

# Angular Création de ma bibliothèque

Amina MARIE







# Exercice - Bibliothèque




# Plan de l'exercice

- Installer Angular CLI et créer un nouveau projet Angular
- Configurer le projet
- Créer un modèle de données (Book)
- Créer un service pour gérer les livres
- Créer des composants pour afficher et gérer les livres
- Utiliser des formulaires pour ajouter des livres
- Mettre en place l'application et vérifier le fonctionnement



# Objectif



Créer un petit programme TypeScript pour gérer une bibliothèque de livres.

Vous allez définir des classes et des interfaces pour modéliser les livres et les opérations de gestion de bibliothèque.



# Créer un nouveau projet Angular



---

**Créer un nouveau projet Angular**  
ng new library-management-angular

**Naviguer dans le repertoire**  
cd library-management-angular

# Configurer le projet



Démarrer le serveur de developpement

*ng server*



# Créer un modèle de données



## Créer un fichier `book.model.ts` dans le dossier `src`

Le fichier `book.model.ts` sert à définir le modèle de données pour représenter un livre dans votre application.

En TypeScript, cela se fait généralement en utilisant une interface ou une classe.

Vous allez devoir créer une interface en utilisant « `export interface Book` » et définir les attributs de votre livre (pensez à y intégrer un identifiant)

## Utilisation de `book.model.ts` dans votre Application

Le modèle `Book` sera utilisé dans d'autres parties de votre application pour s'assurer que les objets livres sont toujours conformes à cette structure.

# Créer un modèle de données



Objectifs du fichier book.model.ts

## 1. Définir la Structure des Données :

- Il spécifie la forme et les types des données d'un livre. Cela permet de s'assurer que chaque livre a les mêmes propriétés avec les types appropriés.

## 2. Faciliter le Typage Statique :

- TypeScript est un langage à typage statique, ce qui signifie qu'il vérifie les types au moment de la compilation. En définissant un modèle, TypeScript peut aider à attraper les erreurs de type avant que le code ne soit exécuté.

## 3. Documentation et Lisibilité



# Créer un service pour gérer les livres



## Utiliser Angular CLI pour générer le service book

ng generate service book

Modifier le fichier src/app/book.service.ts pour:

- Importer l'interface Book
- Implémenter la classe Book qui :
  - Contiendra un tableau pour stocker les livres.
  - Attribuera des identifiants uniques à vos livres
  - Contiendra une méthode addbook pour ajouter des livres, removebook pour supprimer des livres et getbooks pour avoir les listes des livres

@Injectable({ providedIn: 'root' }) : Cette annotation indique que ce service est disponible pour toute l'application (singleton).

# Créer des composants pour afficher et gérer les livres



## Générer un composant pour la liste des livres

Un composant en Angular est une classe avec des métadonnées (annotations) qui contrôlent une vue (template HTML).

Utilisez Angular CLI pour générer un composant.

*ng generate component book-list*

Cette commande crée quatre fichiers :

- book-list.component.ts,
- book-list.component.html,
- book-list.component.css,
- book-list.component.spec.ts.

# Créer des composants pour afficher et gérer les livres



## Implémenter le composant de la liste des livres

Modifiez `src/app/book-list/book-list.component.ts` pour utiliser le service et afficher les livres

```
import { Component, OnInit } from '@angular/core';
import { BookService } from '../book.service';
import { Book } from '../../book.model';

@Component({
  selector: 'app-book-list',
  templateUrl: './book-list.component.html',
  styleUrls: ['./book-list.component.css']
})
export class BookListComponent implements OnInit {
  books: Book[] = [];

  constructor(private bookService: BookService) {}

  ngOnInit(): void {
    this.books = this.bookService.getBooks();
  }

  deleteBook(id: number): void {
    this.bookService.deleteBook(id);
    this.books = this.bookService.getBooks();
  }
}
```



# Créer des composants pour afficher et gérer les livres



BookListComponent : La classe du composant.

BookService : Injecté dans le composant pour accéder aux livres.

ngOnInit : Méthode appelée une fois que le composant est initialisé.

deleteBook : Méthode pour supprimer un livre en utilisant le service.

# Créer des composants pour afficher et gérer les livres



## Ajouter le template du composant de la liste des livres

Modifiez src/app/book-list/book-list.component.html pour définir la structure HTML

```
<h2>Liste des livres</h2>
<ul>
  <li *ngFor="let book of books">
    {{ book.title }} ({{ book.author }} - {{ book.publishedYear }})
    <button (click)="deleteBook(book.id)">Supprimer</button>
  </li>
</ul>
```

\*ngFor="let book of books" : Directive Angular pour boucler sur la liste des livres.

(click)="deleteBook(book.id)" : Binding d'événement pour appeler deleteBook lors du clic sur le bouton.

# Utiliser des formulaires pour ajouter des livres



## Générer un composant pour ajouter des livres

Utilisez Angular CLI pour générer un autre composant.

*ng generate component book-form*

Modifiez `src/app/book-form/book-form.component.ts` pour gérer le formulaire d'ajout de livres :

```
import { Component } from '@angular/core';
import { BookService } from '../book.service';

@Component({
  selector: 'app-book-form',
  templateUrl: './book-form.component.html',
  styleUrls: ['./book-form.component.css']
})
export class BookFormComponent {
  title = '';
  author = '';
  publishedYear: number | null = null;

  constructor(private bookService: BookService) {}

  addBook(): void {
    if (this.title && this.author && this.publishedYear) {
      this.bookService.addBook(this.title, this.author, this.publishedYear);
      this.title = '';
      this.author = '';
      this.publishedYear = null;
    }
  }
}
```



# Utiliser des formulaires pour ajouter des livres


title, author, publishedYear : Propriétés pour lier les champs de formulaire.

addBook : Méthode pour ajouter un livre via le service.

## Ajouter le template du composant de formulaire pour ajouter des livres

Modifiez src/app/book-form/book-form.component.html pour définir le formulaire.

```
<h2>Ajouter un livre</h2>
<form (ngSubmit)="addBook()">
  <label for="title">Titre:</label>
  <input type="text" id="title" [(ngModel)]="title" name="title" required>
  <br>
  <label for="author">Auteur:</label>
  <input type="text" id="author" [(ngModel)]="author" name="author" required>
  <br>
  <label for="publishedYear">Année de publication:</label>
  <input type="number" id="publishedYear" [(ngModel)]="publishedYear" name="publishedYear" required>
  <br>
  <button type="submit">Ajouter</button>
</form>
```




# Mettre en place l'application et vérifier le fonctionnement



## Modifier le composant racine

Modifiez `src/app/app.component.html` pour inclure les nouveaux composants :

```
<h1>Gestion de bibliothèque</h1>
<app-book-form></app-book-form>
<app-book-list></app-book-list>
```



# Mettre en place l'application et vérifier le fonctionnement

## Importer les modules nécessaires

Assurez-vous que le module de formulaires d'Angular est importé dans `app.module.ts` :



Mettre en place  
l'application et  
vérifier le  
fonctionnement



```
// src/app/app.module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { RouterModule, Routes } from '@angular/router';

import { AppComponent } from './app.component';
import { BookListComponent } from './book-list/book-list.component';
import { BookFormComponent } from './book-form/book-form.component';

// Définir les routes
const routes: Routes = [
  { path: '', component: BookListComponent },
  { path: 'add-book', component: BookFormComponent }
];

@NgModule({
  declarations: [
    AppComponent,
    BookListComponent,
    BookFormComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    RouterModule.forRoot(routes) // Configurer les routes ici
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# Vérification et exécution

Démarrer le serveur de développement :

ng serve

Ouvrir le navigateur et accéder à <http://localhost:4200>.

# Encore un exercice



Vous avez compris comment procéder?

Essayez de refaire cet exercice mais cette fois-ci avec une liste de tâches!

Objectif:

- Ajouter une nouvelle tâche.
- Afficher la liste des tâches.
- Marquer une tâche comme terminée.
- Supprimer une tâche.