# Angular – Buchhandel – Teil 3 – Service erstellen

Inhalt:

- 1.)Service erstellen (ng g service shared/book-store)
- 2.)Provider für den Service registrieren in app.module.ts
- 3.)Service verwenden

Mittels Services kann man Code in eigenständige Klassen auslagern.

Definition: Eine Funktion oder Klasse, welche eine Funktionalität für eine andere Funktion oder Klasse bereitstellt wird als Service bezeichnet.

Beispiel:
Man kann in einer Anwendung ein Service für Login (LoginService) und auch ein zweites für die Verwaltung von Aufgaben (TaskService) ohne Probleme verwenden.

Ziel: Buchdaten sollen aus der Listenkomponente in einen Service (Klasse „BookStoreService“) ausgelagert werden. Das ist dann der Grundstein dafür, die Daten in verschiedenen Komponenten zur Verfügung zu haben.

Um den Code zu vereinfachen, soll die Bereitstellung der Daten ausgelagert werden. Der Vorteil ist, dass Daten aus einer zentralen Quelle geladen werden.

- Die Komponente „BookListComponent“ wird somit abgespeckt. Der Code mit dem Bücher-Array wird nun in die Klasse „BookStoreService“ ausgelagert.
- Dadurch ist es auch viel einfacher möglich, später Daten von einer REST-Schnittstelle einzubinden.

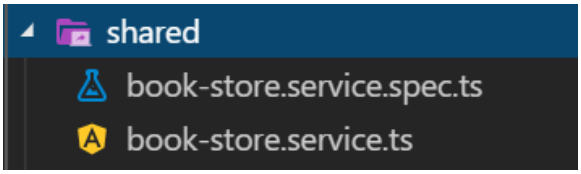
## 1.)Service erstellen

ng g service shared/book-store

Gib diesen Code mit Hilfe der Angular CLI ein.

```
PS C:\angular_buchhandel\buch> ng g service shared/book-store
```

Ergebnis:




**Theorie:**

Um Abhängigkeiten anfordern zu können, verwendet man den Decorator @Injectable.

Services werden also mit @Injectable dekoriert.

Den muss man zuerst einmal importieren und dann verwenden.

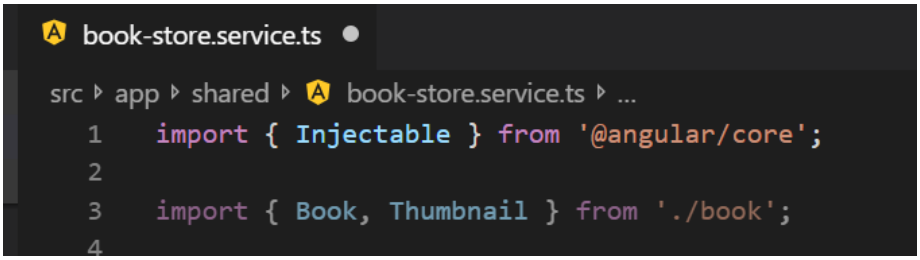
```
import { Injectable } from '@angular/core';
```

```
@Injectable()
```

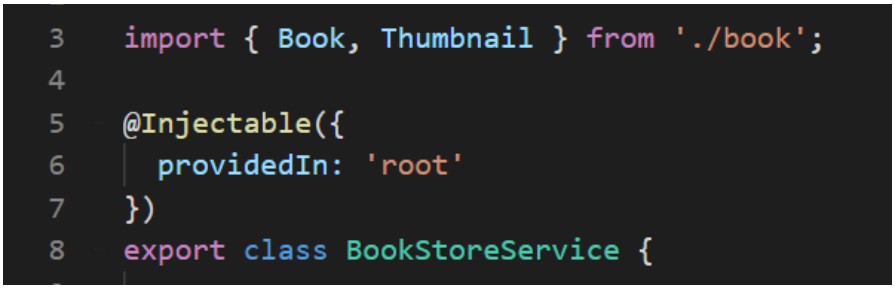
Das wird in der Datei „service.ts“ durchgeführt.

**Übung:**

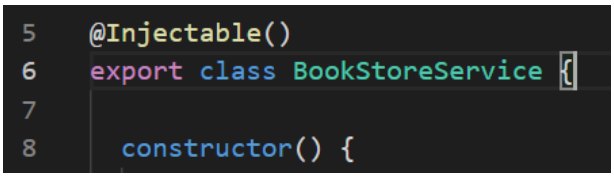
Öffne die Datei „book-store.service.ts“ und schreibe die Zeile 3: hier holt man aus dem gleichen Verzeichnis den Inhalt aus „book.ts“.




Nach der Zeile 5 lösche die Zeile 6 mit dem „providedIn“ mit den umspannenden geschwungenen Klammern.

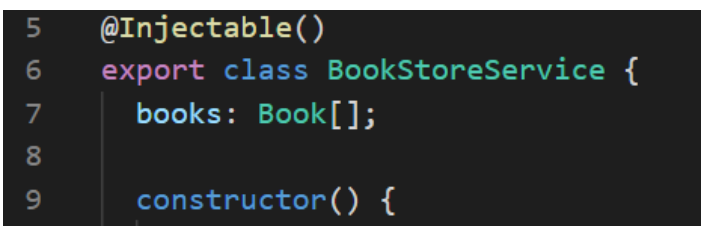



Ergebnis:




Füge nun in Zeile 7 ein:

books: Book[];



Danach kopiere aus der Datei „book-list.component.ts“, ab „this.books“ alle Bücher in den „constructor“ von „book-store.service.ts“.

```
export class BookStoreService {
  books: Book[];

  constructor() {
    this.books = [
      new Book(
        '9783864903571',
        'Angular',
        ['Johannes Hoppe', 'Josef Eberhart'],
        new Date(2019, 3, 1),
        'Grundlagen in Angular',
        5,
        [new Thumbnail('https://ng-buch.de/cover2.jpg', 'Buchcover')],
        'Mit Angular setzen Sie auf ein modernes ....'
```

Vor der letzten Klammer füge ein die Methode „getAll“:

```
        'Dieses Buch zeigt zusammenhaengende Beis
      )
    ];
  }
  getAll() {
    return this.books;
  }
}
```

Erklärung:
Die Methode „getAll()“ gibt alle Bücher zurück.

## 2.)Provider für den Service registrieren

Der neue Service muss noch bekannt gemacht werden. Dies geschieht in der zentralen AppModule, also in der „app.module.ts“.

**Theorie:**

Damit die entsprechenden „Bauanleitungen“ (Provider) übergeben werden können, kann man unterschiedliche Dekoratoren verwenden wie z.B. @NgModule(), @Component() und @Directive.

Normalerweise wird die Dependency per @NgModule() zur Verfügung gestellt.

Das soll stattfinden in „app.module.ts“.

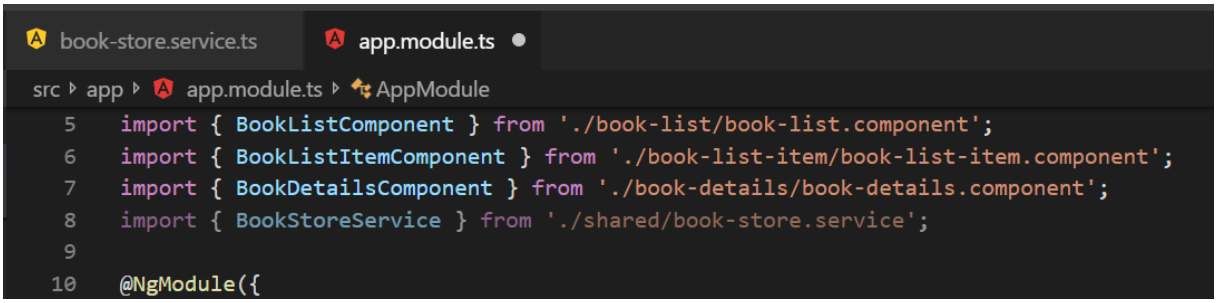
Dazu verwendet man stehts die Eigenschaft „providers“.

```
@NgModule({
  declarations: [
    AppComponent,
    BookListComponent,
    BookListItemComponent,
    BookDetailsComponent,
    HomeComponent
  ],
  imports: [
    BrowserModule, AppRoutingModule
  ],
  providers: [BookStoreService],
  bootstrap: [AppComponent]
})
```

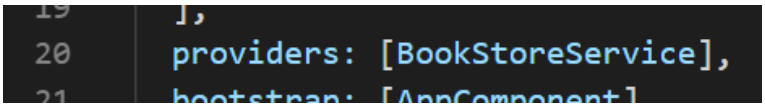
**Übung:**

Öffne die „app.module.ts“

- Füge Zeile 8 ein




- In Zeile 20 war der Provider noch leer. Füge ein:



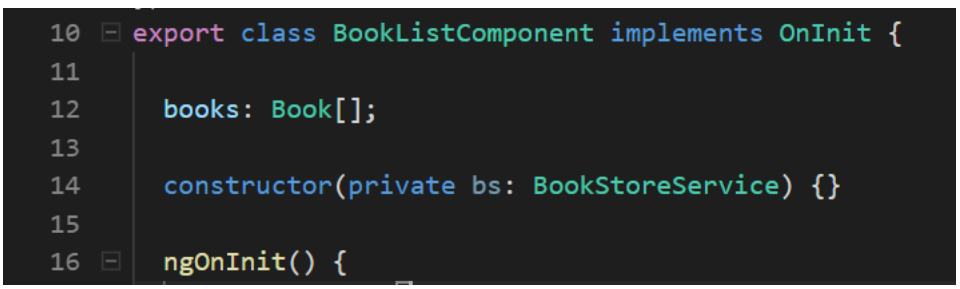

## 3.)Service verwenden

Nun kann man den Service in der Komponente „BookListComponent“ verwenden.

### 3.1.)Öffne „book-list.component.ts“.

- Füge Zeile 3 ein


- Füge in Zeile 14 ein.
- Es wird die Abhängigkeit über den Konstruktor angefordert.
- Ein Konstruktor ist eine besondere Methode, die während der Instanziierung einer Klasse aufgerufen wird. Es werden Werte übergeben, die man später verwendet.
- Durch die Verwendung des Zugriff-Modifizierers „private“ steht die Variable als Klassen-Property zur Verfügung.



- In „ngOnInit“ kann man nun zur Initialisierung des books-Arrays auf die Servicemethode zugreifen.
- Zusätzlich kann man alles, was man gerade an Büchern kopiert hat (in die Datei „book-store.service.ts“), rauslöschen.
  Dafür kommt hinein:

```
16    ngOnInit() {
17      this.books = this.bs.getAll();
18    }
```

- Diese Datei ist nun wieder viel übersichtlicher und kürzer geworden.

**3.2.)Das Service kann man auch in der „book-details.component.ts“ verwenden.**

Öffne diese Datei und füge ebenfalls in den „constructor“ dieses Service ein.

```
11
12    constructor(private bs: BookStoreService) {}
13
14    ngOnInit() {}
15
```

Natürlich kommt eine Fehlermeldung, weil der „import“ dafür fehlt.

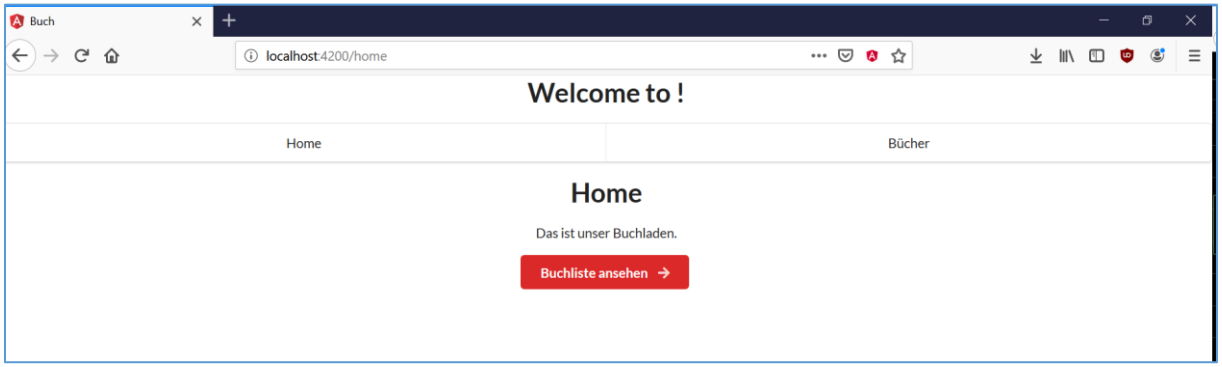
Füge daher in Zeile 4 diesen „import“ auch ein:

book-details.component.ts × | book-list.component.ts

src ▸ app ▸ book-details ▸ book-details.component.ts ▸ ...

```
1  import { Component, OnInit } from '@angular/core';
2
3  import { Book } from '../shared/book';
4  import { BookStoreService } from '../shared/book-store.service';
5
```

Speichern.



Quellen:

Woiwode, Malcher u.a. in: Angular; dpunkt-Verlag, 2018, S. 88-127