Gestion des fichiers en PHP

Manipuler des fichiers en PHP est courant pour lire, écrire et traiter des données.

Les fonctions `file` , `file_get_contents` et `file_put_contents` facilitent ces opérations..

1 - Les fonctions principales

1.1 - file_put_contents

Cette fonction permet d'écrire des données dans un fichier. Si le fichier n'existe pas, il est créé automatiquement. Par défaut, elle écrase le contenu existant.

Exemple :

Écrire dans un fichier :

```php
$contenu = "Alice*alice@email.com\nBob*bob@email.com";
file_put_contents('utilisateurs.txt', $contenu);
```

Ajouter du contenu sans écraser ce qui existe :

```php
$nouveauContenu = "Charlie*charlie@email.com\n";
file_put_contents('utilisateurs.txt', $nouveauContenu, FILE_APPEND);
```

1.1.1 - Sécuriser l'accès concurrent avec le drapeau `LOCK_EX`

Lorsque plusieurs utilisateurs ou processus peuvent écrire sur le même fichier en même temps (par exemple sur un serveur web), il existe un risque que les écritures se mélangent ou que des données soient perdues.

Le drapeau `LOCK_EX` permet de verrouiller le fichier pendant la durée de l'écriture, empêchant ainsi d'autres écritures d'intervenir simultanément.

Exemple d'utilisation de `LOCK_EX` :

```php
file_put_contents('utilisateurs.txt', "Nouvelle entrée sécurisée.\n", FILE_APPEND | LOCK_EX);
```

Dans cet exemple, on ajoute une ligne sans effacer le contenu existant (`FILE_APPEND`) et on protège l'écriture contre d'autres écritures concurrentes (`LOCK_EX`).
Ainsi, même si plusieurs scripts essaient de modifier le fichier en même temps, chaque opération d'écriture sera effectuée l'une après l'autre, sans chevauchement ni corruption du fichier.

Bonnes pratiques : Il est toujours recommandé d'utiliser `LOCK_EX` lors d'ajout ou de modification de fichiers partagés sur des environnements multi-utilisateurs.

1.2 - file_get_contents

Utilisée pour lire tout le contenu d'un fichier dans une chaîne.

Exemple :

Lire un fichier et afficher son contenu :

```php
$contenu = file_get_contents('utilisateurs.txt');
echo $contenu;
```

Vérifiez toujours l'existence du fichier avant de le lire :

```php
if (file_exists('utilisateurs.txt')) {
    $contenu = file_get_contents('utilisateurs.txt');
} else {
    echo "Le fichier n'existe pas.";
}
```

1.3 - file

La fonction `file` lit un fichier et retourne chaque ligne sous forme d'élément dans un tableau.

Exemple :

Lire un fichier ligne par ligne :

```php
$lignes = file('utilisateurs.txt');
foreach ($lignes as $ligne) {
    echo $ligne; // Affiche chaque ligne du fichier
}
```

Exemple pratique : Lire et traiter des données avec un séparateur

2.1 - Cas d'utilisation

Un fichier stocke des informations sous cette forme :

```
Alice*alice@email.com
Bob*bob@email.com
```

2.2 - Objectif

Nous souhaitons lire chaque ligne et extraire les informations (`nom` et `email`) séparées par l'astérisque `*` .

Code :

```php
$lignes = file('utilisateurs.txt'); // Lecture ligne par ligne

foreach ($lignes as $ligne) {
    $ligne = trim($ligne); // Supprime les éventuelles espaces ou retours à la ligne
    list($nom, $email) = explode('*', $ligne); // Sépare les données
    echo "Nom : $nom, Email : $email" . PHP_EOL;
}
```

2.3 - Explication

1. `file()` : Charge chaque ligne du fichier dans un tableau.
2. `trim()` : Supprime les espaces ou sauts de ligne résiduels.
3. `explode('*', $ligne)` : Divise les données en utilisant `*` comme séparateur.
4. `list($nom, $email)` : Associe chaque partie des données à une variable spécifique.

2.4 - Résultat attendu

Pour le contenu suivant :

```
Alice*alice@email.com
Bob*bob@email.com
```

Le script affichera :

```
Nom : Alice, Email : alice@email.com
Nom : Bob, Email : bob@email.com
```

À RETENIR

Fonction	Usage	Exemple
`file_put_contents`	Écrit ou ajoute des données dans un fichier	`file_put_contents('file.txt', 'Texte');`
`file_get_contents`	Lit tout le contenu d'un fichier	`$data = file_get_contents('file.txt');`
`file`	Lit fichier ligne par ligne (tableau)	`$lignes = file('file.txt');`

3.1 - Bonnes pratiques

1. Vérifiez si le fichier existe avant de lire son contenu :

```php
if (file_exists('fichier.txt')) { ... }
```

2. Utilisez `FILE_APPEND` pour ajouter sans écraser :

```php
file_put_contents('file.txt', "Ajout\n", FILE_APPEND);
```