PHP : Introduction - Partie 2

Les Tableaux

Un tableau est une structure de données qui vous permet de stocker plusieurs valeurs dans une seule variable. En PHP, il existe plusieurs types de tableaux, notamment les tableaux indexés, les tableaux associatifs et les tableaux multidimensionnels. Dans ce cours, nous allons explorer ces types de tableaux et apprendre comment les utiliser en PHP.

Tableaux Indexés
Un tableau indexé est un tableau où chaque élément est associé à un indice numérique (0, 1, 2, 3, ...). Voici comment vous déclarez et utilisez un tableau indexé en PHP :

```php
<?php


// Déclaration d'un tableau indexé
$fruits = array("Pomme", "Banane", "Orange", "Fraise");

// Accès aux éléments du tableau
echo $fruits[0]; // Affiche "Pomme"
echo $fruits[1]; // Affiche "Banane"
```

Les Tableaux

Parcours d'un Tableau Indexé

Vous pouvez parcourir un tableau indexé en utilisant une boucle for ou foreach :

```php
// Utilisation de for :
for ($i = 0; $i < count($fruits); $i++) {
    echo $fruits[$i] . "<br>";
}

// Utilisation de foreach :
foreach ($fruits as $fruit) {
    echo $fruit . "<br>";
}
```

Tableaux Associatifs

Un tableau associatif est un tableau où chaque élément est associé à une clé (ou un nom). Voici comment vous déclarez et utilisez un tableau associatif en PHP :

```php
// Déclaration d'un tableau associatif
$etudiant = array("nom" => "Alice", "age" => 25, "ville" => "Paris");

// Accès aux éléments du tableau
echo $etudiant["nom"];   // Affiche "Alice"
echo $etudiant["age"];   // Affiche 25
```

Les Tableaux

Parcours d'un Tableau Associatif

Une boucle foreach en PHP est utilisée pour parcourir des tableaux (**tableaux indexés, tableaux associatifs ou tableaux multidimensionnels**) et traiter chaque élément du tableau de manière séquentielle. La boucle **foreach** est particulièrement utile pour itérer sur les éléments d'un tableau sans avoir à gérer les indices manuellement. Voici comment fonctionne une boucle foreach en PHP.

```php
foreach ($etudiant as $cle => $valeur) {
    echo $cle . ": " . $valeur . "<br>";
}
```

Dans cet exemple, la boucle foreach parcourt le tableau **$etudiant**, où **$cle** prend la clé de **chaque** élément et **$valeur** prend la valeur correspondante. Le code à l'intérieur de la boucle affiche la clé suivie de la valeur pour chaque paire clé-valeur dans le tableau.

Introduction à $_POST

$_POST est une superglobale en PHP qui est utilisée pour collecter des données envoyées depuis un formulaire HTML avec la méthode **HTTP POST**. Les données envoyées via **$_POST** sont généralement utilisées pour traiter des **formulaires**, telles que la saisie d'un nom d'utilisateur et d'un mot de passe sur une page de connexion, ou la soumission d'un formulaire de contact sur un site web.

Lorsqu'un formulaire HTML est soumis avec la méthode **POST**, les données saisies dans les champs du formulaire sont envoyées au serveur, qui les stocke dans **l'array $_POST** pour que vous puissiez les manipuler en PHP.

Pour utiliser $_POST, vous devez d'abord créer un formulaire HTML avec les champs nécessaires. Voici un exemple simple de formulaire de connexion :

```html
<form method="POST">
    <label for="username">Nom d'utilisateur :</label>
    <input type="text" name="username" id="username">
    <br>
    <label for="password">Mot de passe :</label>
    <input type="password" name="password" id="password">
    <br>
    <input type="submit" value="Se connecter">
</form>
```

Introduction à $_POST

Traitement des Données avec $_POST

Une fois que le formulaire est soumis, vous pouvez accéder aux données saisies en utilisant $_POST :

```php
if ( !empty($_POST) ) {

    //  !empty est une fonction empty() qui vérifie si une variable est vide.
    // à l'aide du "!" précédant empty nous vérifions le contraire, soit, s'il n'est PAS vide.

    $username = $_POST["username"];
    $password = $_POST["password"];

    // Vous pouvez maintenant traiter les données (vérifier l'authentification, enregistrer dans une base de données, etc.).

    // Par exemple, afficher les données :
    echo "Nom d'utilisateur : " . $username . "<br>";
    echo "Mot de passe : " . $password;
}
```

$_POST en PHP est un outil puissant pour **collecter et traiter les données envoyées par les utilisateurs via des formulaires HTML**. Cependant, il est essentiel de prendre des mesures de **sécurité** pour valider et **protéger** ces données. En suivant les meilleures pratiques de sécurité et de validation des données, vous pouvez utiliser $_POST de manière sécurisée dans vos applications web PHP.