

Labo Gebruikersinterfaces

Reeks 5: Angular

17-25 maart 2020

1 Inleiding

- Angular is een front end framework om complexe web apps te ontwikkelen.
- Angular is een volledige rewrite van Angular.js. Angular wordt ook Angular2 of Angular2+ genoemd. Wij gebruiken hier versie 9. Angular.js en Angular zijn niet compatibel. Als je iets opzoekt over Angular moet je daarom zeker zijn dat je niet bij Angular.js documentatie terecht komt.
- Angular gebruikt Typescript, een superset van ES6. Met Typescript heb je static typing binnen Javascript. Typescript wordt dan gecompileerd naar Javascript.
- Indien je je laptop gebruikt, zorg je dat je volgende zaken geïnstalleerd hebt:

1. NodeJS en npm: <https://nodejs.org/en/download/>

2. Angular CLI:

```
npm install -g @angular/cli
```

see <https://angular.io/guide/quickstart>

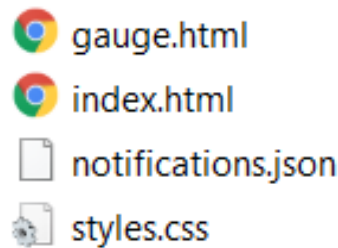
2 Getting started

1. Open een command line venster en navigeer naar de map waarin je het angular-project wil aanmaken (commando `cd`).
2. Maak een nieuw angular project aan:

```
ng new smarthome --routing
```
3. Kies voor CSS als stylesheet format.
4. Ga nu naar de aangemaakte map *smarthome*.
5. Voer volgend commando's uit om het project te compileren en de webserver te starten:

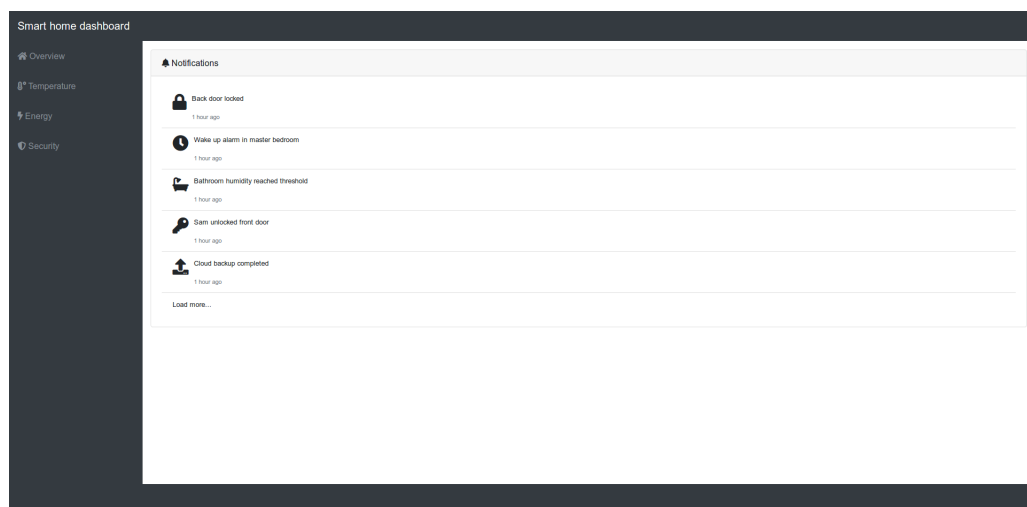
```
ng serve
```

6. Open de browser en ga naar <http://localhost:4200>.
7. Open het project in een IDE naar keuze (Webstorm, IntelliJ,...) en verken de structuur. We gebruiken de IDE enkel om de bestanden aan te passen. Het compileren van de TypeScript en het runnen van de server worden automatisch door “ng” gedaan als je de bestanden opslaat.
8. Waar staat de html die de pagina beschrijft ? Waar staat de CSS? Waar wordt de titel ingesteld ? Wat is het verschil tussen “index.html” en “app.component.html” ? Welke code schrijf je waar ?
9. Op Ufora vind je een aantal losse bestanden (verpakt in een zip).



Deze startcode bevat een statische versie van de applicatie die we gaan ontwikkelen; in dit labo passen we dit aan tot een dynamische angular applicatie. Verhuis de html code uit *index.html*: het gedeelte met de head-tag gaat naar *index.html*, de rest gaat naar *src/app/app.component.html*. (Verhuis de inline css-code gewoon mee.)

Tip: bij foutmeldingen controleer je of de link naar *styles.css* er nog staat (die mag weg), en of de code `<base href="/">` te vinden is in *index.html* (dat moet er wél staan).



3 Angular components

1. De overview pagina toont een lijst van notifications. Voorlopig zijn deze hard gecodeerd in *src/app/app.component.html* (met inline css).
2. We willen dat deze notifications dynamisch aan de pagina worden toegevoegd. We kunnen dit eenvoudig doen zoals in het vorige labo door via Javascript het DOM rechtstreeks te manipuleren maar Angular heeft een elegantere aanpak.

3. Maak een nieuwe component “notification” aan:

`ng generate component notification` (of korter: `ng g c notification`)

4. Binnen `src/app` is er nu een mapje “notification” met daarin:

- (a) `notification.component.html`: html template van de component
- (b) `notification.component.scss`: (s)css styling van de component
- (c) `notification.component.spec.ts`: unit tests voor de component
- (d) `notification.component.ts`: De javascript (typescript) code voor de component.

5. Verplaats de html-code voor één notification naar de `notification.component.html` file. Knip de andere notifications uit `app.component.html` en plak die in een voorlopig document voor later gebruik.

6. Op de plaats van de verdwenen notifications includeer je nu de nieuwe component:

```
<app-notification></app-notification>
```

7. Je zou één enkele notification moeten zien in de browser. Let ook eens op het verschil tussen de code die je gemaakt hebt, de source code die je ziet in je browser (Ctrl+U) en de DOM die je ziet met developer tools (F12).
8. De gegeven html code van de notification bevat inline css styling. Verplaats deze naar de `notification.component.css` file. Gebruik de juiste selector in de css-code. (Controleer door te bekijken in de browser!)
9. Pas de `component.ts` aan zodat elke component een boodschap, een icoon type en een timestamp heeft. Gebruik voorlopig een hardgecodeerde waarde. De timestamp initialiseer je met `new Date()`.
10. Gebruik deze variabelen uit `component.ts` in de html code (angular interpolation).
11. Verwijder nu de hardgecodeerde waarden voor de variabelen voor het bericht en de icon in `notification.component.ts` en plaats `@Input()` ervoor.

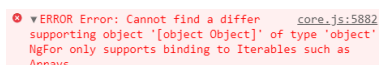
`@Input()` icon: string;

Dit laat ons toe om de waarde van dit attribuut in te stellen in de code van `app.component.html`:

```
<app-notification icon="fa-clock"></app-notification>
```

Tip: vergeet niet `Input` te importeren in `notification.component.ts`

12. Voeg de array “notifications” uit de gegeven startcode in “notifications.json” toe aan de AppComponent-klasse die je in `app.component.ts` vindt. Het is de bedoeling dat de AppComponent-klasse de eigenschap `notifications` heeft, waarvan de initialisatie gebeurt aan de hand van hardgecodeerde waarden.
13. Gebruik `*ngFor` om voor elk object in deze array een nieuwe notification component toe te voegen aan de pagina.
Krijg je deze foutmelding?



Controleer dan of je de array notifications juist geïntialiseerd hebt.

4 Angular routing

1. Maak nieuwe componenten voor de verschillende tabbladen (overview, temperature, energy en security). (Gebruik `ng generate`.)
2. Verplaats de html code die de lijst toont vanuit *app.component.html* naar de overview component. Vergeet ook de lijst met data niet te verplaatsen vanuit *app.component.ts* naar het overeenkomstig bestand bij de overview component. (Pak heel het stuk mee: vanaf de div-tag met attribuut `class="content-wrapper"`.)
3. We willen er nu voor zorgen dat je de verschillende tabbladen kan zien door te surfen naar “/overview”, “/security”, ...
4. Zorg ervoor dat de router-outlet tag in de *app.component.html* staat. (Zie onderaan, misschien staat die er al?)
5. Pas de routing configuratie aan zodat de urls gemapt worden op deze nieuwe componenten.
6. Controleer of je de juiste componenten te zien krijgt als je naar de urls surft. Pas de links in de navigatiebalk aan.
7. Er wordt nu steeds een refresh gedaan van de pagina als je naar een ander tabblad surft. Pas de links aan zodat ze *routerLink* gebruiken in plaats van een *href*. Hiervoor moet je wel de navigatiebalk verhuizen vanuit “index.html” naar een angular component, als dat nog niet gebeurd was. Waarom is dit het geval ?

5 2 way data binding

1. Maak een nieuwe component *TemperatureGauge* aan en voeg deze toe aan de *temperature* component.
2. Gebruik de html en css code vanuit het gegeven bestand *gauge.html* als view voor deze component.
3. Maak attributen aan voor de titel en de temperatuur in de corresponderende klasse en geef ze een defaultwaarde.
4. Zorg voor 2 way data binding van het input veld en het attribuut in de klasse. Gebruik normale property binding voor de andere temperatuur. Beide temperaturen gebruiken hetzelfde attribuut in de klasse. Vergeet niet om de “FormsModule” toe te voegen aan je *app.module.ts* bestand.

6 Event binding

1. Zorg ervoor dat je slechts de eerste 5 notifications toont.
2. Door op de knop onderaan de lijst te klikken kan je er 5 extra tonen.
3. Toon deze knop enkel als er nog notifications zijn die nog niet getoond worden.

7 Angular routing (deel 2)

1. Maak een nieuwe component *NotificationDetail* aan. Deze component zal de details tonen van één specifieke notification.
2. Definieer de route *notification/:id* die naar deze component gaat.
3. Gebruik de routerLink om deze route te volgen als je op een notification klikt.
4. Toon de id van de notification in de detail pagina.

8 Service

1. Om de applicatie nu af te werken gaan we de notifications ophalen via AJAX.
2. Hiervoor gaan we een service gebruiken. Een service is een Angular component die gebruikt kan worden vanuit verschillende andere componenten. Hier zal de service verantwoordelijk zijn voor het ophalen en bijhouden van de notifications.
3. Maak een nieuwe service aan en gebruik dependency injection om een referentie te hebben binnen de overview component.
4. Maak twee methodes aan in de service, één om alle notifications terug te geven en één om één specifieke terug te geven. Voorlopig kan je de notifications nog hard coderen in de service.
5. Pas de overview component aan zodat de data vanuit de service gebruikt wordt.
6. Pas ook de notification detail component aan zodat die de juiste notification toont.
7. Pas nu je service aan zodat de data niet meer hardgecodeerd staat maar dat ze dynamisch opgehaald wordt vanop <http://www.mocky.io/v2/5be453402f00002c00d9f48f>

