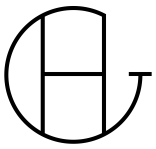

HoGent

Faculteit Bedrijf en Organisatie

Migratie naar Angular 2.0

Mathio Roelandt

Scriptie voorgedragen tot het bekomen van de graad van
Bachelor in de toegepaste informatica

Promotor:
Bert Van Vreckem
Co-promotor:
Joeri Van Steen

Instelling: —

Academiejaar: 2015-2016

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Migratie naar Angular 2.0

Mathio Roelandt

Scriptie voorgedragen tot het bekomen van de graad van
Bachelor in de toegepaste informatica

Promotor:
Bert Van Vreckem
Co-promotor:
Joeri Van Steen

Instelling: —

Academiejaar: 2015-2016

Tweede examenperiode

Samenvatting

Voorwoord

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Probleemstelling en Onderzoeksvragen

Angular 2 is nog maar sinds december 2015 in beta, om deze reden is er nog geen overvloed aan informatie beschikbaar. Als developer of bedrijf de opstap maken van Angular 1.x naar Angular 2.0 zal daarom dus niet vanzelfsprekend zijn. Er zijn reeds artikels die een speculatie maken over de migratie hiernaar, vooral in de weken na de aankondiging van Angular 2. Maar echte concrete of goed onderbouwde stappenplannen om een goede overgang te maken zijn er nog niet. Waarom zou een developer overstappen van Angular 1.x naar Angular 2.0, is de huidige Angular niet meer goed genoeg? Wat zijn de belangrijkste veranderingen, welke stappen zal hij/zijn moeten ondergaan om een naadloze overgang te maken en is het eigenlijk zelfs wel nuttig een overstap te maken met wat momenteel geweten is?

Hoofdstuk 2

Methodologie

Literatuurstudie
Wat is er momenteel geweten over Angular 2?

- Angular developer guide onderzoeken.
- Blogs bekijken van ervaren developers (eventueel van Angular team members zoals Victor Savkin, ...).
- Inhoudelijk bijstaan door vakexperts: Joeri Van Steen (lector webapplicaties Hogent), developers Array51.
- Eventueel interviews regelen met developers. Mogelijke vragen kunnen dan zijn: Zijn er al stappen gezet in het overgaan naar Angular 2, wordt er bij huidige projecten al rekening gehouden met Angular 2, ...

Vergelijking
Wat zijn de grote verschillen tussen angular 1.x en angular 2? De grootste veranderingen qua functionaliteit/performantie aankaarten en bespreken. Dit zal ik doen aan de hand van de developer guides van Angular.

Migratieplan
Aan de hand van alle informatie die ik bij vorige fasen verzameld heb zal ik een goed onderbouwd migratieplan maken. Eén enkel migratieplan zal echter nooit voldoende zijn omdat er zeer veel verschillende soorten projecten en organisaties zijn. Daarom zal ik ook hiermee rekening moeten houden. Moet je meteen overstappen of kan je in kleinere fasen werken?

Hoofdstuk 3

Waarom overstappen?

Angular 1.x werkt zoals in de probleemstelling vermeld werd nog steeds goed en zal de eerstkomende jaren zeker niet verdwijnen. Maar Angular 2 is gemaakt om beter te zijn dan Angular 1. Angular 1 zal de komende jaren dus zeker nog gebruikt kunnen worden om applicaties te ontwikkelen, met als grote reden dat mensen nog steeds oudere technologieën gebruiken bijvoorbeeld: IE8, oude versies van Android, ...

Voor ik overga naar hoe up te graden naar Angular 2 geef ik in dit hoofdstuk enkele redenen waarom Angular 2 het overwegen waard is.

Perfomantie Verschillende modules zullen verwijderd worden uit de Angular core, wat resulteert in beter performantie. Dit wil niet zeggen dat je deze modules niet meer kan gebruiken, je hoeft nadien enkel de modules die je nodig hebt te zoeken en gebruiken.

Betere mobile support Angular 1.x is niet gebouwd met mobile support in het achterhoofd, hoewel er wel enkele frameworks zoals Ionic hier een oplossing voor bieden. Angular 2 heeft uit deze fouten geleerd en meteen frameworks, zoals Ionic, geïmplementeerd wat een grote inpakt heeft op de user experience en performantie ervan.

Toekomst Angular 2 gebruikt alle nieuwe, veelbelovende features die momenteel beschikbaar zijn, zoals ES6, Typescript, WebComponents, ...

Hoofdstuk 4

Veranderingen in Angular 2.0

Een nieuwe versie brengt natuurlijk ook verandering met zich mee. In dit hoofdstuk bespreek ik de grootste veranderen tegenover Angular 1.x.

Typescript Typescript is een open source programmeer taal ontwikkeld door Microsoft. Microsoft werkte gedurende enkele maanden samen met het Angular team om Typescript te verbeteren. Resultaat van deze samenwerking was de nieuwe release van de taal, Typescript 1.5. Dit is bijgevolg ook de taal waarin Angular 2 gebouwd is. Hieraan zijn ook enkele voordelen verbonden.

- Type errors worden compile-time en runtime gegeven. Op die manier is het makkelijker je code te debuggen en te checken of je applicatie werkt.
- Betere auto complete door tekst editors en IDEs.
- De Javascript VM kan beter code optimizations maken.

Typescript is een superset van ES6, wat op zijn beurt een superset van ES5 is. Angular 2 wordt momenteel getranspiled naar ES5 om uitvoerbaar te zijn op moderne browsers. Developers kunnen natuurlijk kiezen in welke taal ze willen werken (ES5, ES6, Typescript, . . .).

Component based Angular 2 is volledig component based, dit wil zeggen dat zaken als controllers en directives niet meer gebruikt zullen worden. De controllers en directives zullen vervangen worden door componenten. Componenten zijn directives met een template die twee zaken bevatten, een selector en een @View. De selector correspondeert met de html tag waarmee de component wil communiceren en de @View specifieert een html template waarin de component gebruikt zal worden.

Een voorbeeld zal meer duidelijkheid schetsen.

Angular 1.x:

```
angular.module('voorbeeld')
        .controller('VoorbeeldCtrl', function() {});
```

Angular 2.0:

```
Import {Component, View} from 'angular/angular2';

@Component({
    selector: 'voorbeeld'
})

@View({
    templateUrl: './components/voorbeeld/voorbeeld.html'
})

Export class Voorbeeld {}
```

Dependency injection Hier komen alle verandering omtrent dependency injection.

$scope Angular 2 zal geen $scope meet bevatten. In plaats van properties in de scope te binden in onze templates kan je nu meteen naar properties binden van je componenten.

```
@Component({
  selector: 'sample-app',
  componentServices: [
    NameList
  ]
})
@Template({
  url: './templates/sample-app.html',
  directives: [Foreach]
})
class SampleApp {
  constructor() {
    this.names = NameList.get();
```

```
    this.newName = '';
  }
  addName(newname) {
    this.names.push(newname.value);
    newname.value = '';
  }
}
```

?Nu kunnen we meteen binden naar ?this.names? in onze template:

```
...
<ul>
  <li *foreach="#name in names"></li>
</ul>
...
```

Hoofdstuk 5

Angular 1.5

Voor ik inga op hoe je de overstap naar Angular 2.0 kan maken wil ik het eerst hebben over Angular 1.5 en waarom het handig kan zijn eerst de overstap hiernaar te maken. Angular 1.5 is namelijk gemaakt met Angular 2 in het achterhoofd. De grootste reden achter deze release is de migratie naar Angular 2 makkelijk ter maken. In deze release zitten features die het mogelijk maken voor developers om Angular 1 applicaties te maken die dichter aanleunen bij de structuur van Angular 2.

De grootste veranderingen zijn vooral gebaseerd rond componenten omdat dit ook een belangrijke verandering is in Angular 2. Maar Angular 1.5 houdt ook enkele andere features in om development te verbeteren, zoals betere support voor ES6 klassen, ngResource, ngRoute, ...

Component based applicaties

Component directives definiëren

Het is nu mogelijk makkelijk een component te definiëren met behulp van de module.component helper methode. Een component moet nu enkel een template, enkele bindings en een controller hebben. Voor de overige directive opties zijn er defaults geset.

```
myModule.component('myComponent', {
  template: '<h1>Hello {{ $ctrl.getFullName() }}</h1>',
  bindings: { firstName: '<', lastName: '<' },
  controller: function() {
    this.getFullName = function() {
      return this.firstName + ' ' + this.lastName;
    };
  }
```

```
});

<my-component first-name="'Alan'" last-name="'Rickman'">
</my-component>
```

Lifecycle hooks

Als je component controller een $onInit() methode heeft zal de compiler deze oproepen wanneer de component geïnitialiseerd is. Dit zorgt voor een duidelijke plaats om de initialisatie code voor je componenten te plaatsen. Dit is ook gelijkaardig aan de ngOnInit() lifecycle hook van Angular 2.

Binding to required directives

Er kan nu een object gespecificeerd worden voor de require property van directives-/componenten. Wanneer gebruikt wordt met bindToController zal de compiler deze required directive controller binden aan je controller. Deze bindings zullen al gebeurd zijn wanneer de $onInit hook wordt opgeroepen.

Default transclusion content

ngTransclude zal niet langer content overschrijven wanneer er geen content is om op te vullen. Nu kan je bijvoorbeeld default content voorzien wanneer de gebruiker van je directive geen content heeft voorzien om in te voegen.

Multi-slot transclusion

One-way binding

ngResource

ngRoute

Hoofdstuk 6

Overstappen naar Angular 2.0

De overstap maken naar Angular 2 in je project kan je op twee manieren aanpakken. De manier die je kiest hangt natuurlijk af van de vereisten van je project.

[Misschien ook handig eerst uit te leggen hoe je een Angular 2 app maakt, zonder overgang van Angular 1? https://angular.io/docs/ts/latest/quickstart.html]

ngForward Wanneer je nog niet zeker weet of nog twijfels heb of je je project wilt upgraden naar Angular 2 kan je het framework ngForward gebruiken. Hiermee kan je Angular 1 applicaties maken die de look en feel van Angular 2 hebben. Op die manier kan je toch profiteren van de voordelen van Angular 2 terwijl je in je comfort zone blijft.

ngUpgrade Wanneer je meer wilt dan enkel de look van Angular 2 en je klaar bent de overstap te maken naar Angular 2 is het tijd je applicatie te upgraden. Hiervoor kan je ook ngForward gebruiken maar het probleem hierbij is wanneer je een groot Angular 1.x project heb, je ook alles zal moeten herschrijven om te voldoen aan de Angular 2 vereisten. Dit is zeker geen ideale oplossing en daarom is ngUpgrade gebundeld met Angular 2.

Hoofdstuk 7

Conclusie

Bibliografie

Lijst van figuren

Lijst van tabellen