PHP & filesystem

En utilisant un éditeur de texte, quand vous ouvrez, modifiez et sauvegardez un fichier, tout se fait de manière visuelle et automatique. Vous cliquez pour ouvrir, vous tapez pour écrire, et vous sauvegardez. Facile et direct.

En informatique, comme avec PHP, nous faisons ces étapes manuellement avec des commandes spécifiques. Voici pourquoi :

1. **Ouvrir un fichier (fopen())** : C'est comme choisir un fichier dans votre éditeur de texte. Mais en informatique, vous devez dire explicitement quel fichier et comment vous voulez l'utiliser (juste pour lire, pour écrire, etc.).
2. **Lire (fread()) et écrire (fwrite())** : Si dans un éditeur de texte, vous lisez simplement ce qui est à l'écran et tapez pour ajouter du texte, en informatique, vous devez dire combien vous voulez lire et exactement quoi écrire. C'est précis.
3. **Fermer le fichier (fclose())** : Après avoir fini avec un fichier, vous le fermez. Dans un éditeur, c'est souvent juste fermer la fenêtre. En informatique, c'est important de le faire avec une commande pour dire au système "j'ai fini avec ce fichier". Cela aide à éviter des problèmes comme la corruption de fichiers ou l'utilisation excessive de la mémoire de l'ordinateur.

En bref, en programmation, vous faites tout ce que fait un éditeur de texte, mais de manière plus contrôlée et précise, en donnant des instructions spécifiques pour chaque étape.

Quelques fonctions utiles :

- **Streaming mode :**
  - fopen() - Ouvre un fichier (ou URL) (Streaming: Real-time playback on the hard drive). Mode d'ouverture :
    - 'r' Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.
    - 'r+' Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.
    - 'w' Ouvre en écriture seule ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
    - 'w+' Ouvre en lecture et écriture ; le comportement est le même que pour 'w'.
    - 'a' Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.
    - 'a+' Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer. Dans ce mode, la fonction fseek() n'affecte que la position de lecture, les écritures surviennent toujours.
    - 'x' Crée et ouvre le fichier en écriture seulement ; place le pointeur de fichier au début du fichier. Si le fichier existe déjà, fopen() va échouer, en retournant false et en générant une erreur de niveau E_WARNING. Si le fichier n'existe pas, fopen() tente de le créer.
    - 'x+' Crée et ouvre le fichier pour lecture et écriture; le comportement est le même que pour 'x'.
    - fgets() - Récupère la ligne courante (à partir du pointeur de fichier) à partir de l'emplacement du pointeur sur fichier
    - fgetc() - Lit un caractère dans un fichier

    - fwrite() - Écrit un fichier en mode binaire
  - fclose() - Ferme la connexion/pointeur du fichier ouvert par fopen()

- **Direct mode :**

  - file() - Lit l'intégralité du fichier dans un tableau par ligne
  - file_put_contents() - Écrire une chaîne PHP dans un fichier
  - file_get_contents() - Lit le fichier entier dans une chaîne PHP
  - readfile() - Outputs a file (= echo directement le contenu du fichier dans le « buffer » PHP de réponse). Généralement accompagné de headers HTTP pour détail la réponse au client (navigateur).
  - tmpfile() - Crée un fichier temporaire (dans l'emplacement « temporaire » des fichiers de l'OS et des autres programmes)
  - file_exists() - Vérifie l'existence d'un fichier ou d'un répertoire
  - rename() - Renomme un fichier ou un répertoire
  - rmdir() - Supprime un dossier
  - unlink() - Supprime un fichier
  - is_dir() - Indique si le nom de fichier est un dossier
  - is_file() - Indique si le nom de fichier est un fichier normal

Exercice :

1. Dans votre espace PHP de cours, utilisez un controller de Test ou créez un fichier « test.php » dans « /public/ » (ou le dossier d'entrée de l'application) pour s'essayer aux commandes de manipulations des fichiers.
2. Depuis l'exploreur/finder ou votre IDE : créez un dossier « storage » à la racine de votre projet de cours.
3. Dans votre méthode de tests :
    a. Créez un fichier « mytext.txt » dans le dossier storage avec « file_put_contents() » dont le contenu sera « $content = '1\n2\n3' » mais avec des simples côtes « ' » uniquement. Ouvrez le fichier avec votre IDE et observer que les \n n'ont pas été interprété. Pour rappel, « \n » est le caratère de saut de ligne (invisible dans les éditeurs de texte sauf s'il n'est pas bien insérer comme ici).
    b. Faites la même action mais avec un contenu dans des double côtes « " ». Observer que les \n ont cette fois été interprété. Et que chaque chiffre 1, 2 et 3 sont sur des lignes séparées.
    c. Ajoutez un backslash « \ » devant les « \n », vous obtenez normalement ceci « "1\\n2\\n3" ». Que constatez-vous ?
    d. Testez la fonction « copy() » sur « mytext.txt ». Il faudra d'avoir trouver comment utiliser la méthode dans la documentation de PHP.
    e. Supprimer la copie, créé juste avant, avec la fonction PHP « unlink() ».
    f. Ajoutez du contenu au choix à « mytext.txt » avec la fonction « fopen() » (ouvrir) (en mode 'a'), « fputs() » (écrire) et « fclose » (fermer).
    g. Ajoutez maintenant du contenu à votre fichier avec les fonctions « file_get_contents() » et « file_put_contents() ».
    h. Créez un sous-dossier « images » à storage avec « mkdir » en PHP.

i. Testez la fonction « file_exists() » sur « mytext.txt », puis sur le dossier « storage/images/ » et sur un fichier qui n'existe pas « not-exist.txt ».