PHP deuxième partie

I. Dates en PHP

PHP propose des fonctions variées pour manipuler les dates et heures. Voici un aperçu des fonctions les plus utiles.

Fonction date()

La fonction date() retourne une chaîne de caractères représentant la date et l'heure courante. Vous pouvez également spécifier une date antérieure ou postérieure.

Syntaxe

```
date(format, timestamp);
```

- **format** : Chaîne de caractères définissant le format de la date.
- **timestamp** : (facultatif) Un entier représentant le moment souhaité. Si omis, la date courante est utilisée.

Exemples

```
echo date("Y"); // Affiche l'année actuelle, e.g., 2025
echo date("d/m/Y"); // Affiche la date au format jj/mm/aaaa, e.g., 06/03/2025
```

Caractères prédéfinis

Voici quelques caractères prédéfinis que vous pouvez utiliser avec date() :

- d : jour du mois (01 à 31)
- j : jour du mois sans zéro initial (1 à 31)
- m : mois (01 à 12)
- n : mois sans zéro initial (1 à 12)
- Y : année sur 4 chiffres
- y : année sur 2 chiffres
- H : heure au format 24h (00 à 23)
- i : minutes (00 à 59)
- s : secondes (00 à 59)
- w : jour de la semaine (0 pour dimanche, 6 pour samedi)

Fonction checkdate()

Cette fonction vérifie la validité d'une date.

Syntaxe

```
checkdate(mois, jour, année);
```

Exemple

```
if (checkdate(02, 30, 2016)) {
    echo "Date valide.";
} else {
    echo "Date invalide."; // Affiche Date invalide
```

```
}
```

Fonction time()

La fonction time() retourne le timestamp Unix (nombre de secondes depuis le 1er janvier 1970).

Exemple

```
echo time(); // Affiche le timestamp actuel
```

Fonction mktime()

Cette fonction calcule le timestamp Unix à partir des paramètres spécifiés.

Syntaxe

```
mktime(heure, minute, seconde, mois, jour, année);
```

Exemple

```
echo mktime(0, 0, 0, 1, 1, 2016); // Affiche le timestamp correspondant au 1er janvier 2016
```

Manipuler des dates antérieures ou postérieures

Pour obtenir une date spécifique, vous pouvez utiliser date() avec mktime().

Exemple

```
echo date("w ", mktime(0, 0, 0, 1, 1, 2016)); // Affiche le jour de la semaine pour le 01/01/2016
```

Exemple d'application

Pour afficher une date au format "Vendredi 1 janvier 2016", vous pouvez utiliser le code suivant :

```
$jour = array(
    "Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"
);
$mois = array(
    "janvier", "février", "mars", "avril", "mai", "juin",
    "juillet", "août", "septembre", "octobre", "novembre", "décembre"
);

echo $jour[date("w", mktime(0, 0, 0, 1, 1, 2016))] . " " .
    date("j", mktime(0, 0, 0, 1, 1, 2016)) . " " .
    $mois[date("n", mktime(0, 0, 0, 1, 1, 2016))] . " " .
    date("Y", mktime(0, 0, 0, 1, 1, 2016));
```

Résultat

Ce code affiche : **Vendredi 1 janvier 2016**.

II. Fonctions en PHP.

En PHP, créer des fonctions vous permet de réutiliser du code et d'organiser vos traitements.

Création et appel d'une fonction

Syntaxe de base

Pour créer une fonction, utilisez le mot clé `function` :

```
function nomDeLaFonction($arg1, $arg2, ...) {
  // Corps de la fonction
  return $valeurDeRetour; // Optionnel
}
```

- **nomDeLaFonction** : Doit commencer par une lettre ou un underscore (_), suivi de lettres, chiffres ou underscores. Le nom ne doit pas correspondre à une fonction prédéfinie.
- **$arg1, $arg2, ...** : Paramètres optionnels que la fonction peut recevoir.
- **Corps de la fonction** : Instructions exécutées lorsque la fonction est appelée.
- **return** : Permet de retourner une valeur.

Exemple simple

Voici un exemple de fonction qui affiche la date actuelle :

```
<?php
function daj() {
   echo date("d/m/Y");
}

// Appel de la fonction
daj(); // Affiche la date d'aujourd'hui
?>
```

Portée des variables

Les variables définies à l'intérieur d'une fonction sont **locales** à cette fonction. Elles ne sont pas accessibles à l'extérieur.

Exemple de portée

```
<?php
$a = "Salut";

function f() {
   $b = " à tous";
   echo $a; // Erreur : $a n'est pas accessible ici
}
```

```
f();
echo $b; // Erreur : $b n'est pas accessible ici
?>
```

Utilisation de `$GLOBALS` et `global`

Pour accéder à des variables globales dans une fonction, utilisez le tableau `$GLOBALS` ou le mot clé `global`.

Exemple avec $GLOBALS

```
<?php
$a = "Salut";

function f() {
    echo $GLOBALS["a"]; // Accède à la variable globale $a
}

f(); // Affiche: Salut
?>
```

Exemple avec global

```
<?php
$a = "Salut";

function f() {
    global $b; // Déclare $b comme variable globale
    $b = " à tous";
    echo $GLOBALS["a"]; // Accède à la variable globale $a
}

f(); // Affiche: Salut
echo $b; // Affiche: à tous
?>
```

III. Inclusions et arrêts prématurés en PHP

Les inclusions

Lorsqu'un morceau de code est utilisé dans plusieurs fichiers d'un projet web, il est judicieux de le centraliser dans un fichier séparé. Cela simplifie la maintenance, par exemple pour les connexions à la base de données ou les éléments communs comme les en-têtes et les pieds de page.

Structure `include`

La structure `include` permet d'insérer un fichier dans un autre. Voici la syntaxe :

```
<?php
include "fichier_à_inclure"; // Sans parenthèses
// ou
include("fichier_à_inclure"); // Avec parenthèses
```

```
?>
```

Exemple

Contenu du fichier `inc.php` :

```
<?php
echo "Ce texte provient de <b>inc.php</b>";
?>
```

Contenu du fichier `appel.php` :

```
<?php
include("inc.php");
?>
```

Après l'exécution de `appel.php`, vous obtiendrez :

```
Ce texte provient de inc.php
```

Gestion des erreurs

Si le fichier à inclure n'existe pas, PHP affichera un avertissement, mais continuera à exécuter le reste du script :

```
<?php
include("incs.php"); // Fichier inexistant
?>
```

Résultat :

```
Warning: include(incs.php): failed to open stream: No such file or directory
```

Structure `require`

La structure `require` fonctionne de manière similaire à `include`, mais elle arrête l'exécution si le fichier n'est pas trouvé :

```
<?php
require("incs.php"); // Arrête l'exécution si le fichier est inexistant
?>
```

Structures `include_once` et `require_once`

Ces structures fonctionnent comme `include` et `require`, mais elles n'incluent le même fichier qu'une seule fois. Cela aide à éviter des erreurs dues à des inclusions multiples.

Exemple d'utilisation

```
<?php
include_once("inc.php"); // Inclus seulement si ce n'est pas déjà inclus
```

```
require_once("inc.php"); // Fonctionne de la même manière
?>
```

IV. Les arrêts prématurés

Un programme PHP s'exécute normalement du début à la fin. Cependant, il peut s'arrêter prématurément pour diverses raisons.

Erreur fatale

Une erreur fatale empêche le programme de continuer. Le compilateur ne s'exécute pas du tout.

Arrêts programmés

Un développeur peut vouloir arrêter l'exécution d'un programme à un moment donné, ce qui est considéré comme normal.

Arrêt avec `exit()` et `die()`

Les fonctions `exit()` et `die()` arrêtent le script à l'endroit où elles sont appelées et peuvent afficher un message.

Exemple

```
<?php
for ($i = 1; $i <= 10; $i++) {
    if ($i > 5) die("Fin");
    echo "Ligne $i <br />";
}
?>
```

Résultat

```
Ligne 1
Ligne 2
Ligne 3
Ligne 4
Ligne 5
Fin
```

Différence avec `return`

La fonction `return` termine une fonction mais permet au code suivant de s'exécuter. À l'inverse, `exit()` et `die()` arrêtent complètement le script.

Les sessions (et variables de session) en PHP

Principe des sessions

Les sessions permettent de stocker des données sur le serveur, spécifiques à chaque utilisateur. Elles sont utiles pour conserver des informations pendant la navigation,

même après que l'utilisateur ait quitté le site. Chaque session est associée à un identifiant unique, envoyé au client sous forme de cookie.

V. Qu'est-ce qu'un cookie ?

Un cookie est un fichier texte stocké sur le disque dur du client par le site web visité. Il permet d'enregistrer des informations qui peuvent être récupérées lors des visites ultérieures. Par exemple, un site peut mémoriser le thème choisi par l'utilisateur.

Utilisation des sessions

Variable de session `$_SESSION`

La variable `$_SESSION` est une superglobale qui contient les valeurs stockées dans les sessions. Elle est accessible depuis n'importe quelle page du site.

Exemple

```php
<?php
session_start(); // Démarre la session
$_SESSION["theme"] = "vert"; // Crée une variable de session
?>
```

Démarrer ou reprendre une session : `session_start()`

La fonction `session_start()` permet de démarrer une nouvelle session ou de reprendre une session existante. Elle doit être appelée avant d'envoyer toute sortie au navigateur.

Exemple Correct

```php
<?php
session_start(); // Doit être avant toute sortie
?>
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8" />
</head>
<body>
   <!-- Contenu de la page -->
</body>
</html>
```

Erreur de session

Si `session_start()` est appelé après une sortie, vous obtiendrez une erreur :

```php
<!DOCTYPE html>
<?php
session_start(); // Erreur ici
?>
<html>
<head>
```

```
    <meta charset="UTF-8" />
</head>
<body>
    <!-- Contenu de la page -->
</body>
</html>
```

Détruire une session : `session_destroy()`

La fonction `session_destroy()` détruit toutes les données de la session. Elle ne supprime pas le cookie associé, ce qui signifie que la session peut être recréée à la prochaine connexion.

Exemple de code

```
// page1.php
<?php
session_start();
$_SESSION["theme"] = "vert"; // Stocke la variable de session
?>

// page2.php
<?php
session_start();
echo $_SESSION["theme"]; // Affiche "vert"
?>

// page3.php
<?php
session_start();
session_destroy(); // Détruit la session
?>
```

Flux de l'exemple

1. **page1.php** : Démarre la session et définit $_SESSION["theme"] à "vert".
2. **page2.php** : Récupère et affiche la valeur de $_SESSION["theme"], affichant "vert".
3. **page3.php** : Détruit la session, rendant $_SESSION["theme"] inaccessible.

Suppression d'une variable de session

Pour supprimer une donnée spécifique d'une session, il est possible d'utiliser `unset()` :

```
<?php
session_start();
unset($_SESSION["theme"]); // Supprime la variable de session "theme"
?>
```

Les sessions et les cookies sont des outils essentiels pour gérer des données persistantes sur un site web. Les sessions permettent de stocker des informations spécifiques à

chaque utilisateur, et leur gestion via `$_SESSION` et les fonctions associées simplifie le développement d'applications web dynamiques.

Exercice - Authentification en PHP

Cet exercice vous permet de mettre en pratique les sessions pour gérer l'authentification des utilisateurs. Nous allons créer trois fichiers PHP :

1. **login.php** : Page d'authentification.
2. **session.php** : Page protégée, accessible uniquement aux utilisateurs authentifiés.
3. **deconnexion.php** : Page pour déconnecter l'utilisateur et détruire la session.

1. **Code source de login.php :** la page d'authentification :

- Commence par démarrer une session avec session_start().
- Récupère les identifiants saisis par l'utilisateur via $_POST.
- Vérifie si les identifiants correspondent aux valeurs définies (user et 1234).
- Si les identifiants sont corrects, crée une variable de session $_SESSION["autoriser"] et redirige vers session.php.
- Si l'authentification échoue, affiche un message d'erreur.

```php
<?php
session_start();
@$login = $_POST["login"];
@$pass = $_POST["pass"];
@$valider = $_POST["valider"];
$bonLogin = "user";
$bonPass = "1234";
$erreur = "";

if (isset($valider)) {
    if ($login == $bonLogin && $pass == $bonPass) {
        $_SESSION["autoriser"] = "oui";
        header("location:session.php");
        exit(); // Ajout de exit pour s'assurer que le script s'arrête ici
    } else {
        $erreur = "Mauvais login ou mot de passe!";
    }
}
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />

</head>
<body onLoad="document.fo.login.focus()">
    <h1>Authentification</h1>
    <div class="erreur"><?php echo $erreur; ?></div>
    <form name="fo" method="post" action="">
        <input type="text" name="login" placeholder="Login" /><br />
        <input type="password" name="pass" placeholder="Mot de passe" /><br />
```

```
        <input type="submit" name="valider" value="S'authentifier" />
    </form>
</body>
</html>
```

1. **Code source de session.php :** pour la page protégée :

- Vérifie si l'utilisateur est authentifié en vérifiant la variable de session.
- Si l'utilisateur n'est pas authentifié, il est redirigé vers login.php.
- Affiche un message de bienvenue personnalisé en fonction de l'heure.

```
<?php
session_start();
if ($_SESSION["autoriser"] != "oui") {
    header("location:login.php");
    exit(); // Assurez-vous d'arrêter l'exécution après la redirection
}

$bienvenue = (date("H") < 18) ? "Bonjour et bienvenue dans votre espace personnel" :
"Bonsoir et bienvenue dans votre espace personnel";
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <style>
        * { font-family: Arial; }
        body { margin: 20px; }
        a { color: #DD7700; text-decoration: none; }
        a:hover { text-decoration: underline; }
    </style>
</head>
<body>
    <h1><?php echo $bienvenue; ?></h1>
    [ <a href="deconnexion.php">Se déconnecter</a> ]
</body>
</html>
```

3. Code source de deconnexion.php : pour déconnecter l'utilisateur :

```
<?php
session_start();
session_destroy(); // Détruit la session
header("location:login.php"); // Redirige vers login.php
exit(); // Assurez-vous d'arrêter l'exécution après la redirection
?>
```

- Détruit la session active avec session_destroy().
- Redirige l'utilisateur vers login.php.

Ces trois fichiers montrent comment gérer l'authentification des utilisateurs en utilisant des sessions en PHP. Vous pouvez étendre cela en intégrant des bases de données pour stocker les identifiants de manière sécurisée.

VI. Variables d'environnement et constantes PHP prédéfinies

Variables d'environnement

Qu'est-ce qu'une variable d'environnement ?

Les variables d'environnement en PHP sont des données qui stockent des informations relatives à l'environnement d'exécution du code. Elles fournissent des détails sur le serveur, le client, et d'autres éléments pertinents. Par exemple, elles peuvent donner des informations sur le type de serveur, son adresse IP, la version du langage, le type de navigateur, etc.

Fonction `phpinfo()`

La fonction `phpinfo()` permet d'obtenir un récapitulatif complet de la configuration PHP sur le serveur, y compris toutes les variables d'environnement. Pour afficher ces informations, il suffit d'exécuter :

```php
<?php
phpinfo();
?>
```

Il suffit de faire défiler la page jusqu'à la section **Environment** pour voir toutes les variables d'environnement disponibles.

Manipulation des variables d'environnement

Il existe deux manières d'accéder aux variables d'environnement :

1. **Variable superglobale** $_SERVER : Un tableau associatif contenant des informations sur les éléments d'environnement.
2. **Fonction** getenv() : Prend en paramètre le nom de la variable d'environnement et retourne sa valeur.

Exemples

```php
// Utilisation de $_SERVER
echo $_SERVER["HTTP_USER_AGENT"]; // Affiche les informations sur le navigateur

// Utilisation de getenv()
echo getenv("HTTP_USER_AGENT"); // Affiche également les informations sur le navigateur
```

Liste des variables d'environnement courantes

Voici quelques-unes des variables d'environnement les plus utilisées :

- SERVER_NAME : Nom du serveur qui exécute le script.
- SERVER_ADDR : Adresse IP du serveur qui exécute le script.
- SERVER_ADMIN : Email de l'administrateur du serveur.
- REQUEST_METHOD : Méthode utilisée pour la requête HTTP (GET, POST, etc.).
- DOCUMENT_ROOT : Chemin de la racine du serveur.
- HTTP_USER_AGENT : Informations sur le navigateur et le système d'exploitation du client.
- REMOTE_ADDR : Adresse IP du client qui appelle le script.

Exemple d'utilisation :

```
echo $_SERVER["HTTP_USER_AGENT"]; // Affiche les informations sur le navigateur
echo $_SERVER["REQUEST_METHOD"];    // Affiche la méthode de la requête (GET/POST)
echo $_SERVER["HTTP_HOST"];        // Affiche le nom du domaine
```

Constantes PHP

Constantes prédéfinies en PHP

Les constantes en PHP sont des valeurs qui ne changent pas durant l'exécution du script. PHP fournit plusieurs constantes prédéfinies qui peuvent être très utiles.

Principales constantes prédéfinies :

- PHP_OS : Retourne des informations sur le système d'exploitation.
- PHP_VERSION : Retourne la version de PHP installée.
- __FILE__ : Retourne le nom du fichier en cours d'exécution.
- __LINE__ : Retourne le numéro de la ligne en cours d'exécution.

Exemple d'utilisation :

```
echo PHP_OS;       // Affiche le système d'exploitation, par exemple : WINNT
echo PHP_VERSION;   // Affiche la version de PHP, par exemple : 8.0.3
echo __FILE__;      // Affiche le chemin complet du fichier
echo __LINE__;      // Affiche le numéro de la ligne en cours
```

Les variables d'environnement et les constantes prédéfinies en PHP sont des outils puissants qui facilitent la gestion des données relatives à l'environnement d'exécution. Elles peuvent grandement simplifier le développement et améliorer la lisibilité et la maintenance du code.

VII. Expressions régulières - Les regex POSIX et PCRE

Généralités

Une expression régulière (regex) est un motif qui permet de vérifier si une chaîne de caractères correspond à un certain format. Par exemple, pour valider une adresse email, on peut vérifier plusieurs critères comme la présence d'un arobas, d'un point avant le TLD, et l'absence d'espaces.

PHP deuxième partie

POSIX et PCRE

En PHP, il existe deux familles d'expressions régulières :

1. **POSIX (Portable Operating System Interface)** : Une approche plus ancienne, moins performante pour des motifs complexes.
2. **PCRE (Perl Compatible Regular Expressions)** : Plus rapide et recommandée pour sa souplesse et sa richesse syntaxique.

Différences clés

- **Performance** : PCRE est généralement plus rapide, surtout pour des motifs longs et complexes.
- **Syntaxe** : Bien que similaires sur certains aspects, chaque famille a des caractères spéciaux qui lui sont propres.

Expressions régulières POSIX

Fonctions Utilisées

Les fonctions les plus courantes pour les expressions régulières POSIX incluent :

- ereg(), ereg_replace(), split() : Ces fonctions sont désormais obsolètes.
- mb_ereg(), mb_ereg_replace(), mb_split() : Versions multi-octets qui prennent en charge des caractères étendus.

Exemple avec ereg()

```php
$str = "Bonjour";
if (ereg("^B", $str)) {
   echo "La chaîne commence par B";
} else {
   echo "La chaîne ne commence pas par B";
}
```

Caractères spéciaux en POSIX

Caractère	Signification
.	Un caractère quelconque
*	0 ou plusieurs occurrences du motif précédent
+	Au moins une occurrence du motif précédent
^	Début de la chaîne
$	Fin de la chaîne

Séquences utiles en POSIX

Séquence	Signification
[[:alnum:]]	Tous les caractères alphanumériques
[[:alpha:]]	Tous les caractères alphabétiques
[[:digit:]]	Tous les chiffres

Exemple de validation d'email

```
$email = "user@domaine.tld";
if (mb_ereg(".+@.+..+", $email)) {
    echo "Email valide.";
} else {
    echo "Email invalide.";
}
```

L'expression régulière `".+@.+..+"` est trop permissive !

Amélioration avec *filter_var()* (Solution recommandée)

PHP fournit une fonction intégrée plus fiable :

```
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email valide";
} else {
    echo "Email invalide";
}
```

Expressions régulières PCRE

Fonctions Utilisées

Les fonctions essentielles pour PCRE comprennent :

- preg_match()
- preg_replace()
- preg_split()

Exemple avec preg_match()

```
$email = "user@domaine.tld";
if (preg_match("#^[a-zA-Z0-9-_.]+@[a-zA-Z0-9-.]+\.[a-zA-Z]{2,6}$#", $email)) {
    echo "Email valide.";
} else {
    echo "Email invalide.";
}
```

Caractères spéciaux en PCRE

Caractère	Signification
\d	Un chiffre (équivalent à [0-9])
\w	Tout caractère alphanumérique
\s	Un espace

Exemple de `preg_replace()`

```
$email = "user@domaine.tld";
echo preg_replace("#\.tld$#", ".com", $email); // user@domaine.com
```

Exemple de `preg_split()`

```
$str = "Bonjour";
$tab = preg_split("#o#", $str);
print_r($tab); // Array ( [0] => B [1] => nj [2] => ur )
```

Les expressions régulières sont un outil puissant pour la validation et la manipulation de chaînes de caractères en PHP. Les familles POSIX et PCRE offrent des fonctionnalités variées, mais PCRE est généralement la meilleure option en raison de sa rapidité et de sa flexibilité. Pour des applications nécessitant une validation de données robuste, l'utilisation des expressions régulières est essentielle.

VIII. Manipulation des fichiers en PHP

En PHP, la manipulation des fichiers se réfère principalement à l'interaction avec des fichiers texte. Ces fichiers peuvent contenir des données à afficher sur une page Web ou servir d'espace de stockage pour les résultats d'exécution de scripts PHP.

Fonctions agissant sur les fichiers

1. Lecture de fichiers

file_get_contents()

Cette fonction permet de retourner le contenu d'un fichier sous forme de chaîne de caractères.

```
echo file_get_contents("document.txt");
```

2. Écriture de fichiers

file_put_contents()

Cette fonction écrit le contenu spécifié dans un fichier. Si le fichier n'existe pas, il sera créé.

```
$str = "Bonjour à tous";
file_put_contents("contenu.txt", $str);
```

Fonctions de test et d'évaluation de fichiers

- file_exists($fichier) : Vérifie si le fichier existe.
- filesize($fichier) : Retourne la taille du fichier en octets.
- filetype($fichier) : Retourne le type du fichier.
- is_file($fichier) : Vérifie si c'est un fichier (et non un répertoire).
- unlink($fichier) : Supprime le fichier.

Exemple

```
$fichier = "index.php";
if (is_file($fichier)) {
    echo number_format(filesize($fichier), 0, "", " ") . " octets";
}
```

Fonctions d'accès aux fichiers

1. Ouvrir et fermer des fichiers

fopen()

Ouvre un fichier avec un mode spécifique et retourne un identifiant de fichier.

Mode d'ouverture	Signification
r	Lecture seule
r+	Lecture et écriture
w	Création et écriture (écrase)
a	Ajout à la fin du fichier

fclose()

Ferme le fichier ouvert.

```
$fp = fopen("fichier.txt", "r");
fclose($fp);
```

2. Lecture et écriture de fichiers

fgets()

Lit une ligne du fichier.

```
$fp = fopen("fichier.txt", "r");
$ligne = fgets($fp);
fclose($fp);
```

fputs()

Écrit une chaîne dans le fichier à la position actuelle du pointeur.

```
$fp = fopen("fichier.txt", "a");
fputs($fp, "Ajout de texte.\n");
fclose($fp);
```

3. Positionnement du pointeur

fseek()

Déplace le pointeur à une position spécifique.

```
fseek($fp, 0); // Retourne au début du fichier
```

Exemple d'application : Compteur de visites

Voici un exemple simple d'un compteur de visites utilisant un fichier texte :

```
session_start();
$fp = fopen("compteur.txt", "r+");
$nbr = fgets($fp, 10); // Lit le nombre de visites

if ($_SESSION["dejaVisitee"] != "oui") {
    $nbr += 1; // Incrémente le compteur
    fseek($fp, 0); // Retourne au début du fichier
    fputs($fp, $nbr); // Écrit le nouveau nombre de visites
    $_SESSION["dejaVisitee"] = "oui"; // Marque la session comme visitée
}

echo "Cette page a été visitée $nbr fois.";
fclose($fp);
```

Remarques

- Créez le fichier compteur.txt et initialisez-le à 0.
- Assurez-vous que les permissions du fichier permettent la lecture et l'écriture par le script PHP.

La manipulation des fichiers en PHP offre une méthode simple et efficace pour gérer des données persistantes. Grâce aux diverses fonctions disponibles, il est possible de lire, écrire, et manipuler des fichiers texte facilement.

IX. Chargement de fichiers en PHP

Le chargement de fichiers (upload) permet d'envoyer des fichiers au serveur via un formulaire web. Contrairement au téléchargement (download), qui est plus connu, l'upload est essentiel pour des fonctionnalités telles que l'ajout de photos de profil sur des réseaux sociaux ou l'envoi de pièces jointes par email.

PHP deuxième partie

Le chargement de fichiers se fait via le protocole HTTP, et non FTP. Pour permettre l'upload, le formulaire HTML doit être configuré correctement.

Partie HTML : Le Formulaire

Pour créer un formulaire permettant le chargement de fichiers, il faut utiliser la méthode POST et spécifier l'attribut enctype avec la valeur multipart/form-data.

Exemple de Formulaire

```
<form method="post" action="" enctype="multipart/form-data">
  <input type="hidden" name="MAX_FILE_SIZE" value="2000000" />
  <input type="file" name="monfichier" /><br />
  <input type="submit" name="valider" value="Uploader" />
</form>
```

Dans cet exemple, le champ MAX_FILE_SIZE limite la taille du fichier à 2 Mo (2 000 000 octets).

Partie PHP : Traitement de l'Upload

La Variable $_FILES

En PHP, la superglobale $_FILES est un tableau associatif qui contient des informations sur le fichier uploadé. Elle a les clés suivantes :

- **name** : Le nom d'origine du fichier.
- **tmp_name** : Le chemin temporaire sur le serveur.
- **type** : Le type MIME du fichier.
- **size** : La taille du fichier en octets.
- **error** : Un code d'erreur si le chargement échoue.

Exemple de Traitement

Voici un exemple complet qui permet aux utilisateurs d'uploader des images JPEG ou PNG :

```
<?php
$message = "";
if (isset($_POST["valider"])) {
    if (!preg_match("#jpeg|png#", $_FILES["monfichier"]["type"])) {
        $message = '<span class="nook">Format invalide!</span>';
    } elseif ($_FILES["monfichier"]["size"] > 2000000) {
        $message = '<span class="nook">Taille trop grande!</span>';
    } else {
        $message = 'Nom du fichier : <b>' . $_FILES["monfichier"]["name"] . '</b><br
/>';
        $message .= 'Nom temporaire : <b>' . $_FILES["monfichier"]["tmp_name"] .
'</b><br />';
        $message .= 'Type : <b>' . $_FILES["monfichier"]["type"] . '</b><br />';
        $message .= 'Taille : <b>' . $_FILES["monfichier"]["size"] . ' octets</b><br />';
```

```
        if (move_uploaded_file($_FILES["monfichier"]["tmp_name"], "uploads/" .
$_FILES["monfichier"]["name"])) {
            $message .= '<span class="ok">Image chargée avec succès</span>';
        }
    }
}
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <style>
        * {
            font-family: Arial, sans-serif;
        }
        .nook {
            color: #DD0000;
        }
        .ok {
            color: #00DD00;
        }
    </style>
</head>
<body>
    <form method="post" action="" enctype="multipart/form-data">
        <input type="hidden" name="MAX_FILE_SIZE" value="2000000" />
        <input type="file" name="monfichier" /><br />
        <input type="submit" name="valider" value="Uploader" />
    </form>
    <?php
        echo $message;
    ?>
</body>
</html>
```

Vérifications

Bien que le champ MAX_FILE_SIZE limite la taille du fichier côté client, il est important de vérifier également la taille et le type du fichier côté serveur pour éviter les abus.

Configuration PHP

La directive upload_max_filesize dans le fichier php.ini définit la taille maximale des fichiers uploadés. Exemple :

```
upload_max_filesize = 2M
```

Gestion des Permissions

Pour gérer les droits d'accès sur le fichier uploadé, utilisez la fonction chmod().

Exemple de chmod()

```
chmod("image.jpg", 0744);
```

Dans cet exemple, le propriétaire a tous les droits (lecture, écriture, exécution), tandis que les autres utilisateurs ont uniquement le droit de lecture.

Le chargement de fichiers en PHP est une fonctionnalité puissante qui permet aux utilisateurs de soumettre des fichiers au serveur. Grâce aux vérifications sur le type et la taille des fichiers, ainsi qu'à la gestion des permissions, il est possible d'assurer un traitement sécurisé et efficace des fichiers uploadés.

X. Connexion à une base de données MySQL en PHP (Extension MySQLi)

La connexion à une base de données est une étape cruciale pour développer des applications web dynamiques. Cette section vous guide à travers l'utilisation de l'extension MySQLi pour interagir avec une base de données MySQL. Bien que MySQLi soit fonctionnel et largement utilisé, il est recommandé d'explorer l'objet PDO (PHP Data Objects) pour une flexibilité accrue et une abstraction de base de données.

Pourquoi Utiliser une Base de Données ?

Les bases de données permettent de stocker, gérer et interroger des données de manière structurée. Elles sont essentielles pour des applications nécessitant un stockage persistant, comme des systèmes de gestion de contenu, des sites e-commerce, ou des applications de réseau social.

Avantages des Bases de Données

- **Structuration des données** : Organisées en tables, les données sont faciles à gérer.
- **Interrogation rapide** : SQL permet d'effectuer des requêtes complexes efficacement.
- **Gestion des relations** : Les bases de données relationnelles gèrent les relations entre différentes entités.

Bases de données et PHP

PHP offre plusieurs méthodes pour interagir avec des bases de données. Les trois principales sont :

1. **Extension MySQL originale** : Désormais obsolète.
2. **Extension MySQLi** : Plus moderne, elle supporte les requêtes orientées objet.
3. **Objet PDO** : Idéal pour l'abstraction de base de données, supportant plusieurs SGBD.

Extension MySQLi

Configuration de l'Extension MySQLi

PHP deuxième partie

L'extension MySQLi est activée par défaut dans PHP. Assurez-vous que la ligne suivante est présente dans votre fichier `php.ini` :

```
extension=php_mysqli.dll
```

Prérequis

Avant de commencer avec MySQLi, il est essentiel de maîtriser le langage SQL, en particulier les opérations CRUD (Create, Read, Update, Delete).

Création de la Base de Données

Utilisation de PHPMyAdmin

Il est recommandé d'utiliser PHPMyAdmin pour gérer vos bases de données. Cela vous permet de créer des bases de données et des tables facilement sans avoir à écrire des commandes SQL à la main.

Exemple de Création d'une Table

Voici un exemple de création d'une table `personnes` :

```
CREATE TABLE `personnes` (
  `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  `date` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
  `nom` VARCHAR(40) NOT NULL,
  `prenom` VARCHAR(40) NOT NULL,
  `login` VARCHAR(40) NOT NULL,
  `pass` VARCHAR(40) NOT NULL,
  PRIMARY KEY (`id`)
);
```

Connexion au Serveur MySQL

Établir la Connexion

Pour se connecter à la base de données, utilisez la fonction `mysqli_connect()`. Voici comment établir une connexion :

```
$id = mysqli_connect("localhost", "root", "root", "mabase") or die("Erreur de
connexion : " . mysqli_connect_error());
```

Gestion des Erreurs

Il est crucial de gérer les erreurs de connexion pour éviter des comportements inattendus dans votre application :

```
if (!$id) {
    die("Échec de la connexion : " . mysqli_connect_error());
}
```

PHP deuxième partie

Exécuter des Opérations CRUD

1. Création d'Enregistrements

Pour insérer un nouvel enregistrement dans la table `personnes`, utilisez :

```
$nom = "Einstein";
$prenom = "Albert";
$login = "a.einstein";
$pass = md5("2020");

$query = "INSERT INTO personnes (nom, prenom, login, pass) VALUES ('$nom',
'$prenom', '$login', '$pass')";
if (mysqli_query($id, $query)) {
    echo "Nouvel enregistrement créé avec succès.";
} else {
    echo "Erreur : " . mysqli_error($id);
}
```

2. Lecture d'Enregistrements

Pour lire des enregistrements de la table, utilisez une requête de sélection :

```
$query = "SELECT nom, prenom FROM personnes";
$result = mysqli_query($id, $query);

if (mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result)) {
        echo "Nom : " . $row["nom"] . " - Prénom : " . $row["prenom"] . "<br>";
    }
} else {
    echo "Aucun enregistrement trouvé.";
}
```

3. Mise à Jour d'Enregistrements

Pour mettre à jour un enregistrement existant :

```
$new_pass = md5("ABCD");
$query = "UPDATE personnes SET pass='$new_pass' WHERE id=1";

if (mysqli_query($id, $query)) {
    echo "Enregistrement mis à jour avec succès.";
} else {
    echo "Erreur : " . mysqli_error($id);
}
```

4. Suppression d'Enregistrements

Pour supprimer un enregistrement :

```php
$query = "DELETE FROM personnes WHERE id=1";

if (mysqli_query($id, $query)) {
    echo "Enregistrement supprimé avec succès.";
} else {
    echo "Erreur : " . mysqli_error($id);
}
```

Exécution de Requêtes de Sélection Avancées

Sélection avec Conditions

Vous pouvez également ajouter des conditions à vos requêtes :

```php
$query = "SELECT * FROM personnes WHERE nom = 'Einstein'";
$result = mysqli_query($id, $query);

if (mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result)) {
        echo "Nom : " . $row["nom"] . " - Prénom : " . $row["prenom"] . " - Login : " .
$row["login"] . "<br>";
    }
} else {
    echo "Aucun résultat trouvé.";
}
```

Compter les Résultats

Pour connaître le nombre d'enregistrements retournés par une requête :

```php
$nbr_resultats = mysqli_num_rows($result);
echo "Nombre d'enregistrements trouvés : " . $nbr_resultats;
```

Cette partie a présenté les bases de l'utilisation de l'extension MySQLi pour interagir avec une base de données MySQL. Bien que MySQLi soit utile, il est conseillé d'explorer l'objet PDO pour une gestion plus avancée et sécurisée des bases de données. Cela inclut des fonctionnalités telles que la protection contre les injections SQL et une meilleure gestion des erreurs.