I/O *suite*: Lecture et écriture de fichiers en C#

- La classe File
- File vs FileStream vs StreamReader
- Les méthodes de la classe File
- Conclusion

La classe `File`

- La classe `File` en C# est utilisée pour la **lecture** et l'**écriture** de fichiers.
- Elle est construite par dessus la classe FileStream pour **faciliter** les opérations sur des fichiers sans avoir besoin de comprendre les détails internes du fichier, comme les octets ou les flux.

Pourquoi utiliser uniquement `File` pour un débutant ?

- `FileStream` est la classe de base pour manipuler des fichiers, mais elle est **plus complexe** car elle opère au niveau des **octets** du fichier. Cela implique de travailler avec des données brutes, ce qui n'est pas nécessaire pour les besoins courants.
- `StreamReader`, tout comme `File`, est un classe construite par-dessus `FileStream`. Elle est utilisé pour lire des fichiers ligne par ligne sans charger tout le fichier en mémoire. C'est utile lorsque le fichier est très volumineux, mais cela est rarement nécessaire pour de petits projets.

Méthodes de la classe `File`

`File.WriteAllText`

- **Description** : Écrit du texte dans un fichier. Si le fichier n'existe pas, il sera créé ; s'il existe déjà, son contenu sera écrasé.
- **Usage** : Pratique pour écrire rapidement du texte dans un fichier.
- **Exemple d'utilisation** : Sauvegarder un message dans un fichier texte.

`File.ReadAllText`

- **Description** : Lit tout le contenu d'un fichier en une seule fois et retourne une chaîne de caractères.
- **Usage** : Idéal pour lire des petits fichiers en texte intégral.
- **Exemple d'utilisation** : Charger un fichier de configuration ou un fichier de log.

`File.AppendAllText`

- **Description** : Ajoute du texte à la fin d'un fichier. Si le fichier n'existe pas, il sera créé.
- **Usage** : Utile pour ajouter des données sans écraser le contenu existant.
- **Exemple d'utilisation** : Ajouter une nouvelle entrée à un journal ou log.

`File.Copy`

- **Description** : Copie un fichier à un autre emplacement.
- **Usage** : Pratique pour créer des copies de sauvegarde d'un fichier existant.
- **Exemple d'utilisation** : Copier un fichier avant de faire des modifications.

`File.Delete`

- **Description** : Supprime un fichier du système de fichiers.
- **Usage** : Permet de nettoyer des fichiers inutiles.
- **Exemple d'utilisation** : Supprimer des fichiers temporaires après utilisation.

`File.Exists`

- **Description** : Vérifie si un fichier existe à l'emplacement spécifié.
- **Usage** : Permet de vérifier l'existence d'un fichier avant de tenter de le lire ou de le modifier.
- **Exemple d'utilisation** : Vérifier si un fichier de configuration est présent avant de le charger.

`File.ReadAllLines`

- **Description** : Lit toutes les lignes d'un fichier et retourne un tableau de chaînes de caractères, où chaque élément correspond à une ligne du fichier.
- **Usage** : Utile lorsque vous devez manipuler un fichier ligne par ligne dans un petit projet.
- **Exemple d'utilisation** : Lire une liste de noms ou d'entrées séparées par des lignes.

`File.WriteAllLines`

- **Description** : Écrit un tableau de chaînes de caractères dans un fichier, chaque chaîne devenant une ligne distincte dans le fichier.
- **Usage** : Permet d'écrire facilement du contenu structuré dans un fichier.
- **Exemple d'utilisation** : Enregistrer une liste de données dans un fichier texte.

`File.Move`

- **Description** : Déplace un fichier d'un emplacement à un autre.
- **Usage** : Permet de réorganiser des fichiers ou de les archiver.
- **Exemple d'utilisation** : Déplacer des fichiers de logs dans un répertoire d'archives.

`File.Open`

- **Description** : Ouvre un fichier et retourne un `FileStream`. Cependant, pour un débutant, l'utilisation de `File.Open` est rarement nécessaire, car les méthodes simplifiées de la classe `File` suffisent dans la majorité des cas.
- **Usage** : Utilisé uniquement si un contrôle plus fin sur la lecture ou l'écriture est nécessaire.

Conclusion

- La classe `File` offre des méthodes simples et puissantes pour gérer les fichiers sans avoir à plonger dans les détails complexes de `FileStream` ou `StreamReader`.
- Il est recommandé de **se concentrer sur la classe `File`** car elle répond à la plupart des besoins courants en matière de gestion de fichiers.
- Les classes comme `FileStream` et `StreamReader` sont utilisées dans des situations spécifiques et avancées, et ne sont généralement pas nécessaires pour des tâches simples.