Angular - erstes Projekt

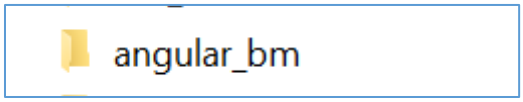
Inhalt:

Übung: Buchhandel

In dieser Webanwendung sollen Bücher verwaltet werden. Dafür soll es eine Übersicht in Listenform geben und per Klick soll ein einzelnes Buch mit mehr Details angezeigt werden können. Danach soll man den Buchbestand selbst verwalten können.

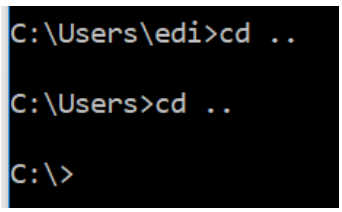
1)Neues Projekt erstellen

- Dazu erstelle am Beginn einen neuen Ordner, hier auf C: namens „angular_bm“. Bm steht hier für „Buchmarkt“.

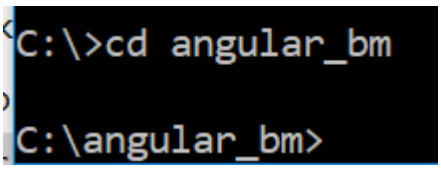


Mittels CMD arbeiten wir direkt in der Angular CLI, um die Grundstruktur des Projektes anzulegen.

- Navigiere dann in CMD (Benutzereingabe) mittels „cd ..“ in den neuen Ordner, zuerst einmal aus dem sich öffnenden bis hin zu C:



- Dann in den neuen Ordner hinein mit „cd angular_bm“



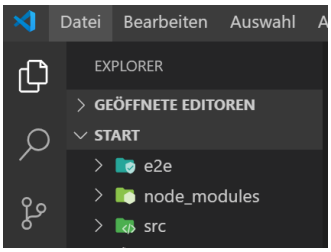
- Erstelle nun hier das neue Projekt namens „start“ mittels dem Code:

  ng new start

```
C:\angular_buchhandel>ng new buch -p bm -is
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
```

2)Visual Studio Code öffnen

Öffne den passenden Ordner „buch“ auf C: mit Datei / Ordner öffnen:



Klicke im Menü „Anzeige“ auf „Terminal“ um in VS-Code direkt auf das Terminal (=CMD) zugreifen zu können.

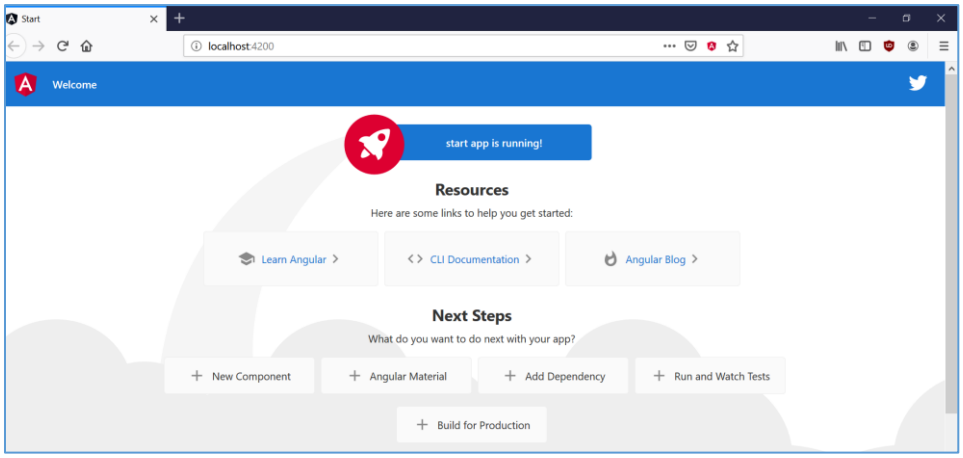
Starte hier nun den Server mit

- ng serve

```
PS C:\angular_bm\start> ng serve
```

Die fertige Anwendung ist nun erreichbar. Starte nun im Browser die Startseite mit Eingabe in die Browserzeile:

localhost:4200



Solange der Server gestartet ist und man Änderungen speichert, aktualisiert sich die geöffnete Website stets automatisch im Browserfenster.

Info: Kurzeingabe zum gleichzeitigen Start im Browser

- **ng serve --open**

```
PS C:\angular_bm\start> ng serve --open
```

Info:
Diese Startseite wird hier erzeugt: app.component.html.

3)Theorie

zu den neuen Komponenten und Templates

Allgemein: Die Quelltexte der Anwendung liegen immer im Ordner „app“.

3.1)Komponenten

sind die Grundbausteine der Anwendung.

Jede Anwendung ist aus vielen Komponenten zusammengesetzt, die jeweils eine bestimmte Aufgabe erfüllen. Eine Komponente beschreibt somit immer einen kleinen Teil der Anwendung z.B. eine Seite.

Eine Komponente hat einen Anzeigebereich, die View, in dem ein Template dargestellt wird. Das Template ist das „Gesicht“ der Komponente, also der Bereich, den der Nutzer sieht.

Beispiel:

Öffne die Komponente „app.component.ts“

app.component.ts

src ▸ app ▸ app.component.ts ▸ ...

```
1  import { Component } from '@angular/core';
2
3
4  @Component({
5    selector: 'bm-root',
6    templateUrl: './app.component.html'
7  })
8
9  export class AppComponent { }
```

3.1.1)Grundgerüst:

Nicht nur bei Komponenten steht am Beginn immer eine oder mehrere „import-Anweisungen“.

Dadurch hat man Zugriff auf die importierten Klassen, hier die Klasse „Component“, und kann diese Klasse verwenden.

Eine Komponente wird immer mit dem Decorator

- @Component() eingeleitet.

Decorator: damit kann man Klassen um zusätzliche Informationen, so genannte Metainformationen, erweitern und die Übersichtlichkeit im Code fördern.
Dem Decorator werden die Metadaten übergeben.

Template: eine Komponente ist immer mit einem Template verknüpft. Das Template wird vom Nutzer gesehen und ist daher meist in HTML geschrieben, da es im Browser ausgeführt wird.

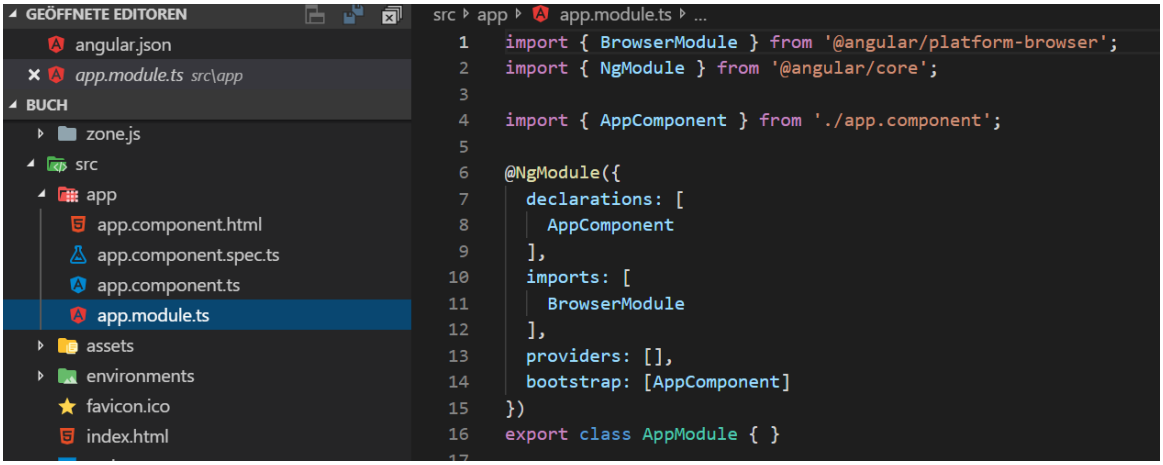
Templates und Komponenten sind eng miteinander verknüpft und können miteinander kommunizieren, über sogenannte Bindings. Das funktioniert in beide Richtungen, nämlich von der Komponente in das Template (= property binding) und vom Template in die Komponente (=event binding). Beim letzteren können Ereignisse im Template abgefangen werden, um von der Komponente verarbeitet zu werden.

Komponente registrieren:

Damit dieser Mechanismus funktioniert, müssen die Komponenten von Angular allerdings erst kennengelernt werden. Die reine Existenz einer Komponente reicht nicht aus. Man muss daher alle Komponenten im zentralen **AppModule registrieren**. Hier die Datei „app.module.ts“.

Dazu verwendet man die Eigenschaft „declarations“ im Decorator @NgModule(). Hier werden alle Komponenten notiert. Damit man alle verwenden kann, muss man alle importieren:

Bereits standartmäßig vorhanden ist siehe Zeile 4 und 6-8:



3.1.2)Template-Syntax

Die Notation von Templates kann in mehreren Varianten erfolgen. Angular erweitert dabei die gewohnte Schreibweise (Syntax) von HTML. Dadurch kann man dynamische Features direkt nutzen, wie z.B. Ausgabe von Daten, Reaktion auf Ereignisse und das Zusammenspiel von mehreren

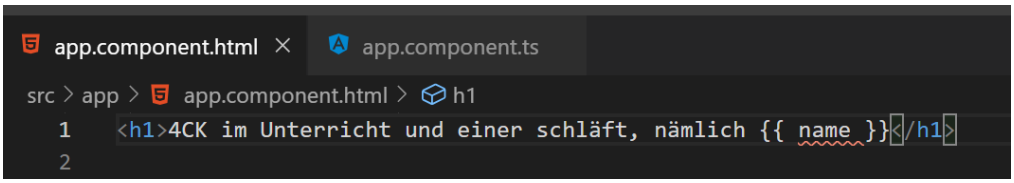
Komponenten mit Bindings. Dadurch lassen sich z.B. Ereignisse wie „click“ oder „mouseover“ anstoßen.

Jedes Verfahren verfügt über eine eigene Schreibweise.

Zeichen	Bezeichnung	Funktion
{{}}	Interpolation	Daten im Template anzeigen
[]	Property Binding	Eigenschaften eines DOM-Elements setzen
()	Event-Binding	Ereignisse abfangen und behandeln
[()]	Two-Way Binding	Eigenschaften lesen und Ereignisse verarbeiten
#	Elementreferenzen	Direktzugriff auf ein DOM-Element
*	Strukturdirektiven	Direktiven, die den DOM-Baum manipulieren
I	Pipe-Operator	Transformation von Daten vor dem Anzeigen

3.1.2.1)Erste Interpolation {{interpolation}}

Öffne die „app.component.html“ und erstelle eine neue <h2> mit einer Interpolation:



Öffne die „app.component.ts“ und schreibe in der Klasse, siehe Zeile 10:

public name = "Jonas";

Nun nutze die Interpolation, indem in der Klasse darunter eine neue Property erstellen, mit dem Namen „schule“, die einen String (Inhalt) erhält.

Dieser Inhalt wird nun mittels Interpolation in der HTML-Datei eingesetzt.

Vorteil: diesen Inhalt der Property ist nicht mehr statisch und man könnte ihn auch aus der WepAPI und somit vom Server nehmen.

Ergebnis:

3.1.2.2)Beispiele: Two-Way-Binding

Beim Einsatz von Formularen gilt es häufig, Eigenschaften aus der Komponente mit Eingabefeldern in der Anwendung abzugleichen. Die Werte der Eigenschaften sind also in Formularfelder zu übernehmen und die Anwendung muss Änderungen an Formularfeldern zurück in die jeweilige Eigenschaft schreiben. Ändert die Komponente hingegen die Eigenschaft, ist der aktualisierte Inhalt erneut in das Eingabefeld zu übernehmen.

```
<input [(ngModel)]="from">
```

Damit man auf den ersten Blick erkennt, dass es sich hier um ein Two-Way-Binding handelt, nutzt Angular ein Paar eckige Klammern in Verbindung mit einem Paar runder Klammern. Man spricht auch von „Banana-in-a-Box" Schreibweise.

Bei „ngModel" handelt es sich um eine sogenannte Direktive. Diese fügen Verhalten zu einer Seite hinzu. In diesem Fall besteht das Verhalten im gewünschten Abgleich mit der angegebenen Eigenschaft.

Quellen:

Woiwode, Malcher, Koppenhagen, Hoppe in: Angular, Verlag dpunkt, 2018, S. 70-79

Steyer, Schwab in: Angular; Verlag O´Reilly, 2017, S. 52-54

Christoph Höller in: Angular, Rheinwerk 2019, S. 234-242

https://www.youtube.com/watch?v=2a6OfacW_-I&list=PLC3y8-rFHvwhBRAgFinJR8KHIrCdTkZcZ&index=5