

# Angular Workshop

## Lab 1 – Setup:

Install angular cli by issuing the following command

**npm install -g @angular/cli**

**Note:** npm install -g angular/cli is a deprecated version

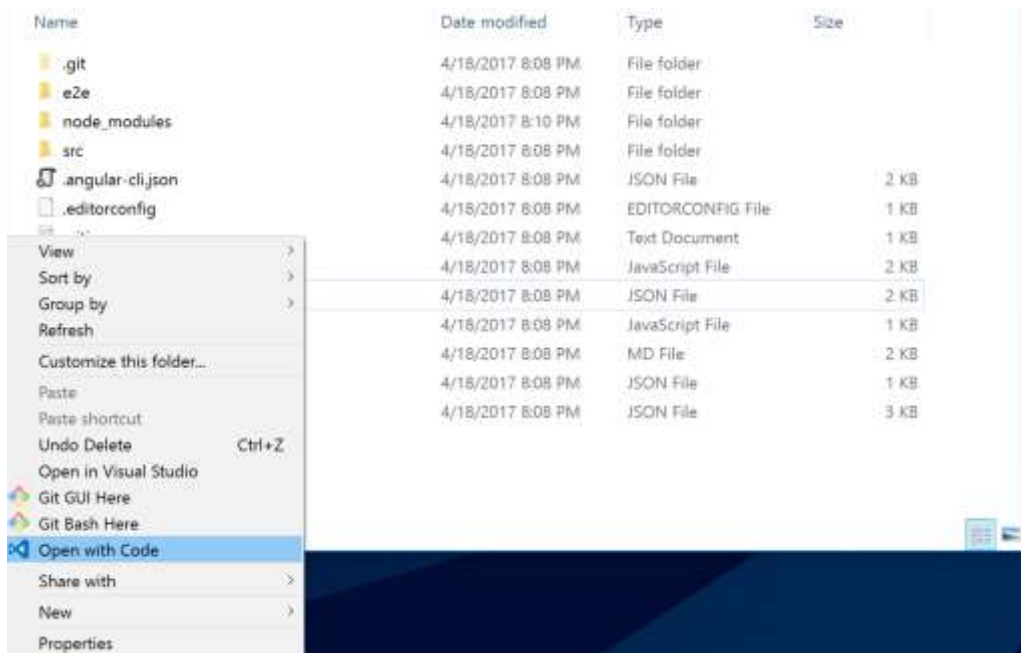
Create a new Angular project using the following command

**ng new Angular-Workshop**

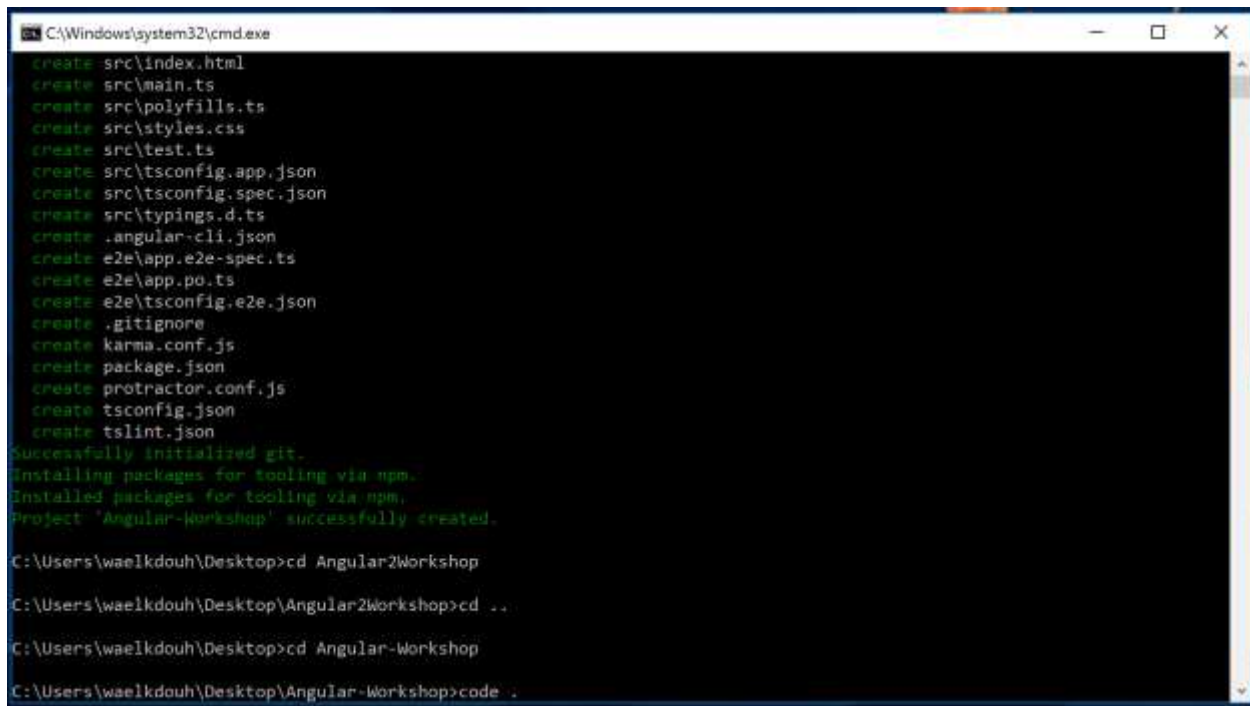
Navigate to the newly created project

**cd Angular-Workshop**

Open the project with vs code. You can either right click inside the folder and open with vs code



or you can type “code .” inside the project directory

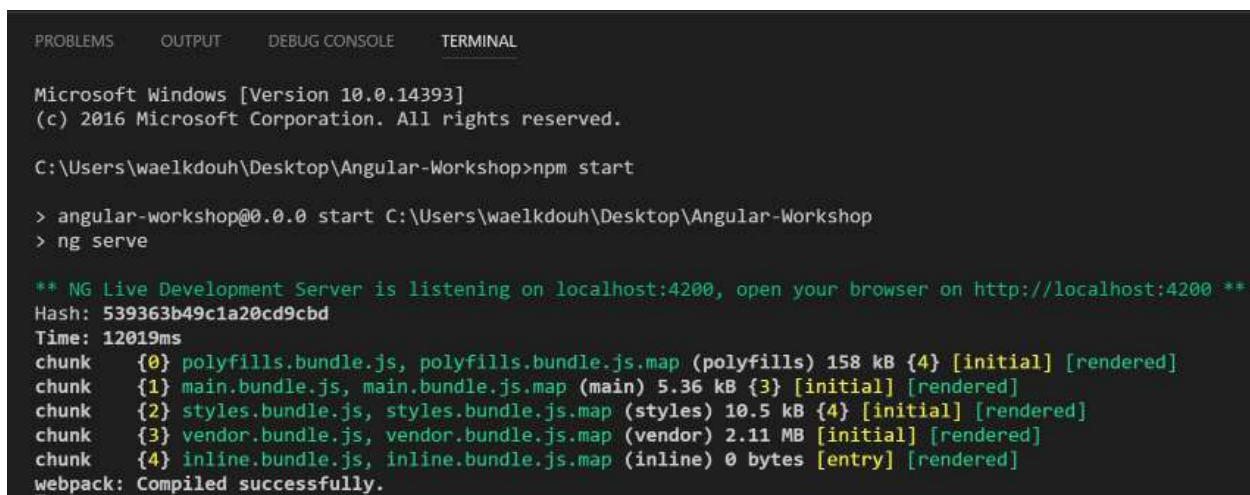


```
C:\Windows\system32\cmd.exe
create src\index.html
create src\main.ts
create src\polyfills.ts
create src\styles.css
create src\test.ts
create src\tsconfig.app.json
create src\tsconfig.spec.json
create src\typings.d.ts
create .angular-cli.json
create e2e\app.e2e-spec.ts
create e2e\app.po.ts
create e2e\tsconfig.e2e.json
create .gitignore
create karma.conf.js
create package.json
create protractor.conf.js
create tsconfig.json
create tslint.json
Successfully initialized git.
Installing packages for tooling via npm.
Installed packages for tooling via npm.
Project 'Angular-Workshop' successfully created.

C:\Users\waelkdouh\Desktop>cd Angular2Workshop
C:\Users\waelkdouh\Desktop\Angular2Workshop>cd ..
C:\Users\waelkdouh\Desktop>cd Angular-Workshop
C:\Users\waelkdouh\Desktop\Angular-Workshop>code .
```

Run the application using the following command inside the vs code command line

**npm start**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\waelkdouh\Desktop\Angular-Workshop>npm start

> angular-workshop@0.0.0 start C:\Users\waelkdouh\Desktop\Angular-Workshop
> ng serve

** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200 **
Hash: 539363b49c1a20cd9cbd
Time: 12019ms
chunk    {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 158 kB {4} [initial] [rendered]
chunk    {1} main.bundle.js, main.bundle.js.map (main) 5.36 kB {3} [initial] [rendered]
chunk    {2} styles.bundle.js, styles.bundle.js.map (styles) 10.5 kB {4} [initial] [rendered]
chunk    {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.11 MB [initial] [rendered]
chunk    {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry] [rendered]
webpack: Compiled successfully.
```

Open a browser and navigate to <http://localhost:4200>

**Tip:** You can click ctrl + hover over the link shown in the command line window to navigate to the url

---

**Welcome to app!!**



Here are some links to help you start:

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)

You have now successfully created/run your first Angular application. You can keep the application running as the server will support live page reloading when making any future changes to the code.

Navigate to app.component.ts file which is located under the app folder and modify the title property found under the AppComponent class from “app” to “My Tiny Library App”. **Save.**

Notice how the browser reflects the changes upon saving the file.

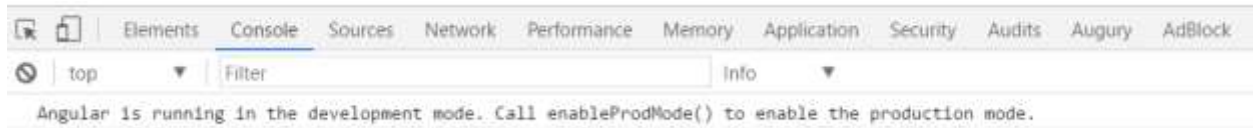
**Welcome to My Tiny Library App!!**



Here are some links to help you start:

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)

**Tip:** Make sure to always have the F12 tools open under your browser (navigate to the console tab) as Angular provides you with valuable debugging information there. Notice by default the Angular runtime is telling you that you are running in development mode. More on that later.



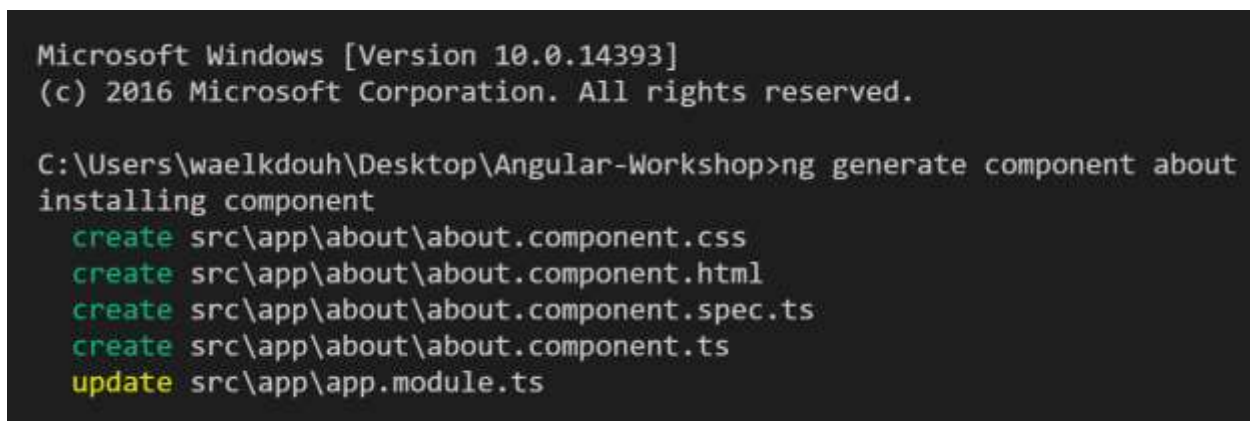
## Lab 2 – Template Containing a Component:

Open a new command window in VS Code by navigating to the terminal tab and clicking on the + icon . This way the first command window will keep the server running for live reload to work.

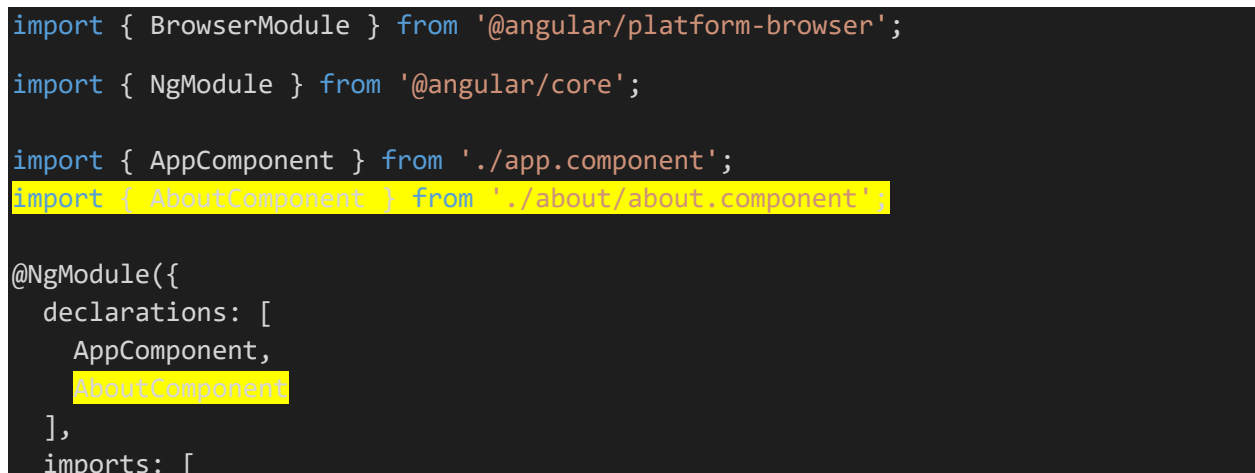


Add a new component to your page by issuing the following command  
**ng generate component about**

This will generate a new component in a new folder called about



Notice that the app.module.ts file has been updated to include the newly added component. Specifically, the AboutComponent has been added to the declarations array and the import statement for the same component has been added at the top of the file.



```
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

**Note:** When importing from a path, the './' operator refers to a **directory relative to that of file you are editing**.

Modify about.component.ts file to include a the pageTitle property that we will bind to in the associated template. Also modify the selector to my-about (we could have kept the default that got generated by the cli as well but its better to use a more descriptive selector).

Note: For the rest of the lab the **purple highlighting is used** to make it easier for you to spot code that you need to add/modify.

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

@Component({
  selector: 'my-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit {

  constructor() { }

  ngOnInit() {}

  pageTitle:string = 'About Me';
}
```

Modify about.component.html file to display the property we just added under the about.component.ts using one way interpolation:

```
<h3>{{pageTitle}}</h3>
<p>Welcome to your personal library </p>
```

Since we are bootstrapping our app using the app-root component we will need to include the newly added about component under the app.component.ts template. Modify the AppComponent template to include the AboutComponent selector. If you want you can also place the html under the app.component.html file and use the templateUrl instead of template. **Save.**

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

@Component({
  selector: 'app-root',
  template: `<h1>My Tiny Library App</h1>
             <my-about></my-about>
            `
})
export class AppComponent {}
```

The application should now display the about page in addition to the other artifacts you already had on the screen.

---

## My Tiny Library App

### About Me

Welcome to your personal library

**Note:** The about page will be moved later to a separate tab when we introduce Angular Material tabs.

## Lab 3 – Data Binding:

For this lab we will utilize a framework called [Angular Material](#) which offers UI controls for Angular applications (e.g. tabs controls, Modal box, etc.). We will also utilize Angular Material for theming our application. Other alternatives to Angular Material is Bootstrap which also offers similar capabilities.

Start by installing Angular Material and Angular CDK inside your project using the following command:

```
npm install --save @angular/cdk@2.0.0-beta.10
```

```
npm install --save @angular/material@2.0.0-beta.10
```

Some Material components depend on the Angular animations module to be able to do more advanced transitions. If you want these animations to work in your app, you have to install the @angular/animations module and include the BrowserAnimationsModule in your app.

```
npm install --save @angular/animations
```

Add the newly installed modules in addition to the FormsModule (this will be used for binding) under the app.module.ts file:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { MaterialModule } from '@angular/material';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { AboutComponent } from './about/about.component';

@NgModule({
  declarations: [
    AppComponent,
    AboutComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    MaterialModule,
    BrowserAnimationsModule,
  ],
})
```



```
providers: [],  
bootstrap: [AppComponent]  
})  
export class AppModule { }
```

Next you want to add a new component called collection which will be responsible for displaying your book library:

### **ng generate component collection**

This will generate a new component in a new folder called component

```
C:\Users\waelkdouh\Desktop\Angular-Workshop>ng g c collection  
Installing component  
create src\app\collection\collection.component.css  
create src\app\collection\collection.component.html  
create src\app\collection\collection.component.spec.ts  
create src\app\collection\collection.component.ts  
update src\app\app.module.ts
```

Update the collection.component.ts file to include the following:

- pageTitle: string, shown in the panel header
- books: Array<IBook>, shown in the table rows
- showOperatingHours: boolean, for message visibility and button text
- a method to toggle whether to show operating hours

Also change the selector name to my-collection. Also you can ignore the error message notifying you that IBook is an unknown type as you will add it next.

```
import { Component, OnInit } from '@angular/core';  
import { IBook } from '../ibook';  
  
@Component({  
  selector: 'my-collection',  
  templateUrl: './collection.component.html',  
  styleUrls: ['./collection.component.css']  
})  
export class CollectionComponent implements OnInit {  
  
  constructor() {  
    this.startTime = new Date();  
    this.startTime.setHours(10, 0);  
    this.endTime = new Date();  
    this.endTime.setHours(15, 0);  
  }  
}
```

```

ngOnInit(): void {

}

pageTitle:string = 'Books';

public books:Array<IBook> =
[
    {
        id: 1,
        title: "JavaScript - The Good Parts",
        author: "Douglas Crockford",
        isCheckedOut: true,
        rating: 3
    },
    {
        id: 2,
        title: "The Wind in the Willows",
        author: "Kenneth Grahame",
        isCheckedOut: false,
        rating: 4
    },
    {
        id: 3,
        title: "Pillars of the Earth",
        author: "Ken Follett",
        isCheckedOut: true,
        rating: 5
    },
    {
        id: 4,
        title: "Harry Potter and the Prisoner of Azkaban",
        author: "J. K. Rowling",
        isCheckedOut: false,
        rating: 5
    }
];

startTime:Date;

endTime:Date;

showOperatingHours:boolean = false; }

```

Add IBook interface using the following command  
**ng generate interface IBook**

```
C:\Users\waelkdouh\Desktop\Angular-Workshop>ng generate interface IBook
installing interface
  create src\app\ibook.ts
```

Modify the IBook interface:

```
export interface IBook {
  id: number,
  title: string,
  author: string,
  isCheckedOut: boolean,
  rating: number
}
```

Modify the collection.component.html file to include the following code:

```
<h3>{{pageTitle}}&nbsp;<md-slide-toggle class="plr-15" color="primary"
[(ngModel)]="showOperatingHours">{{showOperatingHours ? 'Hide' : 'Show'}} library
hours
</md-slide-toggle>
</h3>
<div [hidden]="!showOperatingHours">
  <md-card>
    <md-card-subtitle><strong>Operating Hours</strong></md-card-subtitle>
    <md-card-content>{{startTime}} - {{endTime}} (M-F)</md-card-content>
  </md-card>
</div>
<div>
  <md-list>
    <md-list-item *ngFor="let book of books">
      <md-icon md-list-icon>book</md-icon>
      <h3 md-line><strong>{{book.title}}</strong></h3>

      <p md-line>
        <span>{{book.author}}</span>
      </p>
      <p md-line>
        {{book.rating}}
      </p>
      <p md-line>
```

```

        <span [class]="book.isCheckedOut ? 'chip chip-danger' : 'chip chip-
success'">{{book.isCheckedOut ? 'Checked-Out' : 'Available'}}</span>
    </p>
</md-list-item>
</md-list>
</div>

```

Modify the collection.component.css to include the following code:

```

.mat-list .mat-list-item .mat-list-icon, .mat-nav-list .mat-list-item .mat-list-
icon {
    width: 48px;
    height: 48px;
    font-size: 48px;
    color: #b4bcc2;
}

.chip {
    display: inline;
    padding: .2em .6em .3em;
    font-size: 85%;
    font-weight: bold;
    line-height: 1;
    color: #ffffff;
    text-align: center;
    white-space: nowrap;
    vertical-align: baseline;
    border-radius: .25em;
}

.chip-success {
    background-color: #18bc9c;
}

.chip-danger {
    background-color: #e74c3c;
}

.add-btn {
    padding: 8px 65px;
}

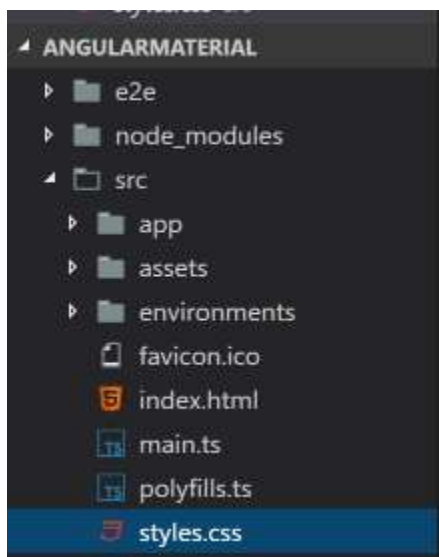
```

Modify the app.component.ts file to include the newly added collection component. Make sure you remove the about component.

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

@Component({
  selector: 'app-root',
  template: `<h1>My Tiny Library App</h1>
    <my-collection></my-collection>
  `
})
export class AppComponent {
}
```

To enable theming using angular material replace the styles.css file generated by the angular cli with the content provided to you in the lab 3 folder. **Save.**



Note: At this point the about page is not included in our application anymore. This is temporary as we will restore it later on when we introduce tabs into our application.

The application should now display the collection page. The two images below show the page with the slide toggle enabled/disabled.

## My Tiny Library App

Books ☐ Show library hours



### JavaScript - The Good Parts

Douglas Crockford

3

Checked-Out



### The Wind in the Willows

Kenneth Grahame

4

Available



### Pillars of the Earth

Ken Follett

5

Checked-Out



### Harry Potter and the Prisoner of Azkaban

J. K. Rowling

5

Available

## My Tiny Library App

Books ☒ Hide library hours

### Operating Hours

Mon Sep 25 2017 10:00:02 GMT-0700 (Pacific Daylight Time) - Mon Sep 25 2017 15:00:02 GMT-0700 (Pacific Daylight Time) (M-F)



### JavaScript - The Good Parts

Douglas Crockford

3

Checked-Out



### The Wind in the Willows

Kenneth Grahame

4

Available



### Pillars of the Earth

Ken Follett

5

Checked-Out



### Harry Potter and the Prisoner of Azkaban

J. K. Rowling

5

Available

## Lab 4 – Pipes:

Add a new pipe called rating-category.pipe

**ng generate pipe pipes/rating-category**

Notice that unlike the case of components, you have to be explicit about creating the pipes under a Pipes folder by providing the full path.

Modify the rating-category.pipe.ts file to return Poor (1-2), Fine (3-4), Excellent (5)

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'ratingCategory'
})
export class RatingCategoryPipe implements PipeTransform {

  transform(value: number): string {
    if (value <= 2) {
      return 'Poor';
    }
    if (value <= 4) {
      return 'Fine';
    }
    return 'Excellent';
  }
}
```

Update the collection.component.html file to use the date pipe (built in pipe into Angular) to the startTime and endTime. Also apply the newly introduced Rating Category pipe (custom Pipe). **Save.**

```
<h3>{{pageTitle}}&nbsp;<md-slide-toggle class="plr-15" color="primary"
[(ngModel)]="showOperatingHours">{{showOperatingHours ? 'Hide' : 'Show'}} library
hours</md-slide-toggle></h3>

<div [hidden]="!showOperatingHours">
  <md-card>
    <md-card-subtitle><strong>Operating Hours</strong></md-card-subtitle>
    <md-card-content>{{startTime | date:'shortTime'}} - {{endTime |
date:'shortTime'}} (M-F)</md-card-content>
  </md-card>
</div>
<div>
  <md-list>
    <md-list-item *ngFor="let book of books">
      <md-icon md-list-icon>book</md-icon>
      <h3 md-line><strong>{{book.title}}</strong></h3>


      <p md-line>
        <span>{{book.author}}</span>
      </p>
      <p md-line>
        {{book.rating | ratingCategory}}
      </p>
      <p md-line>
        <span [class]="book.isCheckedOut ? 'chip chip-danger' : 'chip chip-
success'">{{book.isCheckedOut ? 'Checked-Out' : 'Available'}}</span>
      </p>
    </md-list-item>
  </md-list>
</div>
```

The collection page should now show the rating numbers with text. In addition, the operating hours should now be replaced with a short time.



# My Tiny Library App


Books ☐ Show library hours

- 

JavaScript - The Good Parts

Douglas Crockford


Fine

Checked-Out
- 

The Wind in the Willows

Kenneth Grahame


Fine

Available
- 

Pillars of the Earth

Ken Follett

Excellent

Checked-Out
- 

Harry Potter and the Prisoner of Azkaban

J. K. Rowling

Excellent


Available

# My Tiny Library App

Books ☒ Hide library hours

Operating Hours

10:00 AM - 3:00 PM (M-F)




JavaScript - The Good Parts

Douglas Crockford

Fine

Checked-Out




The Wind in the Willows

Kenneth Grahame

Fine

Available




Pillars of the Earth

Ken Follett

Excellent

Checked-Out



Harry Potter and the Prisoner of Azkaban

J. K. Rowling

Excellent

Available

## Lab 5 – Communication between Parent and Child Components:

Add a Rating Component by issuing the following command

**ng generate component rating**

This will generate a new component in a new folder called rating:

```
C:\Users\waelkdouh\Desktop\Angular-Workshop>ng generate component rating.component
installing component
  create src\app\rating\rating.component.css
  create src\app\rating\rating.component.html
  create src\app\rating\rating.component.spec.ts
  create src\app\rating\rating.component.ts
  update src\app\app.module.ts
```

Modify the rating.component.ts file to include:

- @Input() for rating, @Input() for the book, @Output() for the click EventEmitter.
- ngOnInit() to log a message to the console with the value of the rating.
- ngOnChanges() to log a message to the console with the value of the rating.
- A method to emit the updated book rating via @Output() EventEmitter when the user changes the rating.

```
import { Component, OnInit, OnChanges, Input, Output, EventEmitter } from
 '@angular/core';

import { IBook } from '../ibook';

@Component({
  selector: 'my-rating',
  templateUrl: './rating.component.html',
  styleUrls: ['./rating.component.css']
})
export class RatingComponent implements OnInit, OnChanges {

  constructor() { }

  @Input() rating: number;
  @Input() book: IBook;
  @Output() ratingClicked: EventEmitter<IBook> = new EventEmitter<IBook>();

  ngOnInit(): void {
    //console.log("ngOnInit called for: " + this.rating.toString());
  }
}
```

```

ngOnChanges(): void {
  //console.log("The rating was just set to: " + this.rating.toString());
}

click(rating:number): void {
  this.book.rating = rating;
  this.ratingClicked.emit(this.book);
}
}

```

Modify the rating.component.css file to include the following styling:

```

.material-icons {
  cursor: pointer;
  color: rgba(103,58,183,.15);
}
.material-icons:hover {
  color: rgba(103,58,183,.35);
}
.material-icons.active {
  color: #000;
}

```

Modify the rating.component.html file to show the correct number of stars for the rating:

```

<div>
  <a (click)="click(1)"><i class="material-icons {{rating >= 1 ? 'active' : ''}}">star_rate</i></a>
  <a (click)="click(2)"><i class="material-icons {{rating >= 2 ? 'active' : ''}}">star_rate</i></a>
  <a (click)="click(3)"><i class="material-icons {{rating >= 3 ? 'active' : ''}}">star_rate</i></a>
  <a (click)="click(4)"><i class="material-icons {{rating >= 4 ? 'active' : ''}}">star_rate</i></a>
  <a (click)="click(5)"><i class="material-icons {{rating >= 5 ? 'active' : ''}}">star_rate</i></a>
</div>

```

Modify the collection.component.html to utilize the newly added rating component:

```
<h3>{{pageTitle}}&nbsp;<md-slide-toggle class="plr-15" color="primary"
(ngModel)="showOperatingHours">{{showOperatingHours ? 'Hide' : 'Show'}} library
hours</md-slide-toggle></h3>
<div [hidden]="!showOperatingHours">
  <md-card>
    <md-card-subtitle><strong>Operating Hours</strong></md-card-subtitle>
    <md-card-content>{{startTime | date:'shortTime'}} - {{endTime |
date:'shortTime'}} (M-F)</md-card-content>
  </md-card>
</div>
<div>
  <md-list>
    <md-list-item *ngFor="let book of books">
      <md-icon md-list-icon>book</md-icon>
      <h3 md-line><strong>{{book.title}}</strong></h3>
      <p md-line>
        <span>{{book.author}}</span>
      </p>
      <p md-line>
        <my-rating [rating]="book.rating" [book]="book"
          (ratingClicked)="onRatingUpdate($event)">
        </my-rating>
      </p>
      <p md-line>
        <span [class]="book.isCheckedOut ? 'chip chip-danger' : 'chip chip-
success'">{{book.isCheckedOut ? 'Checked-Out' : 'Available'}}</span>
      </p>
    </md-list-item>
  </md-list></div>
```

Modify collection.component.ts to add a method to show the rating update message confirmation and listen to the click event that is triggered by the rating component (the child component in this example). **Save.**

```
import { Component, OnInit } from '@angular/core';
import { IBook } from 'app/ibook';
import { MdSnackBar } from "@angular/material";

@Component({
  selector: 'my-collection',
  templateUrl: './collection.component.html',
  styleUrls: ['./collection.component.css']
})
export class CollectionComponent implements OnInit {

  constructor(private _snackBar: MdSnackBar) {
    this.startTime = new Date();
    this.startTime.setHours(10, 0);
    this.endTime = new Date();
    this.endTime.setHours(15, 0);
  }

  ngOnInit(): void {

  }

  pageTitle: string = 'Books';

  public books: Array<IBook> =
  [
    {
      id: 1,
      title: "JavaScript - The Good Parts",
      author: "Douglas Crockford",
      isCheckedOut: true,
      rating: 3
    },
    {
      id: 2,
      title: "The Wind in the Willows",
      author: "Kenneth Grahame",
      isCheckedOut: false,
      rating: 4
    },
    {

```

```

        id: 3,
        title: "Pillars of the Earth",
        author: "Ken Follett",
        isCheckedOut: true,
        rating: 5
    },
    {
        id: 4,
        title: "Harry Potter and the Prisoner of Azkaban",
        author: "J. K. Rowling",
        isCheckedOut: false,
        rating: 5
    }
];

startTime: Date;

endTime: Date;

showOperatingHours: boolean = false;

updateMessage(message: string, type: string): void {
    if (message) {
        this._snackBar.open(`${type}: ${message}`, 'DISMISS', {
            duration: 3000
        });
    }
}


onRatingUpdate(book: IBook): void {
    this.updateMessage(book.title, " Rating has been updated");
}
}


```


The collection page should now display stars reflecting the rating. You should also see a message at the bottom of the page upon changing the rating a book.


## My Tiny Library App

Books ☐ Show library hours

- 

JavaScript - The Good Parts  
Douglas Crockford  
★ ★ ☆ ☆ ☆  
Checked-Out
- 

The Wind in the Willows  
Kenneth Grahame  
★ ★ ★ ★ ☆  
Available
- 

Pillars of the Earth  
Ken Follett  
★ ★ ★ ★ ★  
Checked-Out
- 

Harry Potter and the Prisoner of Azkaban  
J. K. Rowling  
★ ★ ★ ★ ★  
Available

Rating has been updated: JavaScript - The Good Parts

DISMISS

## Lab 6 – Build a Service:

Generate a new service using the following command  
**ng generate service services/data**

Add one method to the Data Service called `getBooks`:

- This method will return an array of `IBooks`
- Move the array of books from the `collection.component.ts` file into `getBooks()` method under the `DataService` as the data doesn't belong under the component (Note that it doesn't belong under the service either, but rather the data must be stored on some persistent storage like a DB. We will modify that later in the lab).
- Decorate the Data Service with `@Injectable` and add the required import. This makes the service injectable using the built in Dependency Injection under Angular.

Here is the content of `data.service.ts` file:

```
import { Injectable } from '@angular/core';
import { IBook } from "../ibook";

@Injectable()
export class DataService {

  getBooks(): Array<IBook> {
    return [
      {
        id: 1,
        title: "JavaScript - The Good Parts",
        author: "Douglas Crockford",
        isCheckedOut: true,
        rating: 3
      },
      {
        id: 2,
        title: "The Wind in the Willows",
        author: "Kenneth Grahame",
        isCheckedOut: false,
        rating: 4
      },
      {
        id: 3,
```



```

        title: "Pillars of the Earth",
        author: "Ken Follett",
        isCheckedOut: true,
        rating: 5
      },
      {
        id: 4,
        title: "Harry Potter and the Prisoner of Azkaban",
        author: "J. K. Rowling",
        isCheckedOut: false,
        rating: 5
      }
    ];
  }
}

```

Modify the collection.component.ts file:

- Constructor should now expect an instance of DataService via dependency injection
- ngOnInit should call getBooks() method in Data Service and assign the result to its books property
- Add the required import for the Data Service (or you can simply use the productivity tip introduced under lab 2 which helps you automatically include the necessary import statement)

```

import { Component, OnInit } from '@angular/core';
import { IBook } from '../ibook';
import { MdSnackBar } from "@angular/material";
import { DataService } from '../services/data.service';
@Component({
  selector: 'my-collection',
  templateUrl: './collection.component.html',
  styleUrls: ['./collection.component.css']
})
export class CollectionComponent implements OnInit {

  constructor(private _dataService: DataService, private _snackBar: MdSnackBar)
  {
    this.startTime = new Date();
    this.startTime.setHours(10, 0);
    this.endTime = new Date();
    this.endTime.setHours(15, 0);
  }
}

```

```
ngOnInit(): void {  
    this.books = this._dataService.getBooks();  
}  
  
pageTitle: string = 'Books';  
  
public books: Array<IBook>;  
  
startTime: Date;  
  
endTime: Date;  
  
showOperatingHours: boolean = false;  
  
updateMessage(message: string, type: string): void {  
    if (message) {  
        this._snackBar.open(`${type}: ${message}`, 'DISMISS', {  
            duration: 3000  
        });  
    }  
}  
  
onRatingUpdate(book: IBook): void {  
    this.updateMessage(book.title, " Rating has been updated");  
}  
}
```

Modify `app.component.ts` to include the data service as a provider as it will be utilized by the collection component (Note: you could have also added the provider directly under the collection controller if you know it will only be utilized there. Alternatively you could have also added the `DataService` to the `AppModule` if you wanted to use the Singleton Pattern for the `DataService`). **Save.**

```
import { Component } from '@angular/core';
import { DataService } from '../services/data.service';

@Component({
  selector: 'app-root',
  template: `<h1>My Tiny Library App</h1>
    <my-collection></my-collection>
  `,
  providers: [DataService]
})
export class AppComponent {
}
```

You shouldn't expect the UI to change, but the books are now removed from the collection component to the newly introduced data service.

## Lab 7 – Http Service:

In the previous lab we moved the array of books from the collection component to the data service. In this lab we will remove the books array from the data service and instead point our data service to an Asp.Net Core WebApi which returns an array of books in json format.

Remove the array of books from the data service. Modify the data.service.ts file to include code that is capable of making a server side call instead of simply returning a hard coded array of books. Here are the steps that are required:

- Add a constructor that expects an instance of Http service
- Modify getBooks() to return Observable<IBook[]>
- getBooks() calls \_http.get() and then map() the response
- Add required import for Http and Response from @angular/http
- Add required import for Observable from rxjs/Observable
- Add required import for observable operators

```

import { Injectable } from '@angular/core';
import { IBook } from "../ibook";
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/catch';
import 'rxjs/add/observable/throw';

@Injectable()
export class DataService {

    // Asp.Net Core WebApi
    private _booksUrl = 'http://waelbookservice.azurewebsites.net/api/books';

    constructor(private _http: Http) { }

    getBooks(): Observable<IBook[]> {
        return this._http.get(this._booksUrl)
            .map((response: Response) => {
                let data: IBook[] = <IBook[]>response.json();
                localStorage.setItem('books', JSON.stringify(data));
                return data;
            })
            .catch(this.handleError);
    }

    private handleError(error: any) {
        let errMsg = (error.message) ? error.message : error.status ?
        `${error.status} - ${error.statusText}` : 'Server error';
        console.error(errMsg);
        return Observable.throw(errMsg);
    }
}

```

Update the app.module.ts file to include the HttpClientModule by including it in the imports array:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { MaterialModule } from '@angular/material';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { AboutComponent } from './about/about.component';
import { CollectionComponent } from './collection/collection.component';
import { RatingCategoryPipe } from './rating-category.pipe';
import { RatingComponent } from './rating/rating.component';
import { HttpClientModule } from '@angular/http';

@NgModule({
  declarations: [
    AppComponent,
    AboutComponent,
    CollectionComponent,
    RatingCategoryPipe,
    RatingComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    MaterialModule,
    BrowserAnimationsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Modify the collection.component.ts file to include a getbooks() method which returns an Observable. **Save.**

```

import { Component, OnInit } from '@angular/core';
import { IBook } from 'app/ibook';
import { MdSnackBar } from '@angular/material';
import { DataService } from 'app/data.service';

@Component({
  selector: 'my-collection',
  templateUrl: './collection.component.html',
  styleUrls: ['./collection.component.css']
})
export class CollectionComponent implements OnInit {

  constructor(private _dataService: DataService, private _snackBar: MdSnackBar)
  {

    this.startTime = new Date();
    this.startTime.setHours(10, 0);
    this.endTime = new Date();
    this.endTime.setHours(15, 0);
  }

  ngOnInit(): void {
    this.getBooks();
  }

  pageTitle: string = 'Books';

  public books: Array<IBook>;

  startTime: Date;

  endTime: Date;

  showOperatingHours: boolean = false;

  getBooks(): void {
    this._dataService.getBooks()
      .subscribe(
        books => this.books = books,
        error => this.updateMessage(<any>error, 'ERROR'));
  }

  updateMessage(message: string, type: string): void {
    if (message) {
      this._snackBar.open(`${type}: ${message}`, 'DISMISS', {

```

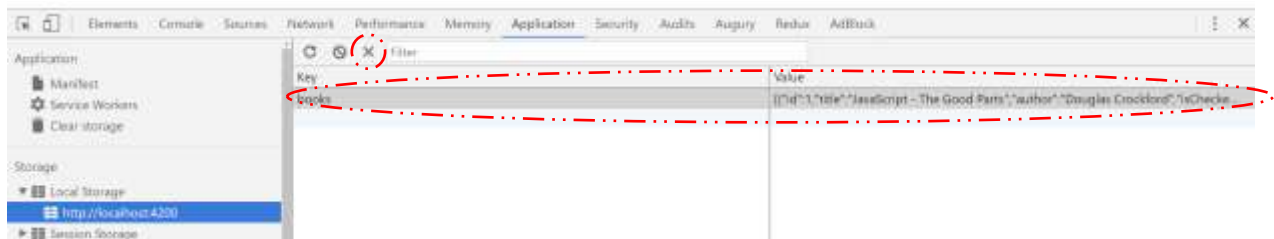
```

        duration: 3000
    });
}

onRatingUpdate(book: IBook): void {
    this.updateMessage(book.title, " Rating has been updated");
}
}

```

In order to test if the updated data service is working against the restful api, load the F12 tools and delete the books array which is stored in the local storage as shown in the image below. Refresh your browser which will attempt to fetch the data from the server as the local storage doesn't include the books locally in the browser cache anymore.





## Lab 8 – Routing:

Add app.routing.ts file to the app folder. This file will include the routing logic in our application.

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AboutComponent } from "../about/about.component";
import { CollectionComponent } from "../collection/collection.component";

const routes: Routes = [
  {
    path: 'about',
    component: AboutComponent
  },
  {
    path: 'collection',
    component: CollectionComponent
  },
  {
    path: '',
    redirectTo: '/about',
    pathMatch: 'full'
  }
];

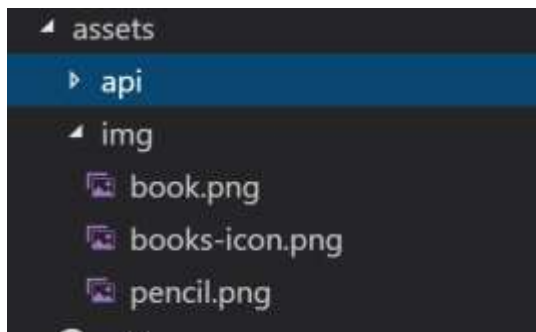
export const routing: ModuleWithProviders = RouterModule.forRoot(routes);
```

Add a tabs components which will be used to create a navigation menu.

### ng generate component tabs

```
C:\Users\waelkdouh\Desktop\Angular-Workshop>ng generate component tabs
installing component
  create src\app\tabs\tabs.component.css
  create src\app\tabs\tabs.component.html
  create src\app\tabs\tabs.component.spec.ts
  create src\app\tabs\tabs.component.ts
  update src\app\app.module.ts
```

Include the img folder provided to you as an asset in lab 8. The updated project structure should now look like this:



Modify the tabs.component.ts to include the following code. The routing collection now includes two routes which allow navigation between the two tabs.

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

@Component({
  selector: 'app-tabs',
  templateUrl: './tabs.component.html',
  styleUrls: ['./tabs.component.css']
})
export class TabsComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  navLinks:Array<object> = [
    {
      path: 'about',
      label: 'ABOUT ME'
    },
    {
      path: 'collection',
      label: 'MY COLLECTION'
    }
  ]
}
```

```

    }
  ];
}

```

Modify the tabs.component.html to include the following code:

```

<nav md-tab-nav-bar>
  <a md-tab-link
    *ngFor="let link of navLinks"
    [hidden]="!link.path"
    [routerLink]="link.path"
    routerLinkActive #rla="routerLinkActive"
    [active]="rla.isActive">
    {{link.label}}
  </a>
</nav>
<router-outlet></router-outlet>

```

Replace the content of the app.component.html with the code that includes a menu to navigate between the collection page and the about page:

```

<div class="container">
  <md-toolbar color="warn">
    <span>{{title}}</span>
    <span class="fill-remaining-space"></span>
    
  </md-toolbar>
  <app-tabs></app-tabs>
</div>

```

Modify app.component.ts to point to the updated app.component.html which now included the newly created tabs component. Also make sure to the template property as you can use either template or templateUrl but not both:

```
import { Component } from '@angular/core';
import { DataService } from './data.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  providers: [DataService]
})
export class AppComponent {
  title = 'Material Library App';
}
```

Modify app.module.ts:

- import { routing } from './app.routing';
- include routing in the imports property of @NgModule
- include RouterModule in the imports property of @NgModule

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { MaterialModule } from '@angular/material';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { AboutComponent } from './about/about.component';
import { CollectionComponent } from './collection/collection.component';
import { RatingCategoryPipe } from './rating-category.pipe';
import { RatingComponent } from './rating/rating.component';
import { HttpModule } from '@angular/http';
import { TabsComponent } from './tabs/tabs.component';
import { RouterModule } from '@angular/router';
import { routing } from 'app/app.routing';
```

```
@NgModule({
  declarations: [
    AppComponent,
    AboutComponent,
    CollectionComponent,
    RatingCategoryPipe,
    RatingComponent,
    TabsComponent
```

```

],
imports: [
  BrowserModule,
  FormsModule,
  MaterialModule,
  BrowserAnimationsModule,
  HttpClientModule,
  RouterModule,
  routing
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }

```

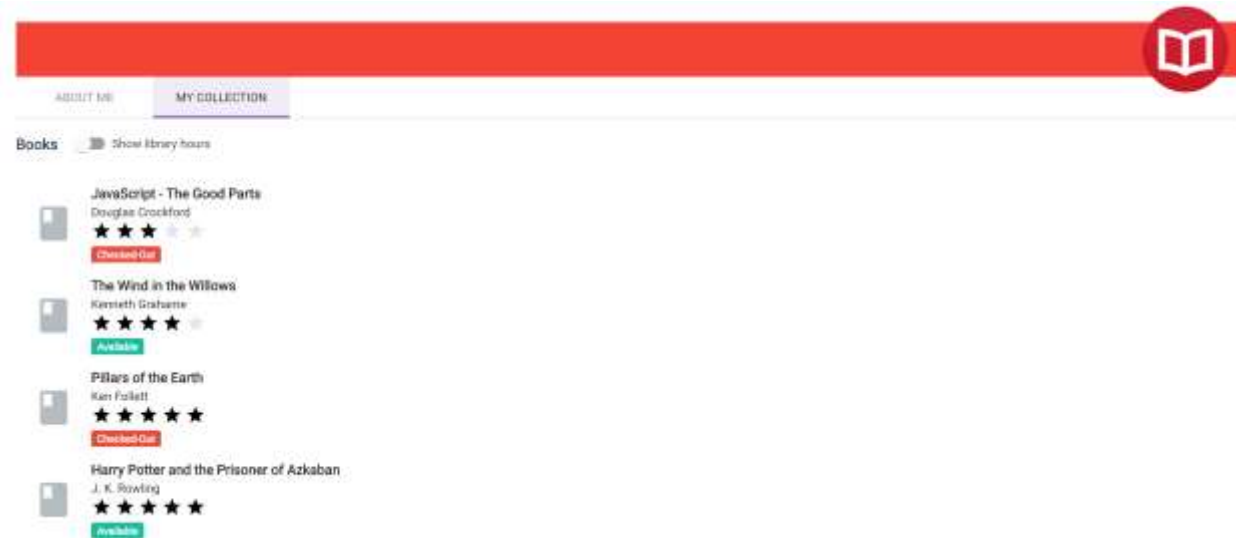
Remove the selector from the collection.component.ts file as the collection page will now be navigated to using the tabs instead of including it as part of the app.component.ts file. **Save.**

```

@Component({
  // selector property not required anymore
  templateUrl: './collection.component.html',
  styleUrls: ['./collection.component.css']
})
export class CollectionComponent implements OnInit {
...

```

You should now have two tabs with the first tab showing the about page and the second tab showing the collection page.



## Lab 9 – Passing Parameters To a Route and Activating a Route with Code:

In this lab we will show the details of a book on a separate page. We will demonstrate that using two different methods. The first is using a modal box and the second is navigating to a totally new page (you would choose one or the other, but we are including both to demonstrate passing parameters as well activating a route with code).

Add a book-detail component as follows:

### **ng generate component book-detail**

Modify data.service.ts to include a `getBook()` method which would allow fetching a single book. Also we will start preparing our application to persisting changes using HTML5 local storage (in reality you would persist to the database). Here is the updated data.service.ts file:

```
import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import { IBook } from '../ibook';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/catch';

@Injectable()
export class DataService {

  private _booksUrl = 'http://waelbookservice.azurewebsites.net/api/books'

  constructor(private _http: Http) { }

  getBooks(): Observable<IBook[]> {
    let localBooks = localStorage.getItem('books');
    if (localBooks) {
      return Observable.create(observer => {
        observer.next(JSON.parse(localBooks));
      });
    }
    return this._http.get(this._booksUrl)
      .map((response: Response) => {
        let data: IBook[] = <IBook[]>response.json();
```



```

        localStorage.setItem('books', JSON.stringify(data));
        return data;
    })
    .catch(this.handleError);
}

getBook(id: number): Observable<IBook> {
    return this.getBooks()
        .map((books: IBook[]) => books.find(b => b.id === id))
        // .do(data => console.log( JSON.stringify(data)))
        .catch(this.handleError);
}

updateBook(book: IBook): Observable<IBook[]> {
    const local:string = localStorage.getItem('books');
    if (!local) return Observable.throw('Local storage not found.');
```

```

    let localBooks:IBook[] = JSON.parse(local);
    localBooks = localBooks.map(b => {
        if (b.id === book.id) {
            return Object.assign(b, book);
        }
        return b;
    });
    localStorage.setItem('books', JSON.stringify(localBooks));
    return Observable.create(observer => {
        observer.next(localBooks);
    });
}

private handleError(error: any) {
    let errMsg = (error.message) ? error.message : error.status ?
`${error.status} - ${error.statusText}` : 'Server error';
    console.error(errMsg);
    return Observable.throw(errMsg);
}
}

```

## Modify book-detail.component.ts file

```
import { Component, OnInit, OnDestroy, Input, Output } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { Subscription } from 'rxjs/Subscription';
import { IBook } from "../ibook";
import { DataService } from "../services/data.service";
import { MdSnackBar } from '@angular/material';

@Component({
  templateUrl: './book-detail.component.html',
  styleUrls: ['./book-detail.component.css'],
  providers: [DataService]
})
export class BookDetailComponent implements OnInit, OnDestroy {
  bookId: number;

  book: IBook;

  sub: Subscription;

  constructor(
    private _route: ActivatedRoute,
    private _router: Router,
    private _dataService: DataService,
    private _snackBar: MdSnackBar) {}

  ngOnInit(): void {
    if (!this.bookId) {
      this.sub = this._route.params.subscribe(
        params => {
          let id = +params['id'];
          this.getBook(id);
        }
      );
      return;
    }
    this.getBook(this.bookId);
  }

  ngOnDestroy(): void {
    if (this.sub) {
      this.sub.unsubscribe();
    }
  }
}
```

```

getBook(id: number): void {
    this._dataService.getBook(id).subscribe(
        book => this.book = book,
        error => this.updateMessage(<any>error, 'Error'));
}

onRatingUpdate(book: IBook): void {
    this.updateBook(book);
}

updateMessage(message:string, type:string, actionText:string = 'DISMISS') {
    if (message) {
        this._snackBar.open(`${type}: ${message}`, actionText, {
            duration: 3000
        });
    }
}

return(): void {
    this._router.navigate(['/collection']);
}

updateBook(book: IBook): void {
    this._dataService.updateBook(book)
        .subscribe(
            books => {
                this._snackBar.open(`"${book.title}" has been updated!`, 'DISMISS', {
                    duration: 3000
                });
            }, error => this.updateMessage(<any>error, 'ERROR'));
}
}

```

Modify book-detail.component.html:

```
<div *ngIf="book">
  <md-card>
    <md-card-header>
      <md-card-title><h4>{{book.title}}</h4></md-card-title>
      <md-card-subtitle>{{book.author}}</md-card-subtitle>
      
    </md-card-header>
    <md-card-content>
      <div>
        <label><strong>Title:</strong></label>
        <span>{{book.title}}</span>
      </div>
      <div>
        <label><strong>Author:</strong></label>
        <span>{{book.author}}</span>
      </div>
      <div>
        <label><strong>Checked Out:</strong></label>
        <span>{{book.isCheckedOut ? 'Yes' : 'No'}}</span>
      </div>
      <div>
        <label><strong>Rating:</strong></label>
        <my-rating [rating]="book.rating" [book]="book"
(ratingClicked)="onRatingUpdate($event)"></my-rating>
      </div>
    </md-card-content>
    <md-card-actions>
      <div class="text-right">
```

```

        <button md-button md-dialog-close *ngIf="bookId"><i class="material-
icons">close</i>CLOSE</button>
        <button md-button (click)="return()" *ngIf="!bookId"><i class="material-
icons">keyboard_arrow_left</i>RETURN</button>
    </div>
</md-card-actions>
</md-card>
</div>

```

Modify book-detail.component.css

```

.mat-card {
    margin-top: 15px;
}
.mat-card-avatar {
    width: 64px;
    height: 64px;
}
.mat-card-header {
    margin-bottom: 10px;
}
.mat-card-header h4 {
    margin-bottom: 0;
    margin-top: 5px;
    font-size: 18px;
}

```

Modify the collection.component.ts to enable users to navigate to the book details page. Also modify the onRatingUpdate method to utilize the newly updated data service which is now persisting on the local storage:

```
import { Component, OnInit } from '@angular/core';
import { IBook } from '../ibook';
import { MdSnackBar, MdDialog } from "@angular/material";
import { DataService } from '../services/data.service';
import { Router } from "@angular/router";
import { BookDetailComponent } from '../book-detail/book-detail.component';

@Component({
  templateUrl: './collection.component.html',
  styleUrls: ['./collection.component.css']
})
export class CollectionComponent implements OnInit {

  constructor(private _dataService: DataService, private _snackBar: MdSnackBar,
private _dialog: MdDialog, private _router: Router) {

    this.startTime = new Date();
    this.startTime.setHours(10, 0);
    this.endTime = new Date();
    this.endTime.setHours(15, 0);
  }

  ngOnInit(): void {
    this.getBooks();
  }

  pageTitle: string = 'Books';

  public books: Array<IBook>;

  startTime: Date;

  endTime: Date;

  showOperatingHours: boolean = false;

  getBooks(): void {
    this._dataService.getBooks()
      .subscribe(
```

```

        books => this.books = books,
        error => this.updateMessage(<any>error, 'ERROR'));
    }

    updateMessage(message: string, type: string): void {
        if (message) {
            this._snackBar.open(`${type}: ${message}`, 'DISMISS', {
                duration: 3000
            });
        }
    }
}

```

```

onRatingUpdate(book: IBook): void {
    this.updateBook(book);
    this.updateMessage(book.title, " Rating has been updated");
}

```

```

updateBook(book: IBook): void {
    this._dataService.updateBook(book)
        .subscribe(
            books => {
                this.books = books;
                this._snackBar.open(`${book.title}" has been updated!`,
                    'DISMISS', {
                        duration: 3000
                    });
            }, error => this.updateMessage(<any>error, 'ERROR'));
}

```

```

openDialog(bookId: number): void {
    let config = { width: '650px', height: '400px', position: { top: '50px' } };
}

```

```

    let dialogRef = this._dialog.open(BookDetailComponent, config);

```

```

    dialogRef.componentInstance.bookId = bookId;

```

```

    dialogRef.afterClosed().subscribe(res => {
        this.getBooks();
    });
}

```

```

openRoute(bookId: number): void {
    this._router.navigate(['/collection', bookId]);
} }

```

Modify the collection.component.html to enable users to navigate to the book details page:

```
<h3>{{pageTitle}}&nbsp;<md-slide-toggle class="plr-15" color="primary"
[(ngModel)]="showOperatingHours">{{showOperatingHours ? 'Hide' : 'Show'}} library
hours</md-slide-toggle></h3>
<div [hidden]="!showOperatingHours">
  <md-card>
    <md-card-subtitle><strong>Operating Hours</strong></md-card-subtitle>
    <md-card-content>{{startTime | date:'shortTime'}} - {{endTime |
date:'shortTime'}} (M-F)</md-card-content>
  </md-card>
</div>
<div>
  <md-list>
    <md-list-item *ngFor="let book of books">
      <md-icon md-list-icon>book</md-icon>
      <h3 md-line><strong>{{book.title}}</strong></h3>
      <div>
        <button md-button (click)="openDialog(book.id)">
          <i class="material-icons">pageview</i>
          Dialog
        </button>
        <button md-button (click)="openRoute(book.id)">
          <i class="material-icons">pageview</i>
          Route
        </button>
      </div>
      <p md-line>
        <span>{{book.author}}</span>
      </p>
      <p md-line>
        <my-rating [rating]="book.rating" [book]="book"
(ratingClicked)="onRatingUpdate($event)">
        </my-rating>
      </p>
      <p md-line>
        <span [class]="book.isCheckedOut ? 'chip chip-danger' : 'chip chip-
success'">{{book.isCheckedOut ? 'Checked-Out' : 'Available'}}</span>
      </p>
    </md-list-item>
  </md-list>
</div>
```



Modify the app.routing.ts to include the individual book-detail path. **Save.**

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AboutComponent } from "../about/about.component";
import { CollectionComponent } from "../collection/collection.component";
import { BookDetailComponent } from "../book-detail/book-detail.component";

const routes: Routes = [
  {
    path: 'about',
    component: AboutComponent
  },
  {
    path: 'collection',
    component: CollectionComponent
  },
  {
    path: 'collection/:id',
    component: BookDetailComponent
  },
  {
    path: '',
    redirectTo: '/about',
    pathMatch: 'full'
  }
];

export const routing: ModuleWithProviders = RouterModule.forRoot(routes);
```


Note: At this point modifying the book rating should now persist across browser refreshes since we are utilizing Html 5 localStorage.

Your application should now provide you with two buttons which allows you to navigate to two different pages with the detailed book information.

ABOUT ME

MY COLLECTION


Books ☐ Show library hours



JavaScript - The Good Parts  
Douglas Crockford  
★ ★ ☆ ☆ ☆  
Checked Out

Dialog


Route



The Wind in the Willows  
Kenneth Grahame  
★ ★ ★ ★ ☆  
Available

Dialog


Route



Pillars of the Earth  
Ken Follett  
★ ★ ★ ★ ★  
Checked Out

Dialog

Route



Harry Potter and the Prisoner of Azkaban  
J. K. Rowling  
★ ★ ★ ★ ★  
Available


Dialog

Route

ABOUT ME

MY COLLECTION


Books ☐ Show library hours



JavaScript - The Good Parts  
Douglas Crockford  
★ ★ ☆ ☆ ☆  
Checked Out

Dialog


Route



The Wind in the Willows  
Kenneth Grahame  
★ ★ ★ ★ ☆  
Available

Dialog


Route



Pillars of the Earth  
Ken Follett  
★ ★ ★ ★ ★  
Checked Out

Dialog


Route



Harry Potter and the Prisoner of Azkaban  
J. K. Rowling  
★ ★ ★ ★ ★  
Available

Dialog

Route




JavaScript - The Good Parts  
Douglas Crockford

Title: JavaScript - The Good Parts  
Author: Douglas Crockford  
Checked Out? Yes  
Rating:  
★ ★ ☆ ☆ ☆

X CLOSE

ABOUT ME

MY COLLECTION



JavaScript - The Good Parts  
Douglas Crockford

Title: JavaScript - The Good Parts  
Author: Douglas Crockford  
Checked Out? Yes  
Rating:  
★ ★ ☆ ☆ ☆

← RETURN

## Lab 10 – Protecting Routes with Guards:

Add a book guard service using the following command:

**ng generate service guards/book-guard**

Modify book-guard.service.ts

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, Router } from '@angular/router';
@Injectable()
export class BookGuardService implements CanActivate {
  constructor(private _router: Router) {
  }

  canActivate(route: ActivatedRouteSnapshot): boolean {
    let id = +route.url[1].path;
    if (isNaN(id) || id < 1) {
      // start a new navigation to redirect to list page
      this._router.navigate(['/collection']);
      // abort current navigation
      return false;
    };
    return true;
  }
}
```

Modify app.routing.ts to to apply the guard to the book detail route:

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AboutComponent } from "../about/about.component";
import { CollectionComponent } from "../collection/collection.component";
import { BookDetailComponent } from "../book-detail/book-detail.component";
import { BookGuardService } from "../guards/book-guard.service";

const routes: Routes = [
  {
    path: 'about',
    component: AboutComponent
  },
  {
    path: 'collection',
    component: CollectionComponent
  },
  {
    path: 'collection/:id',
    canActivate: [BookGuardService],
    component: BookDetailComponent
  },
  {
    path: '',
    redirectTo: '/about',
    pathMatch: 'full'
  }
];

export const routing: ModuleWithProviders = RouterModule.forRoot(routes);
```

Modify app.module.ts to include the BookDetailGuard service. **Save.**

Note: If you attempt to navigate to <http://localhost:4200/collection/-1> you will be rerouted to the main collection page.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { MaterialModule } from '@angular/material';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { AboutComponent } from './about/about.component';
import { CollectionComponent } from './collection/collection.component';
import { RatingCategoryPipe } from './rating-category.pipe';
import { RatingComponent } from './rating/rating.component';
import { HttpModule } from '@angular/http';
import { routing } from 'app/app.routing';
import { TabsComponent } from './tabs/tabs.component';
import { RouterModule } from '@angular/router';
import { BookDetailComponent } from './book-detail/book-detail.component';
import { BookGuardService } from './guards/book-guard.service';
@NgModule({
  declarations: [
    AppComponent,
    AboutComponent,
    CollectionComponent,
    RatingCategoryPipe,
    RatingComponent,
    TabsComponent,
    BookDetailComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    MaterialModule,
    BrowserAnimationsModule,
    HttpModule,
    RouterModule,
    routing
  ],
  providers: [BookGuardService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

## Lab 11 – Adding Forms:

Add a new-book component using the following command:

**ng generate component new-book**

Modify the new-book.component.ts file:

```
import { Component, OnInit } from '@angular/core';
import { IBook } from '../ibook';
import { MdDialogRef } from '@angular/material';

@Component({
  selector: 'app-new-book',
  templateUrl: './new-book.component.html',
  styleUrls: ['./new-book.component.css']
})
export class NewBookComponent implements OnInit {

  book:IBook;

  constructor(private _dialogRef: MdDialogRef<NewBookComponent>) { }

  ngOnInit() {
    this.book = {
      id: 0,
      title: '',
      author: '',
      isCheckedOut: false,
      rating: 0
    }
  }

  cancel(): void {
    this._dialogRef.close();
  }

  save(): void{
    this._dialogRef.close(this.book);
  }
}
```

Modify the new-book.component.css file:

```
.mat-card-avatar {  
  width: 64px;  
  height: 64px;  
  -webkit-border-radius: 0;  
  -moz-border-radius: 0;  
  border-radius: 0;  
}  
.mat-card-header {  
  margin-bottom: 25px;  
}  
.mat-card-header h4 {  
  margin-bottom: 0;  
  margin-top: 15px;  
  font-size: 24px;  
}  
.mat-input-container {  
  width: 100%;  
  line-height: 2.23;  
  margin-bottom: 15px;  
}  
p.rating-label {  
  color: rgba(0,0,0,.38);  
}  
.mat-slide-toggle {  
  height: 45px;  
  line-height: 45px;  
  margin: 10px 0;  
}
```

Modify the new-book.component.html file:

```
<form #newBookForm="ngForm">
  <md-card>
    <md-card-header>
      <md-card-title><h4>Add New Book</h4></md-card-title>
      
    </md-card-header>
    <md-card-content>
      <md-input-container>
        <input mdInput placeholder="Book Title" [(ngModel)]="book.title"
name="title" required />
      </md-input-container>
      <md-input-container>
        <input mdInput placeholder="Author" [(ngModel)]="book.author"
name="author" required />
      </md-input-container>
      <p class="rating-label">Rating</p>
      <my-rating [rating]="book.rating" [book]="book">
      </my-rating>
      <md-slide-toggle color="primary" [(ngModel)]="book.isCheckedOut"
name="checkedOut">Checked out?</md-slide-toggle>
    </md-card-content>
    <md-card-actions>
      <div class="text-right">
        <button type="submit " md-button (click)="save()" color="warn"
[disabled]="newBookForm.form.invalid"><i class="material-
icons">save</i>SAVE</button>
        <button type="reset" md-button (click)="cancel()"><i class="material-
icons">cancel</i>CANCEL</button>
      </div>
    </md-card-actions>
  </md-card>
</form>
```



```
</md-card>
```

```
</form>
```

Modify app.module.ts to include the NewBookComponent as an entryComponent as shown below. You can read more about entryComponent [here](#).

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { MaterialModule } from '@angular/material';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { AboutComponent } from './about/about.component';
import { CollectionComponent } from './collection/collection.component';
import { RatingCategoryPipe } from './rating-category.pipe';
import { RatingComponent } from './rating/rating.component';
import { HttpClientModule } from '@angular/http';
import { routing } from 'app/app.routing';
import { TabsComponent } from './tabs/tabs.component';
import { RouterModule } from '@angular/router';
import { BookDetailComponent } from './book-detail/book-detail.component';
import { BookGuardService } from 'app/book-guard.service';
import { NewBookComponent } from './new-book/new-book.component';

@NgModule({
  declarations: [
    AppComponent,
    AboutComponent,
    CollectionComponent,
    RatingCategoryPipe,
    RatingComponent,
    TabsComponent,
    BookDetailComponent,
    NewBookComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    MaterialModule,
    BrowserAnimationsModule,
```

```

    HttpModule,
    RouterModule,
    routing
  ],
  entryComponents: [
    NewBookComponent
  ],
  providers: [BookGuardService],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Modify collection.component.html to include a button to create a new book:

```

<h3>{{pageTitle}}&nbsp;<md-slide-toggle class="plr-15" color="primary"
[(ngModel)]="showOperatingHours">{{showOperatingHours ? 'Hide' : 'Show'}} library
hours</md-slide-toggle></h3>
<div [hidden]="!showOperatingHours">
  <md-card>
    <md-card-subtitle><strong>Operating Hours</strong></md-card-subtitle>
    <md-card-content>{{startTime | date:'shortTime'}} - {{endTime |
date:'shortTime'}} (M-F)</md-card-content>
  </md-card>
</div>
<div>
  <md-list>
    <md-list-item *ngFor="let book of books">
      <md-icon md-list-icon>book</md-icon>
      <h3 md-line><strong>{{book.title}}</strong></h3>
      <div>
        <button md-button (click)="openDialog(book.id)"><i class="material-
icons">pageview</i> Dialog</button>
        <button md-button (click)="openRoute(book.id)"><i class="material-
icons">pageview</i> Route</button>
      </div>
      <p md-line>
        <span>{{book.author}}</span>
      </p>
      <p md-line>
        <my-rating [rating]="book.rating" [book]="book"
(ratingClicked)="onRatingUpdate($event)">
      </my-rating>

```

```

    </p>
    <p md-line>
      <span [class]="book.isCheckedOut ? 'chip chip-danger' : 'chip chip-
success'">{{book.isCheckedOut ? 'Checked-Out' : 'Available'}}</span>
    </p>
  </md-list-item>
</md-list>
<div class="text-right add-btn">
  <button md-raised-button color="primary" (click)="addBook()"><i
class="material-icons">add_box</i> ADD BOOK</button>
</div>
</div>

```

Modify the data.service.ts file to include the addBook() service. **Save.**

```

import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import { IBook } from "app/ibook";
import 'rxjs/add/operator/do';

@Injectable()
export class DataService {

  private _booksUrl = 'http://waelbookservice.azurewebsites.net/api/books'

  constructor(private _http: Http) { }

  getBooks(): Observable<IBook[]> {
    let localBooks = localStorage.getItem('books');
    if (localBooks) {
      return Observable.create(observer => {
        observer.next(JSON.parse(localBooks));
      });
    }
    return this._http.get(this._booksUrl)
      .map((response: Response) => {
        let data: IBook[] = <IBook[]>response.json();
        localStorage.setItem('books', JSON.stringify(data));
        return data;
      })
      .catch(this.handleError);
  }
}

```

```

getBook(id: number): Observable<IBook> {
  return this.getBooks()
    .map((books: IBook[]) => books.find(b => b.id === id))
    // .do(data => console.log( JSON.stringify(data)))
    .catch(this.handleError);
}

addBook(book: IBook): Observable<IBook[]> {
  const local:string = localStorage.getItem('books');
  if (!local) return Observable.throw('Local storage not found.');
```

```

  let localBooks:IBook[] = JSON.parse(local);
  localBooks.push(book);
  localStorage.setItem('books', JSON.stringify(localBooks));
  return Observable.create(observer => {
    observer.next(localBooks);
  });
}

updateBook(book: IBook): Observable<IBook[]> {
  const local:string = localStorage.getItem('books');
  if (!local) return Observable.throw('Local storage not found.');
```

```

  let localBooks:IBook[] = JSON.parse(local);
  localBooks = localBooks.map(b => {
    if (b.id === book.id) {
      return Object.assign(b, book);
    }
    return b;
  });
  localStorage.setItem('books', JSON.stringify(localBooks));
  return Observable.create(observer => {
    observer.next(localBooks);
  });
}

private handleError(error:any) {
  let errMsg = (error.message) ? error.message : error.status ?
`${error.status} - ${error.statusText}` : 'Server error';
  console.error(errMsg);
  return Observable.throw(errMsg);
}
}

```

Modify the collection.component.ts to include the addBook() method:

```
import { Component, OnInit } from '@angular/core';
import { IBook } from 'app/ibook';
import { DataService } from '../services/data.service';
import { MdSnackBar, MdDialog } from '@angular/material';
import { BookDetailComponent } from 'app/book-detail/book-detail.component';
import { Router } from '@angular/router';
import { NewBookComponent } from '../new-book/new-book.component';

@Component({
  templateUrl: './collection.component.html',
  styleUrls: ['./collection.component.css']
})
export class CollectionComponent implements OnInit {

  constructor(private _dataService: DataService, private _snackBar: MdSnackBar,
private _dialog: MdDialog, private _router: Router) {

    this.startTime = new Date();
    this.startTime.setHours(10, 0);
    this.endTime = new Date();
    this.endTime.setHours(15, 0);
  }

  ngOnInit(): void {
    this.getBooks();
  }

  pageTitle:string = 'Books';

  books:Array<IBook>;

  startTime:Date;

  endTime:Date;

  showOperatingHours:boolean = false;

  updateMessage(message:string, type:string): void {
    if (message) {
      this._snackBar.open(`${type}: ${message}`, 'DISMISS', {
        duration: 3000
      });
    }
  }

  onRatingUpdate(book:IBook): void {
```

```

        this.updateBook(book);
    }

    openDialog(bookId:number): void {
        let config = {width: '650px', height: '400px', position: {top: '50px'}};
        let dialogRef = this._dialog.open(BookDetailComponent, config);
        dialogRef.componentInstance.bookId = bookId;
        dialogRef.afterClosed().subscribe(res => {
            this.getBooks();
        });
    }

    openRoute(bookId:number): void {
        this._router.navigate(['/collection', bookId]);
    }

    getBooks(): void {
        this._dataService.getBooks()
            .subscribe(
                books => this.books = books,
                error => this.updateMessage(<any>error, 'ERROR'));
    }

    addBook(): void {
        let config = {width: '650px', height: '650px', position: {top: '50px'},
            disableClose: true};
        let dialogRef = this._dialog.open(NewBookComponent, config);
        dialogRef.afterClosed().subscribe(newBook => {
            if (newBook) {
                newBook.id = this.books.length + 1;
                this._dataService.addBook(newBook)
                    .subscribe(
                        books => this.books = books,
                        error => this.updateMessage(<any>error, 'ERROR'));
            }
        });
    }

    updateBook(book: IBook): void {
        this._dataService.updateBook(book)
            .subscribe(
                books => {
                    this.books = books;
                    this._snackBar.open(`"${book.title}" has been updated!`, 'DISMISS', {
                        duration: 3000

```

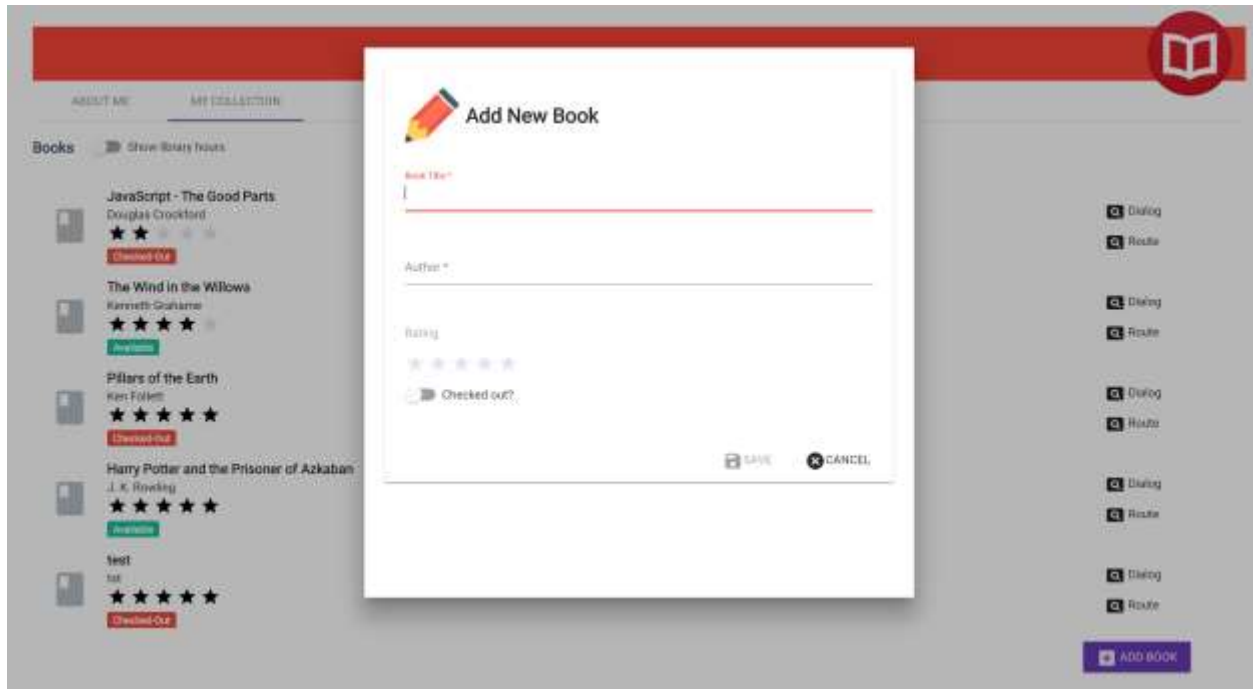
```

    });
    },error => this.updateMessage(<any>error, 'ERROR'));
  }
}

```

You should now have a button at the bottom of the collection page that allows you to add a new book. In addition, you should now have a screen that allows you to add a new book along with the necessary validation.





Congratulations you should now have a fully functional application!!!