

Typescript

Amina MARIE





Exercice - Bibliothèque




Ce que ça vous apporte

Dans cet exercice, vous allez :

- Définir et utiliser des **interfaces** pour modéliser les données (Book).
- Créer des **classes** pour encapsuler la logique et gérer les données (LibraryService).
- Manipuler des données avec des **méthodes** dans les classes (ajouter, supprimer, et récupérer des livres).
- Utiliser les **modules TypeScript** pour organiser le code.
- Configurer et exécuter un projet TypeScript.

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.



Configurer le projet



Créer un nouveau répertoire pour le projet et y naviguer

mkdir library-management

cd library-management

Initialiser un projet Node.js (pour gérer les dépendances)

npm init -y

Cette commande crée un fichier package.json avec les paramètres par défaut.

Installer TypeScript en tant que dépendance de developpement

npm install typescript --save-dev

Configurer le projet

Créer un fichier de configuration TypeScript

npx tsc --init

Cela crée un fichier tsconfig.json avec les configurations par défaut.

Créer un dossier src pour les fichiers TypeScript

mkdir src



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 le Service de Bibliothèque

Créer un fichier `library.service.ts` dans le dossier `src`

Nous allons créer une classe `LibraryService` pour gérer les livres.

1. Dans votre classe vous devrez utiliser `Book` donc commencez par faire un import de votre interface `Book`.
2. Créer une classe `LibraryService` qui:
 1. Contiendra un tableau pour stocker les livres.
 2. Attribuera des identifiants uniques à vos livres
 3. Contiendra une méthode `addbook` pour ajouter des livres, `removebook` pour supprimer des livres et `getbooks` pour avoir les listes des livres

Créer votre programme principale



Créer un fichier main.ts dans le dossier src

Nous allons écrire le programme principal pour tester notre service de bibliothèque.

Les étapes:

1. Importer la classe LibraryService
2. Créer une instance de LibraryService
3. Ajouter des livres et vérifier la liste des livres dans la console
4. Supprimer un livre et vérifier la liste des livres dans la console

Compiler et exécuter votre code

Ajouter un script de compilation dans le package.json

Ouvrez le fichier package.json et ajoutez les scripts suivants

```
"scripts": {  
  "build": "tsc",  
  "start": "tsc && node dist/main.js"  
}
```

"build": "tsc" : Ce script exécute le compilateur TypeScript (tsc) pour convertir les fichiers TypeScript en fichiers JavaScript.

"start": "tsc && node dist/main.js" : Ce script compile le code TypeScript et exécute le fichier JavaScript résultant (main.js) dans le dossier dist.

Compiler et exécuter votre code

Compiler le code TypeScript

npm run build

Cela génère les fichiers JavaScript dans un dossier dist.

Si jamais le dossier dist ne s'est pas créé alors vérifier le fichier de configuration typeScript: tsconfig.json

Vous devez avoir à la fin du fichier

```
- "skipLibCheck":  
  true, /* Skip type  
  checking all .d.ts files. */  
- "rootDir": "./src",  
- "outDir": "./dist"  
- },  
- "include": ["src/**/*"]  
- }
```

Compiler et exécuter votre code




Exécuter le projet

npm start

Vous devriez avoir la console contenant vos log d'ajout... de livres

Résumé

- 
- Configuration du Projet : Créer et configurer un projet TypeScript avec Node.js.
 - Modèle de Données : Définir une interface Book pour représenter les livres.
 - Service de Bibliothèque : Créer une classe LibraryService pour gérer les livres (ajouter, supprimer, obtenir).
 - Programme Principal : Écrire le programme principal pour tester le service de bibliothèque.
 - Compilation et Exécution : Compiler le code TypeScript et exécuter le programme.
- 