Angular – Buchhandel – Teil 4

Inhalt:

- 1)Startseite anlegen (ng generate component home)
- 2)Routing erstellen
- 2.1)Routing Datei erstellen „app-routing.module.ts“
- 2.2)Routes erstellen z.B. { path: 'home', component: HomeComponent }
- 2.3)Einbinden in app.module.ts
- 2.4)RouterOutlet platzieren
- 3)Parameter für Detailrouting
- 4)Navigationsleiste anlegen
- 5)Link von der Buchliste zur Detailseite
- 6)Button auf der Startseite

Üblicherweise wird das Routing automatisch beim Erstellen eines neuen Projekts erstellt. Man muss die Frage einfach mit „Y“ beantworten, wenn man im Terminal gefragt wird „Wollen Sie ein Routing erstellen?“.

Hier in diesem Beispiel wurde ein „n“ eingegeben und man muss nun alle Schritte von Hand selbst anlegen. Dies dient dem besseren Verständnis.

Routing – manuell anlegen

Eine der Kernfunktionalitäten von Webapplikationen ist das Routing, also die Abbildung einer aktuellen Seite auf einer adressierbaren URL.

Als Routing bezeichnet man das Laden von Bereichen der Anwendung abhängig vom Zustand. Der Dienst, der den Zustand der Angular-Anwendung verwaltet, nennt sich Router. Er tauscht automatisch die geladenen Komponenten aus und ermöglicht damit die Navigation zwischen verschiedenen Bereichen.

Ziel: Alle Ansichten sollen vom Nutzer über URLs aufrufbar sein. Über klickbare Links soll man dann durch die Anwendung navigieren.

Ziel:

- Es soll ein Menü im oberen Bereich der Anwendung existieren mit zwei Buttons (Home, Bücher)
    - Top-Level-Routen
    - <a routerLink='#'>Startseite
- Eine Startseite soll vorhanden sein
- Man soll aus der Listenansicht zur Detailansicht eines speziellen Buches wechseln können
    - ChildRoutes – verschachtelte Routenkonfiguration
    - { path: 'home', component: HomeComponent }
- Jede einzelne Ansicht soll durch eine eindeutige URL repräsentiert sein

In der Komponente „AppComponent“ soll eine Navigationsleiste angelegt werden, um zwischen den Ansichten umschalten zu können. Hier soll auch ein „RouterOutlet“ platziert werden. In dem die

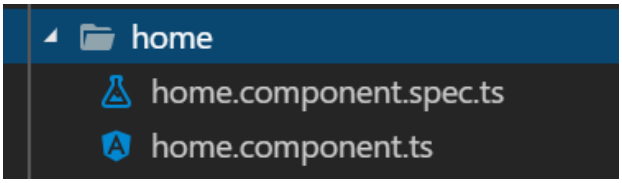
jeweilige geladene Komponente angezeigt wird.
Zum Aufrufen der Komponente „Details“ (BookDetailsComponent) soll die Info mittels ISBN als Parameter übergeben werden.

1.)Startseite anlegen

Erstelle die neue Komponente mit der Angular CLI

```
ng g component home
```

Ergebnis:



2.)Routing erstellen

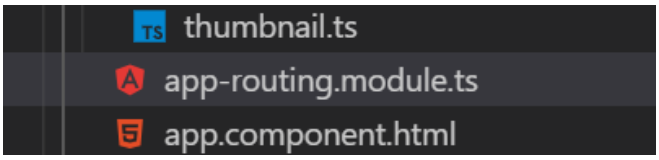
Dafür gibt es prinzipiell drei Schritte

- Router konfigurieren (4.2.2.)
- RouterModule einbinden (4.2.3.)
- RouterOutlet platzieren (4.2.4.) – der Bereich, in dem der Routerinhalt dargestellt werden soll

2.1. Manuell eine Routing-Datei anlegen

Datei erstellen: app-routing.module.ts

Stelle dich auf „app“ und mit rechter Maustaste „Date erstellen“ mit dem Namen „app-routing.module.ts“ anlegen.



Normalerweise wird diese Routing Komponente bereits beim Neuanlegen eines gesamten Projekts mit angelegt, nämlich mit „ng new projektname --routing“. Das wurde hier nicht gemacht, daher kann man das nur manuell nachholen.

2.2.)Router konfigurieren

Öffne „app-routing.module.ts“.

Es müssen drei Routen erstellt werden, nämlich für die neue HomeComponente, die Listen-Site und die Detail-Site.

Theorie:

In der Anwendung möchte man beim Aufruf einer URL eine bestimmte Komponente anzeigen. Diese Zuordnung erledigt man mit der Router Definition. Hier wird ein Zustand der Anwendung durch eine URL definiert. Eine solche Route wird als Objekt notiert und ist folgendermaßen aufgebaut:

```
{ path: 'home', component: HomeComponent }
```

Im Objekt gibt man den

- URL-Pfad an (path) und die
- Komponente, die geladen werden soll (component).

Wichtig ist, dass die Pfade **niemals** einen Slash vorangestellt haben. Das heißt: wird der Pfad ‚home' aufgerufen, dann wird die Komponente geladen.
Damit man den Namen der Komponente direkt verwenden kann, muss man den Typ aus der jeweiligen Datei importieren.
Da sich immer mehrere Routen befinden, legt man diese in einem Array ab. Für das Array legt man den Typ „Routes" fest, der aus dem Paket @angular/router importiert wird.

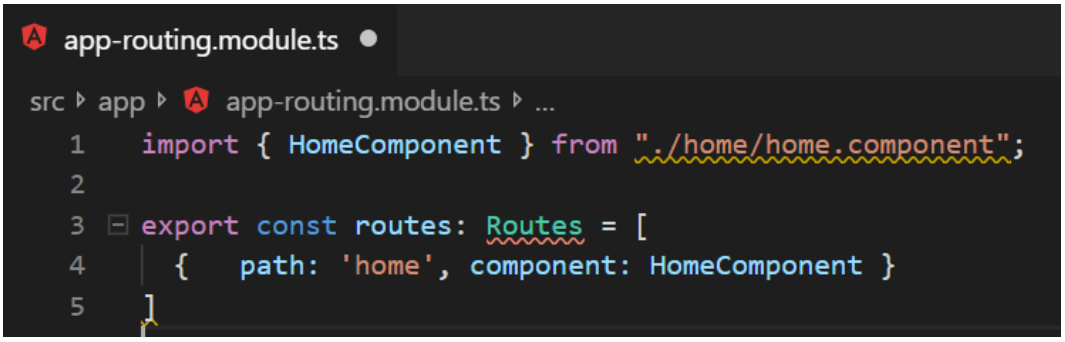
Übung:

Öffne diese neue Datei „app-routing.modules.ts" und erstelle die Konfiguration darin.

- Beginne mit

```
export const routes: Routes = [
  {   path: 'home', component: HomeComponent }
]
```

Der obere Bereich mit „import" kommt automatisch von selbst. Wenn nicht, muss man es eingeben. Denn alle Komponenten, auf die man in den Routen verweist, muss man im Kopf der Datei importieren.



Gib auch die beiden anderen Routen ein: Beistrich nach jeder

Wichtig: Keine Leerzeichen vor oder nach den „books" usw. in Zeile 6 oder 7 oder 8

app-routing.module.ts

```
src ▸ app ▸ app-routing.module.ts ▸ ...
1   import { HomeComponent } from './home/home.component';
2   import { BookListComponent } from './book-list/book-list.component';
3   import { BookDetailsComponent } from './book-details/book-details.component';
4
5   export const routes: Routes = [
6     { path: 'home', component: HomeComponent },
7     { path: 'books', component: BookListComponent },
8     { path: 'books/:isbn', component: BookDetailsComponent }
9   ];
```

Erstelle in den „imports“ auch sofort das Routing für den Typ „RouterModule“. Damit macht man die Schnittstelle zum Angular-Router verfügbar: hier Zeile 2

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

app-routing.module.ts

```
src ▸ app ▸ app-routing.module.ts ▸ ...
1   import { Routes, RouterModule } from '@angular/router';
2   import { HomeComponent } from './home/home.component';
```

<u>Noch zu berücksichtigen:</u>

Beim Start fehlt die Route, weil ja nicht z.B. „home“ eingegeben ist, sondern nur die Start-URL, wie z.B. „localhost:4200“. Aber ein „RouterOutlet“ darf nie leer sein.
Daher muss man eine neue Route anlegen, in der der „path“ leer ist. In der legt man dann fest, dass der Root-Pfad auf die URL /home weiterleiten soll, damit der Nutzer sofort zur Startseite gelangt.
Daher benötigt man eine Routenweiterleitung.

- Man verwendet die Eigenschaft „redirectTo“ und gibt einen Pfad zur Weiterleitung an.
- pathMatch: ‘full‘ – damit greift die Route sicher

Diese Route füge vor der ersten ein.

{ path: '', redirectTo: 'home', pathMatch: 'full'},

(Info: man könnte auch das gleiche Ergebnis erzielen mit: { path: '', component: HomeComponent, pathMatch: 'full'},).

Speichern. Strichpunkt ganz hinten nicht vergessen.

```
5   export const routes: Routes = [
6     { path: '', redirectTo: 'home', pathMatch: 'full' },
7     { path: 'home', component: HomeComponent },
8     { path: 'books', component: BookListComponent },
9     { path: 'books/:isbn', component: BookDetailsComponent }
10  ];
```

Darunter erfolgt die Eingabe von @NgModule(), wobei sich automatisch der Import oben erstellt, siehe Zeile 5.

```
14  @NgModule({
15
16  })
```

```
1  import { Routes, RouterModule } from '@angular/router';
2  import { HomeComponent } from './home/home.component';
3  import { BookListComponent } from './book-list/book-list.component';
4  import { BookDetailsComponent } from './book-details/book-details.component';
5  import { NgModule } from '@angular/core';
```

Das NgModele befüllen:

Code:
@NgModule({
 imports: [RouterModule.forRoot(routes)],
 exports: [RouterModule],
 providers: []
})

In Zeile 15 ruft man die Methode „forRoot()“ auf und übergibt als Argument das Array mit den Routendefinitionen. Die Funktion „forRoot“ nimmt die Liste von Routenkonfigurationen als Parameter entgegen.

```
14  @NgModule({
15    imports: [RouterModule.forRoot(routes)],
16    exports: [RouterModule],
17    providers: []
18  })
```

Danach muss das Modul noch exportiert werden, da es in der Datei „app.module.ts“ benötigt wird. Dort wird damit das Routing in der Anwendung registriert.

```
18  })
19  export class AppRoutingModule { }
```

Das Modul „AppRoutingModule“ erstellt ein fertiges Routing mit den selbst definierten Routen und stellt dieses Modul nach außen zur Verfügung.

Eigentlich ist die gesamte Konfiguration des Routings mit allen nötigen Abhängigkeiten hier gespeichert.

2.3. Routing-Modul einbinden (nur wenn wie hier manuell angelegt wurde)

Dies erfolgt in der „app.module.ts“. Öffne diese. Importiere das AppRoutingModule in die Anwendung.

Füge in Zeile 3 ein:

```
app.module.ts
src ▸ app ▸ app.module.ts ▸ ...
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { AppRoutingModule } from './app-routing.module';
4
5  import { AppComponent } from './app.component';
```

Dann bei „imports“ noch hinzufügen, nach einem Beistrich nach „BrowserModule“:

```
20  imports: [
21    BrowserModule, AppRoutingModule
22  ],
```

Speichern.

Damit ist das Routing global in der Anwendung aktiviert.

2.4.) RouterOutlet platzieren

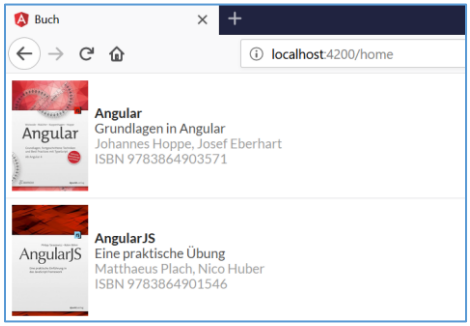
Damit auch eine Anzeige erfolgt, muss man in der „app.component.html“ diese erstellen.
Öffne die Datei und fügen unterhalb der <h1> ein „RouterOutlet“ ein.

Durch das Routing werden die Komponenten nun dynamisch geladen. Dafür existiert die Direktive „RouterOutlet“. Sie ist ein Platzhalter und wird vom Router dynamisch durch die geladene Komponente ersetzt.

```
app.component.html
src ▸ app ▸ app.component.html ▸ div ▸ router-outlet
1  <!--The content below is only a placeholder and c
2  <div style="text-align:center">
3    <h1>
4      Welcome to {{ title }}!
5    </h1>
6    <router-outlet></router-outlet>
```

Teste die Anwendung.

Ergebnis: es wird auf „home“ geleitet.



4) Navigationsleiste anlegen

Hier werden klickbare Links erstellt. Dafür wird eine Navigationsleiste erstellt.

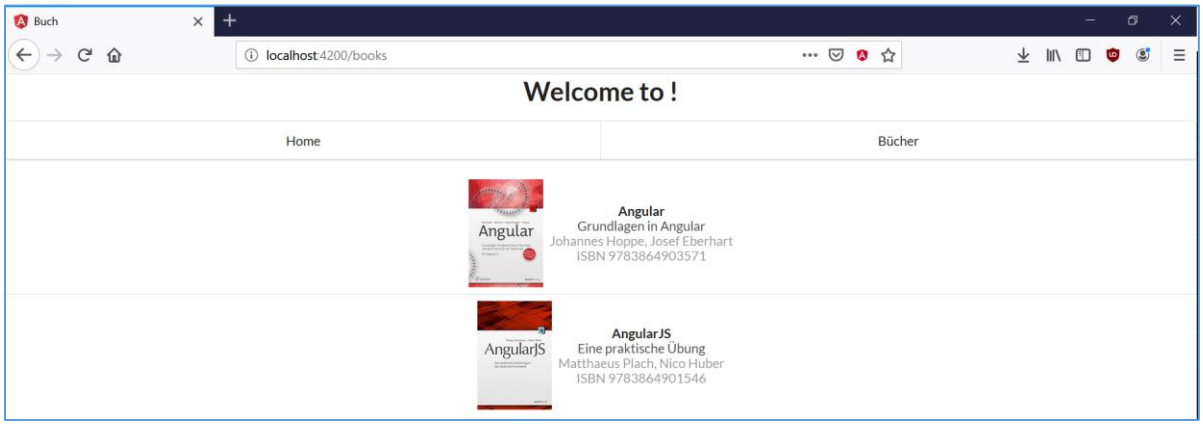
- Der beste Platz ist die „app.component.html“, denn diese Komponente ist während der gesamten Laufzeit sichtbar.
- Unter der Menüleiste befindet sich das RouterOutlet, in das die geroutete Komponente geladen wird.
- Das „Router-Package“ bietet zwei Direktives an zur Benutzung:
    - **routerLink**: Um das Link-Ziel festzulegen, verwendet man nicht das Attribut „href“, sondern setzen die Direktive „RouterLink“ ein, um den „path“ anzugeben.
    - **routerLinkActive**: damit wird ein angeklickter Link hervorgehoben, mit der Klasse „active“. Hier im Beispiel wird aber eine eigene Klasse verwendet „item“.

Angular bringt für Verlinkungen ein eigenes Werkzeug mit – die **Direktive RouterLink**: damit wird beim Klick auf einen Link automatisch der Router informiert, eine neue Route zu laden. Der übrige Teil der Anwendung bleibt bestehen und nur die zu ladende Komponente wird vom Server abgerufen.

app.component.html

```
src ▸ app ▸ app.component.html ▸ ...
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
  <div class="ui two item tabs menu">
    <a routerLink="home" class="item">Home</a>
    <a routerLink="books" class="item">Bücher</a>
  </div>
  <router-outlet></router-outlet>
</div>
```

Ergebnis:

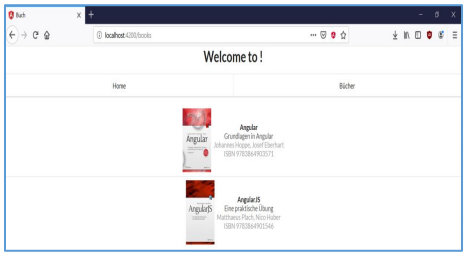


Info:

Eine „View“ besteht bei der Verwendung des Routers immer aus zwei Teilen:

- ein statischer Teil – die Links zu den Unterseiten

- dynamischer Teil – dem sogenannten RouterOutlet, in dem die dynamischen Inhalte der Route gerendert werden



statischer Teil

Dynamischer Teil

<router-outlet>

5.) Link von der Buchliste zur Detailseite

Hier soll der Nutzer die Detailansicht eines Buches erreichen können, wenn er ein Buch in der Listenansicht anklickt. Dazu muss man das Template der Komponente „BookListComponent“ anpassen und das Linkziel für die Einträge der Buchliste festlegen.

Öffne „book-list.component.html“

Der Direktive „RouterLink“ muss man als Property Binding verwenden, denn man übergibt ja keinen String, sondern einen Ausdruck.

Info: Die Buchliste wird durch den Pfad /books geladen. Um von hier auf die ISBN zu gelangen (/books/123456789), reicht es aus, einen Pfad zum relativen Pfad 123456789 zu setzen.

Füge also ein:

[routerLink]="b.isbn"

```
<div class="ui middle aligned selection divided list">
  <a *ngFor="let b of books" [book]="b" [routerLink]="b.isbn"
  class="bm-book-list-item item">
  </a>
</div>
```

6.) Button auf der Startseite

Dieser soll auf die Buchliste verweisen.

Die „HomeComponent“ wird für den Pfad /home geladen. Der relative Pfad zu /books lautet deshalb „../books“

Öffne die home.componente.ts:

Code:

```
<div class="ui container">
    <h1>Home</h1>
    <p>Das ist unser Buchladen.</p>
    <a routerLink="../books" class="ui red button">
    Buchliste ansehen
```

```
<i class="right arrow icon"></i>
</a>
```

home.component.ts

src ▸ app ▸ home ▸ home.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'bm-home',
  template: `
    <div class="ui container">
      <h1>Home</h1>
      <p>Das ist unser Buchladen.</p>
      <a routerLink="../books" class="ui red button">
      Buchliste ansehen
      <i class="right arrow icon"></i>
      </a>
  `
})
```

Ergebnis: