor() { }

    getContactList(): Contact[] {
        return this.contactList;
    }

    addContact(contact: Contact): void {
        this.contactList.push(contact);
```

```
    }

    toggleFavorite(index: number): void {
        this.contactList[index].isFavorite =
!this.contactList[index].isFavorite;
    }
}
```

De ContactService gebruiken

De service is aangemaakt en kan gebruikt worden doorheen onze applicatie. Dit doen we aan de hand van dependency injection. Dit is de manier waarop Angular instanties van jouw services maakt. We kunnen deze instantie van de service **per component** toevoegen, of éénmalig op **globaal niveau**. Wij maken gebruik van de globale instantie. Hiervoor moet je de service toevoegen aan de **providers** in de **app.module.ts:**

```
import { ContactService } from './services/contact.service';

@NgModule({
  providers: [ContactService],
})
```

Vervolgens gaan we naar onze app.component.ts file. Hierin mag je de initialisatie van de contactList verwijderen. De declaratie blijft bestaan.

Hierna voorzie je import van de contactService, en voegen we een referentie toe via de constructor van de component:

```
import { Contact } from './models/contact.model';
import { Component, OnInit } from '@angular/core';
import { ContactService } from './services/contact.service';

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

```
  contactList: Contact[];


  constructor(private service: ContactService) { }
  ngOnInit(): void {
  }
}
```

In de ngOnInit methode voorzien we nu de initialisatie van de contactList aan de hand van de service:

```
  ngOnInit(): void {
    this.contactList = this.service.getContactList();
  }
```

De methode die instaat voor het afhandelen van het formulier voor het toevoegen van contacten, moet ook aangepast worden:

```
  createContact(event: Contact) {
    this.service.addContact(event);
    this.contactList = this.service.getContactList();
  }
```

De bestaande functionaliteiten zijn reeds overgenomen en de applicatie werkt nu met de service. Er is in de service echter ook een toggleFavorite methode voorzien de we nog niet gebruiken. Om deze methode te implementeren, moeten we nog enkele aanpassingen maken.

In de **app.component.html** geven we een extra variabele door naar de contact component, namelijk de index van het huidige item. De app component moet vervolgens ook doorkrijgen of er updates doorgevoerd zijn, zodat de lijst geüpdatet kan worden.

App.component.html

```
<app-contact *ngFor="let contact of contactList; let i = index"
             [contact]="contact" [index]="i"
             (onUpdate)="handleUpdate($event)">
</app-contact>
```

App.component.ts

```
handleUpdate(): void {
    this.contactList = this.service.getContactList();
  }
```

De index variabele moet nog opgevangen worden in de app-contact component. Daarnaast moeten we ook nog de onUpdate methode voorzien om de parent door te geven dat de content geüpdatet is. De eventEmitter die we hiervoor gebruiken gaat geen data teruggeven, dus we gebruiken hier het datatype any.

Voeg volgende properties toe aan contact.component.ts:

```
export class ContactComponent implements OnInit {
    @Input() contact: Contact;
    @Input() index: number;
    @Output() onUpdate: EventEmitter<any> = new EventEmitter();
```

Daarnaast moet de service ook toegevoegd worden aan de contact component. Voeg deze toe via dependency injection:

```
import {ContactService} from '../services/contact.service';

constructor(private service: ContactService) { }
```

Nu kan de view aangepast worden om het geheel aan elkaar te koppelen. Voorzie volgende aanpassingen in contact.component.html:

```
<input type="checkbox" name="isFavorite" id="isFavorite"
[(ngModel)]="contact.isFavorite" (click)="toggleFavorite(index)">
    <label for="isFavorite"
[class.favorite]="contact.isFavorite">Favorite</label>
```

Voeg de methode toggleFavorite toe aan contact.component.ts:

```
toggleFavorite(index: number): void {
        this.service.toggleFavorite(index);
        this.onUpdate.emit();
    }
```

Nu wordt het aanpassen van het favorite veld automatisch aangepast via de service en wordt de lijst geüpdatet bij een aanpassing.