Formation Développement Web : Backend : PHP

Partie : 3

I. Les variables globale

Les variables globales en PHP sont des variables pré-définies accessibles depuis n'importe quelle partie de votre script. Elles sont largement utilisées pour récupérer des données à partir de requêtes HTTP ou pour stocker des informations importantes.

A. $_GET :

$_GET : Cette variable contient les données envoyées via la méthode HTTP GET. Elle est souvent utilisée pour récupérer des données d'URL.

B. $_POST

$_POST : Cette variable contient les données envoyées via la méthode HTTP POST. Elle est principalement utilisée pour envoyer des données depuis un formulaire HTML.

C. $_COOKIES

$_COOKIE : Elle est utilisée pour gérer les cookies stockés sur le navigateur du client.

D. $_SERVER

$_SERVER : Cette variable contient des informations sur le serveur web, telles que l'URL demandée, le type de requête, l'adresse IP du client, etc.

E. $_REQUEST

$_REQUEST : Elle contient des données provenant de plusieurs sources, notamment $_GET, $_POST, et $_COOKIE. Cependant, il est recommandé d'utiliser les variables spécifiques plutôt que $_REQUEST pour des raisons de sécurité.

F. $_SESSION

$_SESSION : Cette variable est utilisée pour gérer les sessions utilisateur, permettant de stocker des informations d'une page à l'autre.

II. Les Requêtes HTTP GET et POST

Les requêtes HTTP GET et POST sont les deux méthodes les plus couramment utilisées pour envoyer des données depuis un client (navigateur) vers un serveur web. Elles sont souvent utilisées pour transmettre des informations à un script PHP.

A. Requête HTTP GET

- Utilisée pour envoyer des données via l'URL.
- Les données sont visibles dans l'URL.
- Convient aux données non sensibles.
- Exemple : http://mondomain.com/page.php?nom=rachid&age=25

B. Requête HTTP POST

- Utilisée pour envoyer des données dans le corps de la requête.
- Les données ne sont pas visibles dans l'URL.
- Convient aux données sensibles, telles que les mots de passe.
- Souvent utilisée avec les formulaires HTML.

C. Intégration de GET et POST dans un Formulaire HTML

Pour utiliser les méthodes GET et POST dans un formulaire HTML, vous devez spécifier l'attribut "method" de la balise <form>. L'attribut "method" détermine comment les données du formulaire seront envoyées au serveur.

1. Exemple 1

< !-- Dans cet exemple, lorsque l'utilisateur soumet le formulaire, les données seront ajoutées à l'URL de la page de traitement (traitement.php) sous forme de paramètres. -->

```
<form action="traitement.php" method="GET">
  <label for="nom">Nom :</label>
  <input type="text" name="nom" id="nom"><br><br>

  <label for="age">Âge :</label>
  <input type="text" name="age" id="age"><br><br>

  <input type="submit" value="Envoyer">
</form>
```

2. Exemple 2

< !-- Dans cet exemple, lorsque l'utilisateur soumet le formulaire, les données seront envoyées dans le corps de la requête HTTP vers la page de traitement (traitement.php), où vous pourrez les récupérer en utilisant PHP. -->

```
<form action="traitement.php" method="POST">
  <label for="nom">Nom :</label>
  <input type="text" name="nom" id="nom"><br><br>

  <label for="age">Âge :</label>
  <input type="text" name="age" id="age"><br><br>

  <input type="submit" value="Envoyer">
</form>
```

III. Utilisation des Variables Globales

Pour récupérer des données envoyées via $_GET, $_POST, $_COOKIE, $_SERVER, $_REQUEST, ou $_SESSION, vous pouvez simplement accéder aux clés correspondantes de ces variables globales. Par exemple, pour récupérer le nom envoyé via $_GET :

A. Exemple

```
<?php
    $nom = $_GET['nom'];
    $adresseIP = $_SERVER['REMOTE_ADDR'];
?>
```

IV. Manipulation des Cookies en PHP

Les cookies sont de petits fichiers de données stockés sur l'ordinateur du client. Ils sont largement utilisés pour stocker des informations temporaires, telles que les préférences de l'utilisateur, les paniers d'achat, et plus encore. En PHP, vous pouvez manipuler les cookies à l'aide de fonctions dédiées.

A. Création de Cookies

Pour créer un cookie en PHP, vous utilisez la fonction setcookie(). Cette fonction prend généralement trois arguments principaux :

- Le nom du cookie.
- La valeur du cookie sous forme de texte.
- La durée de validité du cookie (en secondes depuis l'heure actuelle).

1. Syntaxe

```
<?php
    setcookie($nomCookie, $valeurCookie, time() + $duree);
?>
```

2. La fonction time()

La fonction time() en PHP renvoie l'heure actuelle sous la forme d'un timestamp Unix, qui est le nombre de secondes écoulées depuis le 1er janvier 1970 à 00:00:00 UTC (temps universel coordonné).

3. Exemple

```
<?php
    // Nom du cookie
    $nomCookie = "monCookie";

    // Valeur du cookie
    $valeurCookie = "Ceci est la valeur de mon cookie";

    // Durée de validité du cookie (3600 secondes équivaut à 1 heure)
    $duree = 3600;

    // Création du cookie
    setcookie($nomCookie, $valeurCookie, time() + $duree);
?>
```

B. Récupération de Cookies

Pour récupérer la valeur d'un cookie, vous pouvez utiliser la superglobale $_COOKIE

1. Exemple

```
<?php
    // Nom du cookie à récupérer
    $nomCookie = "monCookie";

    if (isset($_COOKIE[$nomCookie])) {
       $valeurCookie = $_COOKIE[$nomCookie];
       echo "La valeur du cookie est : " . $valeurCookie;
    } else {
       echo "Le cookie n'existe pas.";
    }
?>
```

C. Suppression de Cookies

Pour supprimer un cookie, vous pouvez utiliser la fonction setcookie() avec une date d'expiration passée. Cela signifie que le cookie sera immédiatement expiré et supprimé du navigateur du client.

1. Exemple

```
<?php
    // Suppression du cookie en le faisant expirer immédiatement (en passant une date passée)
    setcookie($nomCookie, "", time() - 3600);
?>
```

V. Inclusion de Fichiers en PHP

En PHP, vous pouvez inclure des fichiers externes dans vos scripts pour réutiliser du code ou séparer votre code en plusieurs fichiers pour une meilleure organisation. Il existe quatre méthodes principales pour inclure des fichiers : **require, include, require_once,** et **include_once**. Chacune de ces méthodes a un comportement légèrement différent.

A. require

'require' est utilisé pour inclure un fichier dans votre script PHP. Si le fichier spécifié n'est pas trouvé, une erreur fatale est générée, et le script s'arrête. C'est utile lorsque le fichier inclus est essentiel à l'exécution du script.

1. Exemple

```
<?php
    // Exemple d'utilisation de require
    require("monFichier.php");
?>
```

B. includes

'include' est similaire à **'require'**, mais s'il ne trouve pas le fichier spécifié, il génère une erreur de type avertissement (warning) au lieu d'une erreur fatale. Le script continue de s'exécuter après l'erreur.

1. Exemple

```
<?php
    // Exemple d'utilisation de include
    include("monFichier.php");
?>
```

C. require_once

'require_once' est similaire à **'require'**, mais il vérifie d'abord si le fichier a déjà été inclus. S'il a déjà été inclus, il n'est pas inclus à nouveau. Cela évite d'inclure le même fichier plusieurs fois.

1. Exemple

```
<?php
    // Exemple d'utilisation de require_once
    require_once("monFichier.php");
?>
```

D. Includes_once

'include_once' est similaire à **'include'**, mais il vérifie d'abord si le fichier a déjà été inclus. S'il a déjà été inclus, il n'est pas inclus à nouveau. Cela évite d'inclure le même fichier plusieurs fois.

1. Exemple

```
<?php
    // Exemple d'utilisation de include_once
    Include_once("monFichier.php");
?>
```

E. Resumé

Méthode	Comportement en cas de fichier introuvable	Inclusion multiple du même fichier	Comportement en cas d'erreur
require	Erreur fatale	Oui	Arrête le script
include	Avertissement (warning)	Oui	Continue l'exécution du script
require_once	Erreur fatale si déjà inclus	Non	Arrête le script
include_once	Avertissement si déjà inclus	Non	Continue l'exécution du script

VI. Hachage de Mots de Passe et Vérification

A. Sécurité des Mots de Passe

1. Utiliser un Algorithme de Hachage Fort

Assurez-vous d'utiliser un algorithme de hachage sécurisé, tel que **'PASSWORD_DEFAULT'** avec **'password_hash'**, qui sera mis à jour automatiquement avec de meilleures options de hachage dans les futures versions de PHP.

2. Sel Unique

Chaque hachage doit avoir un sel unique. **'password_hash'** génère automatiquement un sel aléatoire, ce qui rend les attaques par hachage plus difficiles.

3. Ne Jamais Stocker de Mots de Passe en Clair

Ne stockez jamais de mots de passe en texte brut dans la base de données. Stockez uniquement les hachages.

4. Validation des Mots de Passe Forts

Encouragez les utilisateurs à créer des mots de passe forts en imposant des exigences telles que des caractères majuscules, minuscules, des chiffres et des caractères spéciaux.

B. Hachage de Mots de Passe

Le hachage de mots de passe consiste à convertir un mot de passe en une chaîne de caractères aléatoire (hachage) à l'aide d'un algorithme de hachage. Cette chaîne de caractères est stockée dans la base de données plutôt que le mot de passe brut. Lorsque l'utilisateur se connecte, son mot de passe est haché et comparé au hachage stocké.

1. Fonction de Hachage Password Hash

PHP propose la fonction **'password_hash'** pour hacher un mot de passe. Elle utilise un algorithme de hachage sécurisé et génère automatiquement un sel (une valeur aléatoire) pour chaque hachage, ce qui renforce la sécurité.

a) Exemple

```
<?php
    $motDePasse = "monMotDePasseSecret";
    $motDePasseHache = password_hash($motDePasse, PASSWORD_DEFAULT);
    echo $motDePasse ;
    echo "<br/>" ;
    echo $motDePasseHache ;
?>
```

2. Vérification du Mot de Passe

Lorsque l'utilisateur tente de se connecter, vous pouvez vérifier son mot de passe en utilisant la fonction **'password_verify'**. Elle compare le mot de passe fourni par l'utilisateur avec le hachage stocké.

a) Exemple

```
<?php
    $motDePasseSaisi = " monMotDePasseSecret ";
    if (password_verify($motDePasseSaisi, $motDePasseHache)) {
       echo  "Mot de passe correct, permettre la connexion" ;
    } else {
       echo  "Mot de passe incorrect, afficher un message d'erreur";
    }
?>
```

VII. Redirection avec header()

La redirection est un mécanisme qui permet de diriger les utilisateurs vers une autre page web après le traitement d'une requête. En PHP, la fonction header() est couramment utilisée pour effectuer des redirections.

A. Rediriger vers une Autre Page

La fonction header() est utilisée pour envoyer des en-têtes HTTP spéciaux qui indiquent au navigateur de l'utilisateur de rediriger vers une autre URL.

1. Exemple

```
<?php
    header("Location: nouvelle_page.php");
?>
```

B. Redirection avec un Délai

Vous pouvez également ajouter un délai avant la redirection

1. Exemple

```
<?php
    header("Refresh: 5; URL=nouvelle_page.php");
?>
```

C. Gestion des Erreurs de Redirection

Il est important de gérer les erreurs de redirection. Si une redirection échoue, vous risquez de laisser l'utilisateur sur une page vide ou de provoquer des erreurs.

Pour éviter cela, vous pouvez ajouter une vérification après l'appel à header() pour vous assurer que la redirection a été effectuée avec succès.

1. Exemple

```
<?php
    if (header("Location: nouvelle_page.php")) {
       // La redirection a réussi
       exit;
    } else {
       // Gestion de l'erreur de redirection
       echo "Redirection impossible.";
    }
?>
```