

## **Cours Angular**



# TD MyBooks

Service - Modal

### 1] Lister les ouvrages

On désire maintenant effectuer l'opération inverse, c'est à dire récupérer de Firebase la liste des ouvrages. Il faut pour cela compléter notre service.

#### Création du template

T1.1 Changez le contenu du template books.component.html relatif à la liste des livres (ne pas confondre avec celui de l'ajout):

```
<thead>
 >
  #
  Titre
  Auteur
  Editeur
  Genre
  </thead>
{{ i+1 }}
  {{ book.titre }}
  {{ book.auteur }}
  {{ book.editeur }}
  {{ book.genre }}
  <i class="bi bi-pencil-square" style="color: green;"></i>
  <i class="bi bi-trash" style="color: red;"></i>
```

Notez la directive \*ngIf pour aucun affichage en cas de liste vide...

#### Construction du contrôleur

Le principe sera, outre les importations nécessaires, d'appeler la méthode du service retournant la liste des livres (getBooks) au chargement de la page, donc dans ngOnInit().

T1.2 Mettez le code suivant pour le template du home :

```
import { Component, OnInit } from '@angular/core';
import { IBook } from 'src/app/models/book.model';
import { BookService } from 'src/app/services/book.service';
```

Gilles Chamillard Page 1 sur 5

```
@Component({
    selector: 'app-books',
    templateUrl: './books.component.html',
    styleUrls: ['./books.component.scss']
})
export class BooksComponent implements OnInit {
    books: IBook[] = [];

    constructor(private bookService: BookService,) { }

    ngOnInit(): void {
        this.bookService.getBooks().subscribe((res: IBook[]) => {
            this.books = res;
        })
    }
}
```

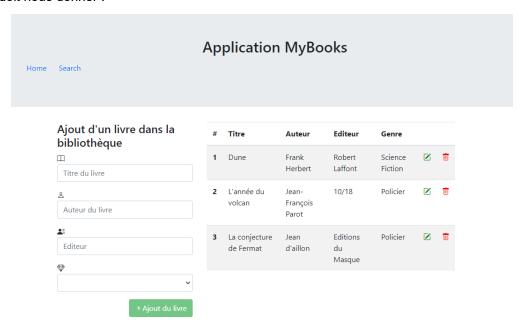
T1.3 La méthode getBooks () doit bien sûr être ajoutée dans le service :

```
getBooks(): Observable<IBook[]> {
  const livresRef = collection(this.firestore, 'livres');
  return collectionData(livresRef, { idField: 'id' }) as Observable<IBook[]>;
}
```

Le type Observable s'utilise pour transmettre des valeurs dans le cadre (entre autres) de notifications automatiques entre les éléments de l'application. Pour plus d'information, allez ici :

### https://angular.io/guide/observables

Tout ceci doit nous donner:



### 2] Autres opérations CRUD

#### Suppression d'un livre

Cette opération entraîne classiquement une modification dans le template pour la prise en compte de la corbeille au niveau du livre, de l'ajout de la méthode correspondante pour l'appel au service et enfin de la création de la méthode dans le service pour la suppression définitive dans Firebase.

Gilles Chamillard Page 2 sur 5

T2.1 Modification du template par la prise en compte d'un clic souris :

```
<i class="bi bi-trash" (click)="deleteBook (book)" style="color: red;"></i>
```

T2.2 Ajout de la méthode dans le contrôleur :

```
deleteBook(book: IBook) {
  if (confirm('Etes-vous de supprimer ce livre ?') == true) {
    this.bookService.deleteBook(book).then(() =>
        console.log('SUppression effectuée'));
  }
}
```

On se permet l'appel à une fonction native JavaScript pour la confirmation...

T2.3 Ajout de la méthode dans le service :

```
deleteBook(book: IBook) {
  const livreDocRef = doc(this.firestore, `livres/${book.id}`);
  return deleteDoc(livreDocRef);
}
```

La suppression devrait être opérationnelle!

Modification d'un livre

Cette opération est un peu plus complexe car elle fait appel à une fenêtre modale pour l'édition du livre à modifier.

T2.4 Modification du template par la prise en compte d'un clic souris :

```
<i class="bi bi-pencil-square" (click)="editModal(book)" style="color: green;"></i>
```

T2.5 Ajout de la méthode dans le contrôleur pour l'appel du composant edit-book:

```
editModal(book: IBook) {
  const modalRef = this.modal.open(EditBookComponent, {
    size: 'lg',
    centered: true,
    windowClass: 'dark-modal',
  });

  modalRef.componentInstance.id = book.id;
}
```

Vous devez insérez au début l'appel à deux importations nécessaires :

```
import { NgbModal } from '@ng-bootstrap/ng-bootstrap';
import { EditBookComponent } from '../../modal/edit-book/edit-book.component';
```

T2.6 Ajout de la méthode dans le service :

```
updateBook(book: IBook) {
  const livreDocRef = doc(this.firestore, `livres/${book.id}`);
  return setDoc(livreDocRef, book);
}
```

La méthode setDoc() remplace en détruisant l'ancien enregistrement et le créé si éventuellement il n'existait pas. On peut modifier un champ par la méthode updateDoc() mais cela ne sera pas vu ici.

Gilles Chamillard Page 3 sur 5

Le reste se passe au niveau du composant edit-book.

### T2.7 Modification du template du composant :

```
<div *ngIf="book" class="container">
   <h4>Modification d'un livre</h4>
   <form #bookForm="ngForm" (ngSubmit)="onUpdate()">
      <div class="form-group">
        <label>Titre de l'ouvrage</label>
        <input class="form-control" name="name" #nameCtrl="nqModel"</pre>
          [(ngModel)]="book.titre" placeholder="" required>
      </div>
     <div class="form-group">
        <label>Auteur</label>
        <input class="form-control" name="auteur" #authorCtrl="ngModel"</pre>
          [(ngModel)]="book.auteur" placeholder="" required>
      </div>
     <div class="form-group">
       <label>Editeur</label>
       <input class="form-control" name="editeur" #authorCtrl="ngModel"</pre>
          [(ngModel)]="book.auteur" placeholder="" required>
     <div class="form-group">
       <label>Genre</label>
       <select class="form-control" name="genre" #genreCtrl=ngModel</pre>
       [(ngModel)]="book.genre">
        <option value="Aventures">Aventures
        <option value="Policier">Policier</option>
       <option value="Science Fiction">Science Fiction</option>
       </select>
      </div>
     <div class="form-group float-right">
       <button type="submit" class="btn btn-success" [disabled]="!bookForm.valid">
         <i class="bi bi-plus"></i>Effectuer la mise à jour</button>
       </div>
   </form>
 </div>
```

Ce template présente bien sûr des similitudes avec la création d'un livre.

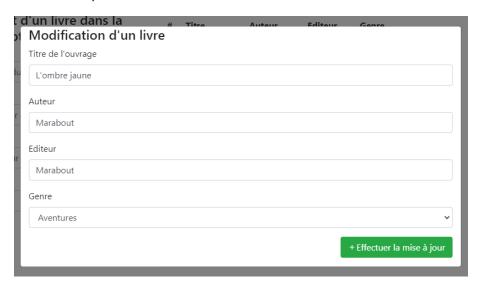
### **T2.8** Modification du contrôleur du composant edit-book:

```
import { Component, Input, OnInit } from '@angular/core';
import { NgbActiveModal } from '@ng-bootstrap/ng-bootstrap';
import { IBook } from 'src/app/models/book.model';
import { BookService } from 'src/app/services/book.service';
@Component({
 selector: 'app-edit-book',
  templateUrl: './edit-book.component.html',
 styleUrls: ['./edit-book.component.scss']
export class EditBookComponent implements OnInit {
 @Input() id!: string;
 book!: IBook;
  constructor(
   private bookService: BookService,
   public activeModal: NgbActiveModal)
     { }
  ngOnInit() {
    if (this.id)
```

Gilles Chamillard Page 4 sur 5

```
this.bookService.getBookByID(this.id).subscribe(res => {
        this.book = res
    });
}
onUpdate() {
   this.bookService.updateBook(this.book).then(() => {
        this.activeModal.close();
        console.log('Données ajoutées !');
   })
}
```

La modification devrait être opérationnelle :



## 3] Exercice

Dans cette application, il manque le traitement de la recherche d'un livre. Ceci peut faire l'objet d'un exercice !

Pour vous aider, vous avez des tutoriels sur Internet dont en voici un exemple :

https://ozenero.com/angular-12-firebase-autocomplete-search-example-with-angularfire2-v4

Gilles Chamillard Page 5 sur 5