## Tutoriel : Création d'un Projet Backend en TypeScript

***Objectif :***

L'objectif de ce tutoriel est de configurer un projet **backend** en **TypeScript**, tout en lançant un serveur **Express** qui écoute sur le port 3000 via la commande `npm start`. Le projet ne génère aucun fichier JavaScript, grâce à l'utilisation de **TypeScript** et **ts-node**.

---

### Étape 1 : Initialisation du Projet

#### *1.1. Créer un nouveau projet Node.js*

La première étape consiste à initialiser un nouveau projet Node.js. Cela va générer un fichier `package.json`, qui contiendra les métadonnées du projet ainsi que les dépendances nécessaires.

Exécuter la commande suivante dans un terminal à la racine de votre projet :

```bash
npm init -y
```

Cela génère un fichier `package.json` avec une configuration par défaut.

#### *1.2. Installer les dépendances*

Pour mettre en place un serveur avec **Express** et utiliser **TypeScript**, les dépendances suivantes doivent être installées :

- **Express** : Le framework pour gérer les routes et le serveur HTTP.
- **TypeScript** : Pour écrire le code en TypeScript.
- **ts-node** : Permet d'exécuter directement des fichiers TypeScript sans les compiler en JavaScript.
- **@types/express** et **@types/node** : Pour ajouter des types TypeScript à Express et Node.js.

Voici la commande à exécuter :

```
npm install express
npm install --save-dev typescript ts-node @types/express @types/node
```

---

### Étape 2 : Configuration TypeScript

#### *2.1. Générer un fichier `tsconfig.json`*

Le fichier `tsconfig.json` permet de configurer le compilateur **TypeScript**. Pour le créer automatiquement, exécuter la commande suivante :

```
npx tsc --init
```

Cela va générer un fichier `tsconfig.json` par défaut. Nous allons ensuite le modifier pour correspondre à nos besoins.

### *2.2. Configurer `tsconfig.json`*

Voici quelques configurations importantes à ajouter dans `tsconfig.json` :

- **rootDir** : Définit le dossier racine des fichiers TypeScript (ici, `src/`).
- **outDir** : Spécifie où mettre les fichiers compilés (ceci est optionnel, mais pour l'instant on ne va pas compiler en `.js`).
- **esModuleInterop** : Permet l'interopérabilité des modules ES avec CommonJS, nécessaire pour utiliser `import` avec des modules comme Express.

Voici un exemple de configuration minimaliste :

```
{
  "compilerOptions": {
    "target": "ES6",
    "module": "commonjs",
    "rootDir": "./src",
    "strict": true,
    "esModuleInterop": true
  },
  "include": ["src/**/*"]
}
```

Cela dit à TypeScript de prendre tous les fichiers source depuis le dossier `src/`.

---

**Étape 3 : Création de la Structure**

Maintenant que la configuration TypeScript est en place, passons à la création de la structure du projet.

### *3.1. Structure du projet*

Voici la structure de fichiers à mettre en place :

```
/project-root
  ├── src/
  │     ├── models/
  │         │        └── userModel.ts
  │         └── server.ts
  ├── node_modules/
  ├── package.json
  ├── package-lock.json
  ├── tsconfig.json
  └── .gitignore
```

- `src/` : Contient tous les fichiers source du projet en TypeScript.
  - `models/` : Contient les modèles de données, contient userModel.ts.
  - `server.ts` : Fichier principal où Express est configuré et lancé.

---

**Étape 4 : Configuration du Serveur Express**

***4.1. Création du fichier `server.ts`***

Dans le fichier `server.ts`, on va créer un serveur **Express** basique qui écoute sur le port 3000.

```
// src/server.ts
import express from 'express';

const app = express();
const PORT = 3000;

app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

Ce code crée un serveur **Express** qui ne fait rien d'autre qu'écouter sur le port 3000.

---

**Étape 5 : Lancer le Serveur en TypeScript**

***5.1. Utilisation de `ts-node` pour exécuter le serveur***

Au lieu de compiler les fichiers TypeScript en JavaScript, on va utiliser **`ts-node`** pour exécuter directement les fichiers `.ts`.

Ouvrez le fichier `package.json` et ajoutez le script suivant pour lancer le serveur en développement :

```
"scripts": {
   "start": "ts-node src/server.ts"
}
```

Cela permet de démarrer le serveur avec la commande :

```
npm start
```

***5.2. Ignorer les fichiers non nécessaires***

Pour éviter de suivre les fichiers compilés et les modules dans Git, ajoutez un fichier `.gitignore` avec les règles suivantes :

```
node_modules/
```

---

**Résultat Final**

1. Le projet est maintenant configuré pour utiliser uniquement des fichiers TypeScript sans générer de fichiers JavaScript.
2. Le serveur **Express** écoute sur le port **3000** et est démarré via `npm start`, exécuté directement en TypeScript avec **ts-node**.

---

Récapitulatif des Commandes Utilisées

- Initialisation du projet :

```
npm init -y
```

- Installation des dépendances :

```
npm install express
npm install --save-dev typescript ts-node @types/express @types/node
```

- Génération du fichier `tsconfig.json` :

```
npx tsc --init
```

- Lancer le serveur :

```
npm start
```