Manipulation des fichiers en PHP

Par Robert DIASSÉ

1. Introduction aux fichiers en PHP

PHP permet de manipuler des fichiers stockés sur le serveur pour lire, écrire, modifier et supprimer des données. La gestion des fichiers est essentielle pour stocker des informations persistantes sans utiliser une base de données.

Les principales opérations sur les fichiers en PHP sont :

- **Création et écriture de fichiers**
- **Lecture des fichiers**
- **Manipulation de fichiers JSON et CSV**
- **Vérification et gestion des permissions**

Nous allons nous concentrer sur `file_put_contents()` et `file_get_contents()`, qui sont les méthodes les plus simples et efficaces pour gérer les fichiers.

2. Lecture et écriture avec `file_put_contents()` et `file_get_contents()`

2.1. Écriture avec `file_put_contents()`

La fonction `file_put_contents()` permet d'écrire facilement du texte dans un fichier.

Exemple : écrire dans un fichier texte

```php
<?php
$contenu = "Bonjour, ceci est un fichier texte en PHP.";
file_put_contents("mon_fichier.txt", $contenu);
echo "Fichier créé avec succès.";
?>
```

Explication :

- `file_put_contents("mon_fichier.txt", $contenu)` crée un fichier `mon_fichier.txt` et y écrit la chaîne `$contenu`.
- Si le fichier existe, son contenu est remplacé.

2.2. Lecture avec `file_get_contents()`

Pour récupérer le contenu d'un fichier, on utilise `file_get_contents()`.

Exemple : lire un fichier texte

```php
<?php
$contenu = file_get_contents("mon_fichier.txt");
echo "Contenu du fichier : " . $contenu;
?>
```

Explication :

- `file_get_contents("mon_fichier.txt")` lit l'intégralité du fichier et retourne son contenu sous forme de chaîne.
- On affiche ensuite ce contenu avec `echo`.

2.3. Ajouter du contenu sans écraser

Pour ajouter du contenu au lieu de remplacer le fichier, on utilise `FILE_APPEND`.

```php
<?php
file_put_contents("mon_fichier.txt", "\nLigne ajoutée.", FILE_APPEND);
echo "Texte ajouté au fichier.";
?>
```

Explication :

- `FILE_APPEND` permet d'ajouter du contenu à la fin du fichier sans écraser les données existantes.

---

3. Autres méthodes : `fopen()`, `fwrite()`, `fread()`

3.1. Création et écriture avec `fopen()` et `fwrite()`

```php
<?php
$fichier = fopen("mon_fichier2.txt", "w"); // Ouvre le fichier en mode écriture
fwrite($fichier, "Ceci est un test.");
fclose($fichier);
echo "Fichier écrit avec fopen().";
?>
```

Explication :

- `fopen("mon_fichier2.txt", "w")` ouvre un fichier en mode écriture (`w` = write). S'il existe, il est écrasé.
- `fwrite()` écrit du texte dans le fichier.
- `fclose()` ferme le fichier.

3.2. Lecture avec `fopen()` et `fread()`

```php
<?php
$fichier = fopen("mon_fichier2.txt", "r"); // Ouvre en mode lecture
$contenu = fread($fichier, filesize("mon_fichier2.txt"));
fclose($fichier);
echo "Contenu : " . $contenu;
?>
```

Explication :

- `fopen("mon_fichier2.txt", "r")` ouvre le fichier en mode lecture.
- `fread($fichier, filesize("mon_fichier2.txt"))` lit tout le contenu du fichier.
- `fclose($fichier)` ferme le fichier.

---

4. Manipulation de fichiers JSON et CSV

4.1. Manipulation de fichiers JSON

Écriture dans un fichier JSON

```php
<?php
$donnees = ["nom" => "Alice", "age" => 25];
$json = json_encode($donnees);
file_put_contents("donnees.json", $json);
echo "Fichier JSON créé.";
?>
```

Lecture d'un fichier JSON

```php
<?php
$json = file_get_contents("donnees.json");
$donnees = json_decode($json, true);
print_r($donnees);
?>
```

4.2. Manipulation de fichiers CSV

Écriture dans un fichier CSV

```php
<?php
$donnees = [
    ["Nom", "Age"],
    ["Alice", 25],
    ["Bob", 30]
];
$fichier = fopen("donnees.csv", "w");
foreach ($donnees as $ligne) {
    fputcsv($fichier, $ligne);
}
fclose($fichier);
echo "Fichier CSV créé.";
?>
```

Lecture d'un fichier CSV

```php
<?php
$fichier = fopen("donnees.csv", "r");
while (($ligne = fgetcsv($fichier)) !== false) {
    print_r($ligne);
}
fclose($fichier);
?>
```

---

5. Comparaison des méthodes

Méthode	Avantages	Inconvénients
`file_put_contents()`	Simple et rapide	Pas de contrôle avancé
`file_get_contents()`	Facile pour lire un fichier	Charge tout en mémoire
`fopen()` + `fwrite()`	Plus de contrôle sur l'écriture	Plus de lignes de code
`fopen()` + `fread()`	Lecture par morceaux possible	Complexe pour les débutants
`fputcsv()`	Spécifique aux fichiers CSV	Nécessite `fopen()`
`json_encode()` + `file_put_contents()`	Facile pour stocker des objets PHP	Conversion JSON obligatoire

---

6. Gestion des permissions et sécurité des fichiers

6.1. Vérifier l'existence d'un fichier

```php
<?php
if (file_exists("mon_fichier.txt")) {
    echo "Le fichier existe.";
} else {
    echo "Le fichier n'existe pas.";
}
?>
```

6.2. Modifier les permissions d'un fichier

```php
<?php
chmod("mon_fichier.txt", 0755);
echo "Permissions modifiées.";
?>
```

Conclusion

L'utilisation de `file_put_contents()` et `file_get_contents()` est la méthode la plus rapide et efficace pour manipuler des fichiers en PHP. Cependant, `fopen()`, `fwrite()`, et `fread()` offrent plus de flexibilité. La gestion des fichiers JSON et CSV est également simple avec les fonctions adaptées. Enfin, la vérification de l'existence et la modification des permissions sont essentielles pour assurer la sécurité des fichiers.