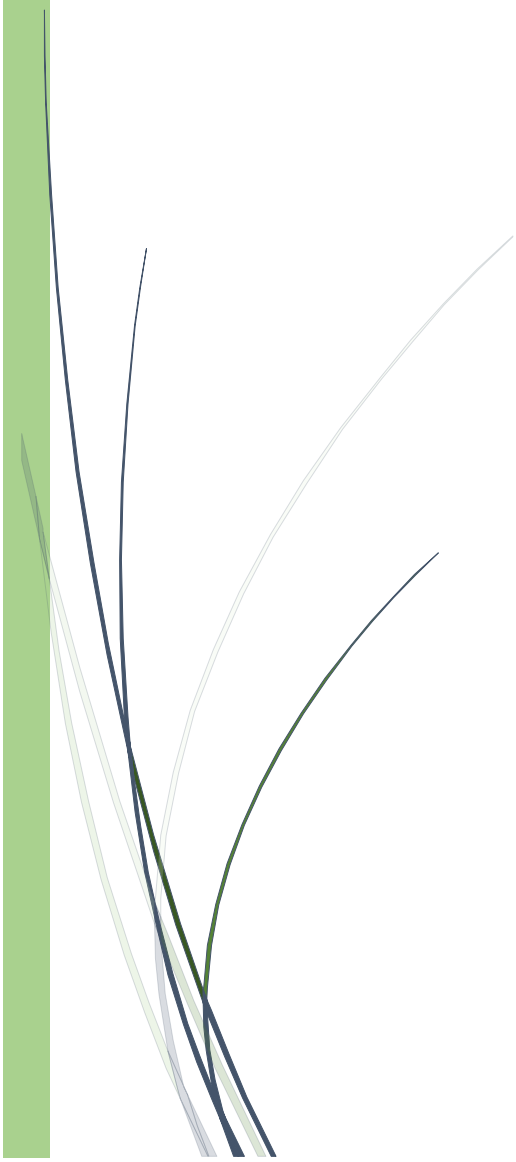
11/01/2025

Gestion des Fichiers avec Node.js et Express.js

Manipulation et Intégration des Fichiers dans des Applications Web



MAHDI KELLOUCH
ISMO TETOUAN

Table des matières

Introduction

Node.js fournit un module natif, fs (file system), pour interagir avec le système de fichiers. Avec **fs**, il est possible de lire, écrire, supprimer et manipuler les fichiers et dossiers sur un serveur.

Ce chapitre couvre les fonctionnalités clés de "**fs**" ainsi que son intégration dans des applications web en utilisant **Express.js**.

Objectifs

- Comprendre les opérations de base sur les fichiers et dossiers avec **fs**.
- Intégrer les fonctionnalités du module fs dans une application **Express.js**.
- Appliquer les connaissances acquises à travers des exercices pratiques.

I. Présentation du module fs

1.1 Importation du module fs

Pour utiliser le module **fs**, il doit être importé au début de votre fichier Node.js :

```
const fs = require("fs");
// ou bien
import fs from 'fs'
```

1.2 Fonctionnalités principales

fs propose deux modes de fonctionnement :

- **Asynchrone** : Les opérations sont non bloquantes et utilisent des callbacks ou des promesses.
- **Synchrone** : Les opérations bloquent le thread principal.

Exemple :

- **Asynchrone** :

```
fs.readFile('example.txt', 'utf8', (err, data) => {
    if (err) {
        console.error(err);
        return;
    }
    console.log(data);
});
```

- **Synchrone** :

```
try {
    const data = fs.readFileSync('example.txt', 'utf8');
    console.log(data);
} catch (err) {
    console.error(err);
}
```

II. Manipulation des fichiers

2.1. Lecture de fichiers

fs.readFile : Lire le contenu d'un fichier.

```
fs.readFile('example.txt', 'utf8', (err, data) => {
    if (err) throw err;
    console.log(data);
});
```

2.2 Écriture dans des fichiers

fs.writeFile : Créer un fichier ou remplacer son contenu.

```
fs.writeFile('example.txt', 'Bonjour le monde!', (err) => {
    if (err) throw err;
    console.log('Fichier créé avec succès !');
});
```

2.3 Ajout de contenu

fs.appendFile : Ajouter du contenu à la fin d'un fichier.

```
fs.appendFile('example.txt', '\nAjout de contenu.', (err) => {
    if (err) throw err;
    console.log('Contenu ajouté !');
});
```

1.4 Suppression de fichiers

fs.unlink : Supprimer un fichier.

```
fs.unlink('example.txt', (err) => {
    if (err) throw err;
```

```
    console.log('Fichier supprimé avec succès !');
});
```

III. Manipulation des dossiers

3.1 Création de dossiers

fs.mkdir : Créer un dossier.

```
fs.mkdir('monDossier', (err) => {
    if (err) throw err;
    console.log('Dossier créé !');
});
```

3.2 Lecture du contenu d'un dossier

fs.readdir : Lister les fichiers et sous-dossiers.

```
fs.readdir('./', (err, files) => {
    if (err) throw err;
    console.log('Contenu du dossier :', files);
});
```

3.3 Suppression de dossiers

fs.rmdir : Supprimer un dossier vide.

```
fs.rmdir('monDossier', (err) => {
    if (err) throw err;
    console.log('Dossier supprimé !');
});
```

IV. Intégration avec Express.js

4.1 Exemple : Upload et gestion de fichiers

Créons une application Express.js permettant de téléverser et de gérer des fichiers.

a. Installation des dépendances

```
npm install express multer
```

b. Code:

```
const express = require('express');
const multer = require('multer');
const fs = require('fs');
const app = express();
const upload = multer({ dest: 'uploads/' });

app.use(express.json());

// Route pour téléverser un fichier
app.post('/upload', upload.single('file'), (req, res) => {
    const tempPath = req.file.path;
    const targetPath = `uploads/${req.file.originalname}`;

    fs.rename(tempPath, targetPath, (err) => {
        if (err) return res.status(500).send(err);
        res.send('Fichier téléversé avec succès !');
    });
});

// Route pour lister les fichiers téléversés
app.get('/files', (req, res) => {
    fs.readdir('uploads/', (err, files) => {
        if (err) return res.status(500).send(err);
        res.json(files);
    });
});

app.listen(3000, () => {
    console.log('Serveur démarré sur le port 3000');
});
```

V. Exercices pratiques

5.1. Exercice 1 : Gestion des fichiers

Créez un script Node.js qui :

a. Crée un fichier "test.txt".
b. Ajoute du texte à ce fichier.
c. Lit et affiche le contenu du fichier.
d. Supprime le fichier.

5.2 Exercice 2 : Application Express.js

Ajoutez une fonctionnalité à l'application Express.js pour :

a. Télécharger plusieurs fichiers.
b. Supprimer un fichier spécifique via une requête HTTP DELETE.

VI. Conclusion

Le module **fs** de **Node.js** offre des outils puissants pour interagir avec les fichiers et dossiers. En combinaison avec **Express.js**, vous pouvez créer des applications robustes permettant de gérer facilement les fichiers.