Projektdokumentation Angular JS 2
show4you | Eventapp

Jorge Ayala und Jörg Kandzioraaaaaaaaaaa

12.07.2016

Inhaltsverzeichnis

Teil I
Idee und Ziel

1 Idee

Jörg hatte die Idee eine Anwendung mit Angular2 zu erstellen, bei der man Events eintragen und verwalten kann. Zusätzliche soll diese Anwendung Zwei API's von Google verbunden sein. Man soll mit Google Maps sehen wo diese Events stattfinden und man soll das Datum und die Uhrzeit direkt in seinen persöhnlichen Google Calendar eintragen können. In der eigentlichen Anwendung gibt es ein Formular, bei dem man alle wichtigen Daten zur Veranstaltung eintragen kann. Es gibt eine Liste die alle Events anzeigt und wenn man auf ein einzelnes Event klickt bekommt man die details.

2 Ziel Definition

Wir haben uns zum Ziel gesetzt diverse Angular2 Funktionen und Attribute dafür zu verwenden. Zum einen Nuten wir Datenobjekte(Models) mit denen wir Venue, Datum/Uhrzeit, Bands, Genre etc darstellen. Mit einer Listenansicht sollen die verschiedenen Events angezeigt werden. Mit der Einbindung der Google Maps Api soll dann noch angezeigt werden wo sich die Venues befinden. In den Event Details, die Angezeigt werden wenn man auf ein einzelnes Event klickt, soll man das Event seinem persönlichen Google Calendar hinzufügen können. Mittels Routing erstellen wir unterseiten, wie maps(hier werden alle aktuellen Veranstaltungen angezeigt), calendar(hier soll der persönliche google calendar eingeblendet werden(Monatsansicht)), Add shows(hier soll man neue Veranstaltungen hinzufügen können).

Teil II
Erstellung einer einfachen Angular2 App

3 Erste Editierung

Zum Anfang hat Jörg eine App mit Hilfe des Tutorials "Tour Of Heroesërstellt. Hier wurde erstmal die Grundstruktur erstellt und mit app.component.ts die erst ts Datei. In der app.component.ts wurde dann unsere erste Klasse AppComponent erstellt von der alles aus gehen soll. Zudem haben wir die Klasse Show erstellt, welche als Atribute erstmal nur eine id und einen Namen bekommen haben. Beispielsweise wurde auch gleich eine Show erstellt. Eine Template wurde erstellt, welches die erste Ausgabe darstellt und direkt unsere erste Show anzeigt. Mit show.name und show.id könnnen wir uns die Show ausgeben lassen. Mit <input> und [(ngModel)] gehen wir dann zum two way binding und können unsere Show in Echtzeit verändern. Somit steht unsere erste kleine Anwendung. Die Html Tags die für die Ansicht da sind werden im @Component, welches mit Component from '@angular/core'; importiert wird, mit template eingefügt. Hier ist auch der selector definiert bei dem man den App namen angibt.

4 Interaktion und Detail Ansicht + Styling

Jörg hat dann an der Liste gearbeitet die die Shows anzeigen soll und die Detailansicht, welche die Details der einzelnen Shows anzeigen soll. Zudem stylen wir die Liste. Als ersten haben wir mit cont SHOWS: Show[] = [....] einen Array von einigen Shows erstellt. Mit public shows = SHOW machen wir die property dann öffentlich. Mi *ngFor="let show of Shows"lassen wir uns dann alle shows aus dem eben erstellten array auflisten.

4.1 Detail Ansicht mit +ngif

Jorge hat dann in die liste mit (click)=önSelect(show)"die einzelnen Events auswählbar gemacht. Mit *ngIf=ßelectedShow"konnte das event binding an eine neue Ansicht gebunden werden. Mit selecteShow.id wurde somit erstmal die Id der einzelnen Show mittels rauf klicken angezeigt. Das Problem ist, dass das jetzt noch alles recht unschön in einer Datei drinnen ist.

Teil III

Components, Services, Html Auslagerung Jörg || Parrallel neue Angular2 update und Http Jorge

5 Components

Jörg hat dann mittels components die einzelnen Abschintte ausgelagert.Statt mit template kann man mit templateUrl: ” die die html Tags auslagern in einer extra html Datai. Mit styleUrls: [”] kann man dann auch noch das Styling auslagern. Und schon sieht die Datei gleich übersichtlicher aus. Jetzt wird eine richtig Componente erstellt. Die show-detail.component.ts soll nur für die Details der Shows da sein. Und die show.ts nur für eine einzelne schow, also die Definition der einzelnen Attribute(bis jeztzt nur Name und Id). In unserer app.component.ts können wir die beiden Components dann importieren. Sobald die importiert sind können wir alles aus den Compents jetzt zum Beispielt in unserer ausgelagerten Html Datei benutzen.

```
import {ShowsDetailComponent}   from './shows-detail.component';
import {Show}                   from './show';
```

5.1 show.ts

In die shot.ts kommt erstmal nur die Klasse Show rein, in der wie die id und den namen mit number und string definieren. Die KLasse muss natürlich mit ëxport class Showëxportiert werden, damit wir sie woanders benutzen können.

5.2 show-detail.component.ts

Die Component ist dann schon aufwendiger. Hier wird erstmal auch die shot.ts importiert wie in der app.component.ts. Zudem brauchen wir hier auch das Component und wollen das <Input feld ja verwendne also muss man das mit import Component, Input from ’@angular/core’; auch einbinden. Im @Component () wird dann auch ein template eingebunden welches wir auch mit templateUrl auslagern können, da hier aber auch in Zukunft nicht mehr viel hinzu kommt wird es direkt in die Datei geschrieben. Wir erstellen dann noch die Klasse ShowDetailComponent die wie oben exportiert wird. Hinein kommt @Input() show: Show; die wir ja dann anzeigen lassen wollen.

6 Services

Jetzt erstellen wir für unser Projekt einige services. Dazu erstelle ich erstmal eine mock-show.ts welche nur dafür dient die Attribute unserer show zu haben. Sie importiert von show.ts und exportiert selber mit SHOWS: Show []= [...] eine Liste von Shows mit deren Attributen. Hier füge ich gleich mal unseren weiteren Attribute hinzu. Dann kommt die show.service.ts welche Show von der hero.ts und SHOWS von der mock-show.ts importiert. Zudem wird injectable von @angular/core importiert. Mit @injectable() können wir die Klasse ShowService exportieren, welche eine getter Methode hat. Mit return Promis.resolve(Shows). In der app.component.ts wird ShowService importiert und in der @Component beim neuen Feld providers: mit [ShowService] hinzu gefügt. Zusätzlich brauchen wir jetzt einen Constructor der private showService: ShowService deklariert. Mit folgenden Methode können wir uns dann die Shows holen.

```
getShows() {
    this.showService.getShows().then(shows => this.shows = shows);
}
```

7 Angular2 Update

8 Http

Teil IV
Routing

Beim Routing hat durch die unvollständige Dokumentation von Angular, sehr die aus unserer Arbeitsgruppe geholfen. Als erstes wurde unsere app.component ëntlastetïn dem sie nur noch zum verlinken/routing da sein soll. Dazu wurde eine neue Component erstellt show-list.component, die ab jetzt die ganzen shows in einer Liste anzgeigen soll.

Zudem wurden jetzt extra Components und dazuhgehörige htmls und ab und zu auch css Dateien für eine extra Maps Ansicht, eine Calendar Ansicht, eine Venues Ansicht und eine Addshows Ansicht erstellt. Die Addshows ist dazu da shows hinzuzufügen, abseits von der Show Ansicht.

Die Venues Unterseite soll wieder eine Lexika mit allen in Berlin vorhanden Veranstaltungsräumen funktionieren, heißt man kann nachschauen was es so für welche gibt und sieht gleich auf einer google maps Karte wo die sich befinden.

Bei der Maps Ansicht werden auf einer Karte alle Veranstaltungen gleichzeitig angezeigt, damit man sich einen guten überblick verschaffen kann.

Eingebunden wurde das Routing in der maint.ts mit Router Providers. In der Index.html bei den maps und packages mit @angular/router. Dann natürlich auch bei unserer neuen Hauptnavigationsseite app.component mit

```
import { Routes, ROUTER_DIRECTIVES } from '@angular/router';
```

Teil V
Api Anbindungen

9 Maps Api

Jorge hat die hat nach der anbindung der Maps Api recherchiert. Und eingebunden. Dazu musste einmal die Maps Api in die main.ts mit

```
import {GOOGLE_MAPS_PROVIDERS, provideLazyMapsAPILoaderConfig} from 'angular2-google-maps/core';
```

eingebunden werden. Zudem musste im bootstrap

```
GOOGLE_MAPS_PROVIDERS,
```

eingetragen werden. Zudem musste die in die Index.html eingebunden werden. Einmal in die map: mit

```
'angular2-google-maps'        : 'node_modules/angular2-google-maps'
```

und dann bei den packages
Somit kann man dann auf die Maps Api zugreifen.

```
'angular2-google-maps/core'        : {main: 'index.js'}
```

In den Components in denen wir dann die Maps benutzen kann diese dann importiert werden. Die muss dann auch bei den directives eingetragen werden. Es wurde eine Grundkoordinate festgelegt welche mit lat:number = 52.5243700; lng:number = 13.4105300; den Bereich von Berlin zeigt.

```
<sebm-google-map [latitude]="lat" [longitude]="lng" [zoom]="12">
    <sebm-google-map-marker
            *ngFor="let show of shows; let i = index"
            [latitude]="show.address_lat"
            [longitude]="show.address_lng"
            [markerDraggable]="true">

        <sebm-google-map-info-window>
            <strong  *ngFor="let show of shows"> {{show.name}} | {{show.address}}</strong>

        </sebm-google-map-info-window>


    </sebm-google-map-marker>
</sebm-google-map>
```

10 Calendar Api

Bei der Calender Api war die Herandgehensweise ähnlich wie bei der Maps Api, außer dass es weit aus mehr komplikationen gab. Was zum einen an der noch sehr schlechten Dokumentation lag.

Teil VI
Finalisierung

Bei der Finalisierung wurden verschiedene Verbesserungen vorgenommen. So wurde zum einen die Detail Ansicht der einzelnen Shows aufgeteilt in eine View und eine Edit Ansicht, die sich per Button wechseln lassen. Zum anderen kann man die Detail Ansicht jetzt wieder schließen. Des Weiteren Zeigt jetzt die Kurzinformation in der Liste nicht nur noch den Namen der Show an sonder auch das Datum und die Venue. Die Addshows Component wurde so verändert dass nach dem eintragen und speicher dass Formular erstmal verschwindet, falls man noch eine weitere Show hinzufügen möchte kann man das Formular wieder mit einen Button aufrufen.

11 Weiteres

11.1 Literatur

- K. Suffer – Ray Tracing from Ground Up – Kapitel 13
- P. Shirley – Fundamental of Computer Graphics – Kapitel 19

11.2 Grafiken und Quellen

Stephan Rehfeld: http://rehfeld.beuth-hochschule.de/