M KARA SAMB
FORMATEUR-DEVELOPPEUR
TEL : 77543 92 82

Présentation du langage et premier site en PHP

PHP (Hypertext Preprocessor) est un langage de programmation côté serveur utilisé pour développer des sites web dynamiques. Il est facile à apprendre pour les débutants et très populaire pour la création de sites web en raison de sa compatibilité avec les systèmes de gestion de contenu (CMS) tels que WordPress.

Pour créer votre premier site en PHP, vous pouvez suivre les étapes suivantes:

Configurez votre environnement de développement en installant un serveur web (par exemple, Apache) et un interpréteur PHP sur votre ordinateur.

Créez un nouveau fichier .php et écrivez du code PHP.

Utilisez des balises PHP pour afficher du contenu HTML sur votre page web.

Téléchargez le fichier sur votre serveur web et accédez-y via un navigateur pour voir le résultat.

Exemple de code PHP simple pour afficher "Bonjour, monde !" sur une page web:

```
<!DOCTYPE html>
<html>
<head>
 <title>Mon premier site en PHP</title>
</head>
<body>
 <?php
echo "Bonjour, mes étudiants !";
 ?>
</body>
</html>
```

Il est important de noter que le code PHP doit être exécuté sur un serveur web pour produire le résultat final, et non sur votre ordinateur local.

Automatisation d'une page Web

L'automatisation d'une page web consiste à automatiser certaines tâches répétitives pour simplifier et accélérer le processus de développement web. Il existe plusieurs façons d'automatiser une page web :

Utilisation de frameworks : Les frameworks tels que Laravel, Ruby on Rails, Django, etc. fournissent des outils et des méthodologies pour automatiser les tâches courantes, telles que la gestion des formulaires, la validation des données, la génération de modèles de base de données, etc.

Utilisation de CMS : Les systèmes de gestion de contenu (CMS) tels que WordPress, Drupal, Joomla, etc. peuvent également aider à automatiser certaines tâches, notamment la gestion des pages, des articles, des images, etc.

Utilisation de scripts : Vous pouvez écrire des scripts pour automatiser certaines tâches spécifiques à votre site web, tels que la mise à jour de données en temps réel, la génération de pages dynamiques, etc.

Utilisation de task runners : Les task runners tels que Grunt, Gulp, etc. peuvent également être utilisés pour automatiser certaines tâches, telles que la compression de fichiers CSS et JavaScript, la génération de pages optimisées pour le référencement, etc.

En général, l'automatisation d'une page web peut considérablement améliorer la qualité, la vitesse et la flexibilité de votre développement web.

Les principes client serveur

Le modèle client-serveur décrit une architecture informatique dans laquelle une partie (le serveur) fournit des services à une autre partie (le client). Les principes fondamentaux du modèle client-serveur incluent :

Séparation des rôles : Le serveur s'occupe de fournir des services, tandis que le client en fait la demande.

Communications asynchrones : Les communications entre le client et le serveur se font de manière asynchrone, ce qui signifie que le client n'a pas besoin d'attendre la réponse du serveur pour continuer à fonctionner.

Scalabilité : Le modèle client-serveur est conçu pour permettre une scalabilité horizontale, ce qui signifie que plusieurs serveurs peuvent être utilisés pour répondre aux demandes des clients.

Fiabilité : Le modèle client-serveur permet de concevoir des systèmes fiables, en évitant les erreurs de code sur le client et en permettant une maintenance facile du serveur.

Flexibilité : Le modèle client-serveur permet de concevoir des systèmes flexibles, car le client peut être développé séparément du serveur, ce qui permet de faire évoluer chacun des composants indépendamment.

Le modèle client-serveur est largement utilisé dans de nombreux systèmes informatiques, tels que les sites web, les applications de messagerie, les applications de jeux en ligne, les systèmes de gestion de bases de données, etc. Il permet de concevoir des systèmes distribués, en

répartissant les tâches entre le client et le serveur pour une meilleure efficacité et une meilleure utilisation des ressources.

Quelques éléments de base du langage de programmation PHP :

Syntaxe : PHP utilise une syntaxe similaire à celle de C et de Java, avec des instructions terminées par un point-virgule.

Variables : PHP utilise des variables pour stocker des valeurs. Les variables en PHP sont précédées par un signe dollar **($)**.

Types de données : PHP prend en charge de nombreux types de données, tels que les chaînes de caractères, les nombres entiers, les nombres flottants, les booléens, les tableaux, etc.

Opérateurs : PHP prend en charge de nombreux opérateurs, tels que l'opérateur d'affectation (=), les opérateurs arithmétiques (+, -, *, /), les opérateurs de comparaison (==, !=, >, <, >=, <=), etc.

Structures de contrôle : PHP prend en charge les structures de contrôle telles que les instructions conditionnelles (if, else, elseif), les boucles (for, while, do while), les instructions de saut (break, continue, return), etc.

Fonctions : PHP prend en charge les fonctions, qui sont des blocs de code encapsulés pouvant être réutilisés plusieurs fois dans le même script.

Ces éléments sont les fondements du langage PHP et il est important de les comprendre avant de commencer à développer des applications plus complexes en utilisant ce langage.

Exemple simple de code PHP :

```
<?php

$nom = "Tata";
$age = 3;

echo "Bonjour, je m'appelle " . $nom . " et j'ai " . $age . " ans.";

?>
```

Dans cet exemple, nous déclarons deux variables, $nom et $age, et nous les utilisons pour construire une chaîne de caractères qui est ensuite affichée à l'aide de la fonction **echo.**

M KARA SAMB
FORMATEUR-DEVELOPPEUR
TEL : 77543 92 82

Instructions de contrôle

Les instructions de contrôle en PHP permettent de contrôler le flot d'exécution d'un script en fonction de certaines conditions. Voici quelques exemples d'instructions de contrôle en PHP :

Instruction if : L'instruction if permet de vérifier une condition et d'exécuter un bloc de code si la condition est vraie.

```
<?php
$age = 18;

if ($age >= 18) {
  echo "Vous êtes majeur.";
}
?>
```

Instruction if ... else : L'instruction if ... else permet de vérifier une condition et d'exécuter un bloc de code si la condition est vraie, ou un autre bloc de code si la condition est fausse.

```
<?php
$age = 17;

if ($age >= 18) {
  echo "Vous êtes majeur.";
} else {
  echo "Vous êtes mineur.";
}

?>
```

Instruction switch : L'instruction switch permet de vérifier une expression et d'exécuter un bloc de code en fonction de la valeur de l'expression.

```
<?php
$couleur = "rouge";
```

```
switch ($couleur) {
  case "rouge":
    echo "La couleur est rouge.";
    break;
  case "vert":
    echo "La couleur est vert.";
    break;
  case "bleu":
    echo "La couleur est bleu.";
    break;
  default:
    echo "La couleur est inconnue.";
    break;
}
?>
```

Boucle for : La boucle for permet de répéter un bloc de code un nombre prédéterminé de fois.

```
<?php


for ($i = 0; $i < 5; $i++) {
  echo $i . " ";
}


?>
```

Boucle while : La boucle while permet de répéter un bloc de code tant qu'une condition est vraie.

```
<?php
$i = 0;
while ($i < 5) {
  echo $i . " ";
  $i++;
```

```
}
?>
```

Ces sont quelques exemples d'instructions de contrôle en PHP. Il en existe d'autres, telles que la boucle do ... while, les instructions break et continue, etc. Il est important de comprendre comment utiliser les instructions de contrôle pour concevoir des scripts PHP efficaces et fiables.

Exemple simple de traitement de formulaire en PHP :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Formulaire de contact</title>
 </head>
 <body>
  <?php
   if (isset($_POST["submit"])) {
    $nom = $_POST["nom"];
    $email = $_POST["email"];
    $message = $_POST["message"];
    echo "Merci " . $nom . ", votre message a bien été envoyé.";
   } else {
  ?>
  <form action="" method="post">
   <label for="nom">Nom :</label>
   <input type="text" id="nom" name="nom"><br><br>

   <label for="email">Email :</label>
   <input type="email" id="email" name="email"><br><br>

   <label for="message">Message :</label>
```

```
    <textarea id="message" name="message"></textarea><br><br>


    <input type="submit" name="submit" value="Envoyer">
  </form>
  <?php
    }
  ?>
 </body>
</html>
```

Lorsque ce code est exécuté, il affichera un formulaire de contact qui demande le nom, l'email et le message de l'utilisateur. Lorsque l'utilisateur soumet le formulaire, les données sont envoyées au script PHP via la méthode POST. Le script PHP vérifie si le bouton submit a été cliqué en utilisant la fonction isset() et la superglobale $_POST. Si le bouton a été cliqué, le script PHP récupère les données du formulaire en utilisant les clés correspondantes de la superglobale $_POST. Enfin, le script PHP affiche un message de remerciement.

Exemple avec un formulaire avec $_REQUEST

Voici un exemple simple de traitement de formulaire en PHP en utilisant la superglobale $_REQUEST :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Formulaire de contact</title>
 </head>
 <body>
  <?php
   if (isset($_REQUEST["submit"])) {
     $nom = $_REQUEST["nom"];
     $email = $_REQUEST["email"];
     $message = $_REQUEST["message"];
```

```
      echo "Merci " . $nom . ", votre message a bien été envoyé.";
    } else {
  ?>
  <form action="" method="post">
    <label for="nom">Nom :</label>
    <input type="text" id="nom" name="nom"><br><br>
    <label for="email">Email :</label>
    <input type="email" id="email" name="email"><br><br>
    <label for="message">Message :</label>
    <textarea id="message" name="message"></textarea><br><br>
    <input type="submit" name="submit" value="Envoyer">
  </form>
  <?php
    }
  ?>
 </body>
</html>
```

Lorsque ce code est exécuté, il affichera un formulaire de contact qui demande le nom, l'email et le message de l'utilisateur. Lorsque l'utilisateur soumet le formulaire, les données sont envoyées au script PHP via la méthode POST. Le script PHP vérifie si le bouton submit a été cliqué en utilisant la fonction isset() et la superglobale $_REQUEST. Si le bouton a été cliqué, le script PHP récupère les données du formulaire en utilisant les clés correspondantes de la superglobale $_REQUEST. Enfin, le script PHP affiche un message de remerciement.

Intégration de PHP dans une page HTML

Il existe deux façons d'intégrer du code PHP dans une page HTML : en ligne et en utilisant des blocs.

Intégration en ligne :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
```

```
  <title>Page PHP</title>
 </head>
 <body>
  <h1>Bonjour <?php echo "le monde"; ?></h1>
 </body>
</html>
```

Intégration en utilisant des blocs :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Page PHP</title>
 </head>
 <body>
  <h1>
   <?php
    echo "Bonjour le monde";
   ?>
  </h1>
 </body>
</html>
```

Dans les deux exemples, le code PHP s'exécute sur le serveur, génère du code HTML et renvoie le résultat au navigateur client. Dans le premier exemple, le code PHP est intégré en ligne à l'intérieur de la balise <h1>, tandis que dans le second exemple, il est intégré en utilisant des blocs PHP.

Notez que les fichiers contenant du code PHP doivent être enregistrés avec l'extension .php pour que le serveur sache comment les traiter.

Variables scalaires, tableaux

Les variables scalaires dans PHP sont des variables qui peuvent stocker une seule valeur. Il existe quatre types de variables scalaires dans PHP : integer (entier), float (nombre flottant), string (chaîne de caractères) et boolean (booléen).

Exemple de déclaration de variables scalaires :

```
<?php
 $age = 30;
 $price = 19.99;
 $name = "John Doe";
 $is_admin = true;
?>
```

Les tableaux en PHP sont des structures de données permettant de stocker plusieurs valeurs sous un même nom. Il existe deux types de tableaux en PHP : les tableaux indexés et les tableaux associatifs.

Exemple de déclaration de tableaux indexés :

```
<?php
 $fruits = array("banane", "pomme", "orange");
 echo $fruits[1]; // Affiche "pomme"
?>
```

Exemple de déclaration de tableaux associatifs :

```
<?php
 $person = array("nom" => "John Doe", "age" => 30, "ville" => "Paris");
 echo $person["ville"]; // Affiche "Paris"
?>
```

Exemple complet de traitement d'un formulaire avec PHP :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
```

```
  <title>Formulaire PHP</title>
 </head>
 <body>
  <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
      $name = $_REQUEST["nom"];
      $email = $_REQUEST["email"];
      echo "Bonjour $name, votre adresse email est $email.";
    } else {
  ?>
    <form action="" method="post">
      <label for="nom">Nom :</label>
      <input type="text" name="nom" id="nom" required>

      <label for="email">Email :</label>
      <input type="email" name="email" id="email" required>

      <input type="submit" value="Envoyer">
    </form>
  <?php
    }
  ?>
 </body>
</html>
```

Dans ce code, le formulaire est affiché si la méthode HTTP utilisée pour accéder à la page est GET. Si la méthode utilisée est POST, c'est-à-dire si le formulaire a été soumis, les données envoyées sont stockées dans les variables $name et $email en utilisant la superglobale $_REQUEST. Le code affiche ensuite un message de bienvenue en utilisant les valeurs de ces variables.

Fonctions & Portée

Les fonctions sont des blocs de code qui peuvent être réutilisés plusieurs fois dans un programme. Elles permettent de diviser un programme en parties plus petites et plus faciles à comprendre et à maintenir.

Exemple simple de fonction en PHP :

```
<?php
 function saluer($nom) {
   echo "Bonjour $nom";
 }

 saluer("John"); // Affiche "Bonjour John"
?>
```

La portée des variables détermine la visibilité d'une variable dans un programme. Il existe deux types de portée en PHP : la portée globale et la portée locale.

Les variables déclarées en dehors de toutes les fonctions ont une portée globale et peuvent être accédées depuis n'importe où dans le programme.

Les variables déclarées à l'intérieur d'une fonction ont une portée locale et ne peuvent être accédées que depuis la fonction où elles ont été déclarées.

Exemple montrant la différence entre la portée globale et la portée locale :

```
<?php
 $nom = "John"; // Portée globale

 function saluer() {
   $nom = "Jane"; // Portée locale
   echo "Bonjour $nom";
 }
```

```
 saluer(); // Affiche "Bonjour Jane"
 echo $nom; // Affiche "John"
?>
```

Exemple montrant comment utiliser des fonctions et la portée des variables dans le traitement d'un formulaire avec PHP :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Formulaire PHP</title>
 </head>
 <body>
  <?php
   function traiterFormulaire() {
    $nom = $_REQUEST["nom"];
    $email = $_REQUEST["email"];
    echo "Bonjour $nom, votre adresse email est $email.";
   }

   if ($_SERVER["REQUEST_METHOD"] == "POST") {
    traiterFormulaire();
   } else {
  ?>
   <form action="" method="post">
    <label for="nom">Nom :</label>
    <input type="text" name="nom" id="nom" required>

    <label for="email">Email :</label>
    <input type="email" name="email" id="email" required>
```

```
    <input type="submit" value="Envoyer">
  </form>
 <?php
  }
 ?>
</body>
</html>
```

Dans ce code, nous avons défini une fonction traiterFormulaire pour traiter les données envoyées par le formulaire. Cette fonction stocke les données envoyées dans les variables locales $nom et $email et affiche un message de bienvenue en utilisant ces variables. Si la méthode HTTP utilisée pour accéder à la page est POST, c'est-à-dire si le formulaire a été soumis, nous appelons la fonction traiterFormulaire pour traiter les données. Si la méthode est GET, nous affichons le formulaire.

Librairies & Inclusion (SSI)

Les bibliothèques sont des collections de fonctions, de classes et de définitions de constantes qui peuvent être incluses dans un programme pour étendre ses fonctionnalités. En PHP, les bibliothèques sont généralement stockées dans des fichiers séparés et inclues dans un programme en utilisant la fonction include ou require.

Exemple d'inclusion de bibliothèque en PHP :

```
// fichier bibliotheque.php
<?php
 function somme($a, $b) {
  return $a + $b;
 }
?>
```

```
// fichier programme.php
<?php
 require "bibliotheque.php";

 echo somme(2, 3); // Affiche 5
```

```
?>
```

L'inclusion de serveur (Server Side Include, SSI) est un mécanisme qui permet d'inclure du contenu HTML dans une page HTML côté serveur avant que la page ne soit envoyée au client. Cela peut être utile pour inclure du contenu répétitif sur plusieurs pages, tels que les en-têtes et les pieds de page.

Pour utiliser les SSI en PHP, vous pouvez inclure du contenu en utilisant la directive include ou require :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Page avec SSI</title>
 </head>
 <body>
  <header>
   <?php include "header.html"; ?>
  </header>

  <main>
   <p>Contenu de la page</p>
  </main>

  <footer>
   <?php include "footer.html"; ?>
  </footer>
 </body>
</html>
```

Dans ce code, nous incluons le contenu des fichiers header.html et footer.html dans les sections header et footer de la page en utilisant la directive include.

Exemple d'utilisation des SSI en PHP :

```
<!-- header.html -->
<header>
  <nav>
    <ul>
      <li><a href="#">Accueil</a></li>
      <li><a href="#">À propos</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>
</header>

<!-- footer.html -->
<footer>
  <p>Copyright © 2023 Mon site</p>
</footer>

<!-- index.php -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Page d'accueil</title>
  </head>
  <body>
    <!-- Inclusion du header -->
    <?php include "header.html"; ?>

    <!-- Contenu principal de la page -->
```

```
  <main>
    <h1>Bienvenue sur mon site !</h1>
    <p>Ceci est la page d'accueil de mon site.</p>
  </main>

  <!-- Inclusion du footer -->
  <?php include "footer.html"; ?>
 </body>
</html>
```

Dans ce code, nous avons défini le header et le footer dans les fichiers header.html et footer.html respectivement. Dans le fichier index.php, nous incluons ces deux fichiers en utilisant la directive include. Ainsi, le header et le footer sont inclus dans la page index.php et le code est réutilisable sur d'autres pages du site.

Fonctions principales, variables serveur et variable PHP

Voici les principales fonctions en PHP :

strlen() : retourne la longueur d'une chaîne de caractères

strpos() : retourne la position d'une sous-chaîne dans une chaîne de caractères

str_replace() : remplace toutes les occurrences d'une chaîne de caractères par une autre

array_keys() : retourne toutes les clés d'un tableau associatif

array_values() : retourne toutes les valeurs d'un tableau associatif

count() : retourne le nombre d'éléments d'un tableau

Voici les principales variables PHP du serveur :

$_SERVER : contient des informations sur le serveur et l'environnement d'exécution

$_GET : contient les données envoyées à la page via la méthode GET

$_POST : contient les données envoyées à la page via la méthode POST

$_FILES : contient les informations sur les fichiers téléchargés via un formulaire

$_COOKIE : contient les données stockées sur le navigateur sous forme de cookies

$_SESSION : contient les données associées à la session en cours

Voici les principales variables PHP internes :

$php_errormsg : contient le message d'erreur généré par la dernière fonction PHP

$HTTP_RAW_POST_DATA : contient les données brutes envoyées avec une requête POST

$argc : contient le nombre d'arguments passés à un script

$argv : contient un tableau des arguments passés à un script.

Exemple d'utilisation de la fonction strlen() :

```
$chaine = "Bonjour";
$longueur = strlen($chaine);
echo "La chaîne '$chaine' contient $longueur caractères.";
Sortie : La chaîne 'Bonjour' contient 7 caractères.
```

Exemple d'utilisation de la variable $_SERVER :

```
echo "Nom du serveur : ".$_SERVER['SERVER_NAME'];
echo "Adresse IP du serveur : ".$_SERVER['SERVER_ADDR'];
echo "Script en cours d'exécution : ".$_SERVER['PHP_SELF'];
Sortie :
```

Nom du serveur : example.com

Adresse IP du serveur : 192.168.1.1

Script en cours d'exécution : /index.php

Exemple de traitement d'un formulaire en PHP :

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

```
 <br>
 <input type="submit" value="Envoyer">
</form>
Traitement des données du formulaire dans le script traitement.php :
php
Copy code
<?php
$nom = $_POST['nom'];
$prenom = $_POST['prenom'];


echo "Bonjour $prenom $nom !";
?>
```

Lorsque l'utilisateur soumet le formulaire en appuyant sur le bouton "Envoyer", les données sont envoyées à traitement.php via la méthode POST et les valeurs de $nom et $prenom sont extraites des données envoyées.

La sortie du script sera alors "Bonjour [prenom] [nom] !".

Contrôles de flux et boucles

Les contrôles de flux permettent de contrôler le flot d'exécution d'un script en fonction de certaines conditions. Les principaux contrôles de flux en PHP sont if, else et switch.

Exemple d'utilisation de la structure if :

```
$age = 20;


if ($age >= 18) {
 echo "Vous êtes majeur.";
} else {
 echo "Vous êtes mineur.";
}
Sortie : Vous êtes majeur.
```

Les boucles permettent de répéter un bloc de code plusieurs fois. Les principales boucles en PHP sont for, while et do while.

Exemple d'utilisation de la boucle for :

```
for ($i = 1; $i <= 10; $i++) {
  echo $i . " ";
}
```

Sortie : 1 2 3 4 5 6 7 8 9 10

Travaux Pratiques

Création d'une Fiche Client

Réalisation de fonctions personnalisées.(TTC, frais de port...etc.)

Voici un exemple de création d'une fiche client en PHP :

Création du formulaire HTML pour saisir les informations du client :

```
<form action="fiche_client.php" method="post">
  <label for="nom">Nom :</label>
  <input type="text" id="nom" name="nom">
  <br>
  <label for="prenom">Prénom :</label>
  <input type="text" id="prenom" name="prenom">
  <br>
  <label for="adresse">Adresse :</label>
  <input type="text" id="adresse" name="adresse">
  <br>
  <label for="ville">Ville :</label>
  <input type="text" id="ville" name="ville">
  <br>
  <input type="submit" value="Enregistrer">
</form>
```

Traitement des données du formulaire dans le script fiche_client.php :

```
php
Copy code
<?php
$nom = $_POST['nom'];
$prenom = $_POST['prenom'];
$adresse = $_POST['adresse'];
$ville = $_POST['ville'];

echo "Fiche client : <br>";
echo "Nom : $nom <br>";
echo "Prénom : $prenom <br>";
echo "Adresse : $adresse <br>";
echo "Ville : $ville <br>";
?>
```

Vous pouvez ensuite définir des fonctions personnalisées pour calculer le montant total TTC (toutes taxes comprises) d'un panier d'achats, ou les frais de port, par exemple.

Exemple de fonction pour calculer le montant total TTC :

```
function calculer_ttc($montant_ht, $taux_tva) {
  return $montant_ht + ($montant_ht * $taux_tva / 100);
}

$montant_ht = 100;
$taux_tva = 20;
$montant_ttc = calculer_ttc($montant_ht, $taux_tva);

echo "Montant HT : $montant_ht € <br>";
echo "Taux TVA : $taux_tva % <br>";
echo "Montant TTC : $montant_ttc € <br>";
```

Sortie :

Montant HT : 100 €

Taux TVA : 20 %

Montant TTC : 120 €

L'approche MVC (modèle vue contrôleur)

L'approche MVC (Modèle-Vue-Contrôleur) est un patron de conception utilisé pour développer des applications web. Il sépare les données de l'application (Modèle), l'interface utilisateur (Vue), et la logique de contrôle (Contrôleur) en trois parties distinctes.

Le modèle représente les données de l'application et les algorithmes qui les gèrent. Il est généralement implémenté en utilisant une base de données et des classes PHP qui représentent les objets de l'application.

La vue est responsable de l'affichage de l'interface utilisateur. Elle reçoit les données du modèle et les affiche à l'utilisateur.

Le contrôleur est un élément central de l'architecture MVC qui assure la coordination entre le modèle et la vue. Il reçoit les entrées de l'utilisateur depuis la vue, les traite et décide de l'action à entreprendre. Il peut également demander des données au modèle et les envoyer à la vue pour affichage.

Ainsi, l'approche MVC permet une meilleure organisation du code et une séparation claire des responsabilités dans le développement d'une application web, ce qui facilite la maintenance et le développement futur de l'application.

Les formulaires simples

Les formulaires simples sont des formulaires web qui permettent à un utilisateur de saisir des données et de les envoyer à un serveur pour traitement. Ils sont généralement créés à l'aide d'HTML et peuvent être soumis à un serveur web à l'aide de PHP pour traitement ultérieur.

Exemple simple de formulaire HTML :

```
<form action="traitement_formulaire.php" method="post">
  <label for="nom">Nom :</label>
  <input type="text" id="nom" name="nom"><br><br>
  <label for="email">Email :</label>
  <input type="email" id="email" name="email"><br><br>
```

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

Lorsque ce formulaire est soumis, les données sont envoyées à "traitement_formulaire.php" pour traitement. Le fichier PHP peut alors accéder aux données envoyées à l'aide de variables telles que $_POST['nom'] et $_POST['email'].

Il est important de noter que les formulaires simples sont soumis en utilisant la méthode HTTP POST, ce qui signifie que les données sont envoyées au serveur en arrière-plan et ne sont pas visible dans l'URL de la page.

Les verbes http

Les verbes HTTP sont les méthodes utilisées pour envoyer des requêtes à un serveur web. Les verbes HTTP les plus couramment utilisés sont :

GET : utilisé pour récupérer des informations du serveur sans modification.

POST : utilisé pour envoyer des données au serveur pour traitement.

PUT : utilisé pour mettre à jour les données existantes sur le serveur.

DELETE : utilisé pour supprimer des données du serveur.

HEAD : similaire à GET, mais renvoie uniquement les en-têtes de réponse sans le corps de la réponse.

OPTIONS : utilisé pour déterminer les options de communication disponibles pour une ressource particulière.

Lors de la création de formulaires web, il est important de choisir le verbe HTTP approprié en fonction de l'opération souhaitée (lecture, modification, suppression, etc.). Par exemple, pour un formulaire de saisie de données, la méthode POST sera généralement utilisée, tandis que pour un formulaire de suppression de données, la méthode DELETE sera utilisée.

Exemple d'utilisation des verbes HTTP avec PHP :

```
GET :
<?php
// Récupération des informations de la page actuelle
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
 $name = $_GET['name'];
 echo "Bonjour, " . $name;
```

```
}
?>

<form action="index.php" method="get">
  <input type="text" name="name">
  <button type="submit">Envoyer</button>
</form>
```

```
POST :
<?php
// Envoi des données pour traitement
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  $name = $_POST['name'];
  echo "Bonjour, " . $name;
}
?>

<form action="index.php" method="post">
  <input type="text" name="name">
  <button type="submit">Envoyer</button>
</form>
```

```
PUT :
<?php
// Mise à jour des données sur le serveur
if ($_SERVER['REQUEST_METHOD'] == 'PUT') {
  parse_str(file_get_contents("php://input"), $data);
  $name = $data['name'];
  echo "Bonjour, " . $name;
```

```
}
?>


<form action="index.php" method="post">
  <input type="hidden" name="_method" value="put">
  <input type="text" name="name">
  <button type="submit">Envoyer</button>
</form>
```

DELETE :

```
<?php
// Suppression des données du serveur
if ($_SERVER['REQUEST_METHOD'] == 'DELETE') {
  parse_str(file_get_contents("php://input"), $data);
  $name = $data['name'];
  echo "Bonjour, " . $name . " a été supprimé.";
}
?>


<form action="index.php" method="post">
  <input type="hidden" name="_method" value="delete">
  <input type="text" name="name">
  <button type="submit">Envoyer</button>
</form>
```

Notez que les verbes PUT et DELETE ne peuvent pas être utilisés directement dans un formulaire HTML standard, il est donc nécessaire de les masquer en utilisant une méthode telle que celle indiquée ci-dessus.

Les variables Système de PHP

Les variables système de PHP sont des variables prédéfinies qui fournissent des informations sur le système d'exploitation, les paramètres du serveur et les données en provenance de l'utilisateur. Parmi les variables système les plus couramment utilisées, on peut citer :

$_SERVER : contient des informations sur le serveur et les en-têtes HTTP

$_GET : contient des informations sur les variables envoyées via l'URL

$_POST : contient des informations sur les variables envoyées via un formulaire avec la méthode POST

$_REQUEST : contient des informations sur les variables envoyées via l'URL ou un formulaire

$_COOKIE : contient des informations sur les cookies envoyés par le navigateur.

Exemple d'utilisation de la variable $_SERVER :

```
<?php
  echo "Le nom de votre serveur est : " . $_SERVER['SERVER_NAME'] . "<br>";
  echo "Vous utilisez le protocole : " . $_SERVER['SERVER_PROTOCOL'] . "<br>";
  echo "Le script en cours d'exécution est : " . $_SERVER['PHP_SELF'] . "<br>";
  echo "L'adresse IP de votre ordinateur est : " . $_SERVER['REMOTE_ADDR'];
?>
```

Ce code affichera les informations suivantes :

Le nom de votre serveur

Le protocole utilisé (par exemple, HTTP/1.1)

Le script en cours d'exécution

L'adresse IP de l'ordinateur qui a demandé la page.

Analyse des principaux types de champ

Il existe plusieurs types de champs qui peuvent être utilisés dans un formulaire HTML :

Texte : Ce champ permet à l'utilisateur d'entrer du texte libre, généralement limité à une certaine longueur.

Zone de texte : Ce champ permet à l'utilisateur d'entrer du texte sur plusieurs lignes.

Bouton radio : Ce champ permet à l'utilisateur de sélectionner une seule option parmi plusieurs.

Cases à cocher : Ce champ permet à l'utilisateur de sélectionner plusieurs options en même temps.

Liste déroulante : Ce champ permet à l'utilisateur de sélectionner une option parmi plusieurs à partir d'une liste déroulante.

Bouton de soumission : Ce champ permet à l'utilisateur de soumettre les données du formulaire.

Champ caché : Ce champ n'apparaît pas à l'utilisateur, mais il est utilisé pour transmettre des informations supplémentaires au serveur.

Fichier : Ce champ permet à l'utilisateur de télécharger un fichier sur le serveur.

Chacun de ces champs peut être personnalisé à l'aide d'attributs tels que la longueur maximum, les valeurs par défaut et les options de sélection. Il est important de valider les données entrées par l'utilisateur pour s'assurer de leur validité et de la sécurité de l'application.

Exemple de formulaire HTML avec différents types de champs :

```
<form action="submit.php" method="post">
 <label for="name">Nom :</label>
 <input type="text" id="name" name="name" required>
 <br><br>

 <label for="email">Adresse e-mail :</label>
 <input type="email" id="email" name="email" required>
 <br><br>

 <label for="gender">Genre :</label>
 <input type="radio" id="male" name="gender" value="male">
 <label for="male">Masculin</label>
 <input type="radio" id="female" name="gender" value="female">
 <label for="female">Féminin</label>
 <br><br>
```

```
 <label for="fruits">Fruits préférés :</label>
 <input type="checkbox" id="apple" name="fruits[]" value="apple">
 <label for="apple">Pomme</label>
 <input type="checkbox" id="banana" name="fruits[]" value="banana">
 <label for="banana">Banane</label>
 <br><br>

 <label for="country">Pays :</label>
 <select id="country" name="country">
  <option value="">Sélectionnez un pays</option>
  <option value="france">France</option>
  <option value="germany">Allemagne</option>
  <option value="italy">Italie</option>
 </select>
 <br><br>
 <input type="submit" value="Envoyer">
</form>
```

Ce formulaire comprend des champs de type texte, radio, cases à cocher, liste déroulante et bouton de soumission. Il peut être soumis à un script PHP pour traitement et validation des données entrées.

Travaux Pratiques

Réalisation d'une page de recherche : l'utilisateur peut définir plusieurs critères et les transmettre au moteur de recherche

Voici un exemple de page HTML avec un formulaire de recherche qui peut être soumis à un script PHP pour traitement et exécution de la recherche :

```
<form action="search.php" method="post">
 <label for="keywords">Mots-clés :</label>
 <input type="text" id="keywords" name="keywords" required>
 <br><br>
 <label for="category">Catégorie :</label>
 <select id="category" name="category">
```

```
  <option value="">Sélectionnez une catégorie</option>
  <option value="books">Livres</option>
  <option value="music">Musique</option>
  <option value="movies">Films</option>
 </select>
 <br><br>
 <label for="price">Prix :</label>
 <input type="text" id="price" name="price">
 <br><br>
 <input type="submit" value="Rechercher">
</form>
```

Lorsque l'utilisateur soumet le formulaire, les données sont envoyées au script PHP search.php qui peut les utiliser pour effectuer une requête de recherche en utilisant une base de données ou une API externe. Les résultats peuvent être affichés sur une page HTML distincte.

Exemple de script search.php

Exemple de script PHP qui peut être utilisé pour traiter les données soumises à partir du formulaire de recherche :

```
<?php
// Récupération des données soumises via le formulaire
$keywords = $_POST['keywords'];
$category = $_POST['category'];
$price = $_POST['price'];

// Validation des données
if (!empty($keywords)) {
 // Traitement de la recherche
 // ...

 // Récupération des résultats
 // ...
```

```
 // Affichage des résultats
 echo '<h2>Résultats de la recherche</h2>';
 echo '<ul>';
 foreach ($results as $result) {
   echo '<li>' . $result . '</li>';
 }
 echo '</ul>';
} else {
 echo '<p>Aucun mot-clé de recherche n\'a été saisi.</p>';
}
?>
```

Ce script PHP vérifie d'abord si les mots-clés de recherche ont été soumis. Si c'est le cas, il peut effectuer une requête de recherche en utilisant les données soumises et afficher les résultats. Si les mots-clés de recherche n'ont pas été soumis, un message est affiché indiquant qu'aucun mot-clé de recherche n'a été saisi.

Analyse des principaux types de champ

Il existe plusieurs types de champs qui peuvent être utilisés dans les formulaires web pour soumettre des données :

Champs de texte : Ce sont les champs les plus courants et permettent à l'utilisateur de saisir du texte.

Champs de sélection : Ce sont des menus déroulants qui permettent à l'utilisateur de choisir une option parmi plusieurs.

Cases à cocher : Ce sont des cases qui permettent à l'utilisateur de sélectionner plusieurs options.

Boutons radio : Ce sont des boutons qui permettent à l'utilisateur de sélectionner une seule option parmi plusieurs.

Champs cachés : Ce sont des champs qui ne sont pas visibles pour l'utilisateur, mais qui peuvent transmettre des données supplémentaires au serveur.

Champs de fichiers : Ce sont des champs qui permettent à l'utilisateur de charger des fichiers sur le serveur.

Champs de zone de texte : Ce sont des champs qui permettent à l'utilisateur de saisir plusieurs lignes de texte.

Il est important de choisir le type de champ approprié en fonction des données que vous souhaitez soumettre et des types d'options que vous souhaitez offrir à l'utilisateur.

Exemple de formulaire HTML qui utilise plusieurs types de champs :

```
<form action="traitement.php" method="post">
 <label for="nom">Nom :</label>
 <input type="text" id="nom" name="nom"><br><br>
 <label for="sexe">Sexe :</label>
 <input type="radio" id="homme" name="sexe" value="homme">
 <label for="homme">Homme</label>
 <input type="radio" id="femme" name="sexe" value="femme">
 <label for="femme">Femme</label><br><br>

 <label for="ville">Ville :</label>
 <select id="ville" name="ville">
  <option value="paris">Paris</option>
  <option value="lyon">Lyon</option>
  <option value="marseille">Marseille</option>
 </select><br><br>

 <label for="interets">Intérêts :</label>
 <input type="checkbox" id="sport" name="interets[]" value="sport">
 <label for="sport">Sport</label>
 <input type="checkbox" id="cinema" name="interets[]" value="cinema">
 <label for="cinema">Cinéma</label>
 <input type="checkbox" id="musique" name="interets[]" value="musique">
 <label for="musique">Musique</label><br><br>
 <label for="message">Message :</label>
 <textarea id="message" name="message"></textarea><br><br>
```

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

Ce formulaire utilise un champ de texte pour saisir le nom, des boutons radio pour sélectionner le sexe, un menu déroulant pour choisir la ville, des cases à cocher pour sélectionner les intérêts, une zone de texte pour saisir un message et un bouton de soumission pour envoyer les données au serveur.

Travaux Pratiques

Réalisation d'une page de recherche : l'utilisateur peut définir plusieurs critères et les transmettre au moteur de recherche

Pour réaliser une page de recherche avec plusieurs critères, il est nécessaire de créer un formulaire HTML qui permettra à l'utilisateur de définir ses critères de recherche. Ce formulaire doit être envoyé à un script PHP, appelé "moteur de recherche", qui traitera les données soumises et retournera les résultats correspondants.

Exemple de code pour un formulaire HTML de recherche :

```
<form action="search.php" method="post">
 <label for="keyword">Mot-clé :</label>
 <input type="text" name="keyword" id="keyword" required>

 <label for="category">Catégorie :</label>
 <select name="category" id="category">
  <option value="">Toutes les catégories</option>
  <option value="1">Catégorie 1</option>
  <option value="2">Catégorie 2</option>
  <option value="3">Catégorie 3</option>
 </select>

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

Exemple de code pour le script PHP "search.php" :

```
<?php
```

```
if (isset($_POST['keyword'])) {
  // Récupération des critères de recherche
  $keyword = $_POST['keyword'];
  $category = $_POST['category'];

  // Traitement des données et recherche des résultats
  $results = search($keyword, $category);

  // Affichage des résultats
  echo "<h2>Résultats de la recherche</h2>";
  echo "<ul>";
  foreach ($results as $result) {
    echo "<li>" . $result . "</li>";
  }
  echo "</ul>";
}

function search($keyword, $category) {
  // Implémentation de la logique de recherche (à adapter selon les besoins)
  return array("Résultat 1", "Résultat 2", "Résultat 3");
}
?>
```

Cet exemple montre comment créer un formulaire HTML pour définir les critères de recherche et comment traiter les données soumises à l'aide d'un script PHP. La fonction search() permet de traiter les données et de retourner les résultats correspondants.

Fichiers texte et binaires

En PHP, il est possible de manipuler des fichiers textes et binaires. Les fichiers textes peuvent être lus et modifiés en utilisant les fonctions PHP telles que fopen, fread, fwrite, fclose, etc. Les fichiers binaires, quant à eux, peuvent être manipulés en utilisant les fonctions de gestion de fichiers binaires telles que file_get_contents, file_put_contents, etc. Il est également possible d'uploader des fichiers sur le serveur à l'aide de formulaires HTML et de les traiter avec PHP pour les enregistrer sur le disque dur du serveur.

M KARA SAMB
FORMATEUR-DEVELOPPEUR
TEL : 77543 92 82

Importance des fichiers dans un développement moderne

Les fichiers jouent un rôle important dans le développement moderne de plusieurs manières:

Stockage de données: Les fichiers sont souvent utilisés pour stocker des données sur le serveur, telles que les informations sur les utilisateurs, les produits, etc.

Upload et partage de fichiers: Les fichiers peuvent être uploadés par les utilisateurs sur le serveur et partagés entre les différentes parties intéressées.

Logs: Les fichiers peuvent être utilisés pour enregistrer les activités sur le serveur, les erreurs, les performances, etc. pour une analyse ultérieure.

Sauvegarde et restauration de données: Les fichiers peuvent être utilisés pour sauvegarder des données importantes et les restaurer en cas de perte ou de défaillance du système.

Intégration avec d'autres systèmes: Les fichiers peuvent être importés ou exportés pour intégrer les données avec d'autres systèmes ou applications.

Exemple simple d'utilisation des fichiers dans un développement moderne avec PHP :

Supposons que nous voulions enregistrer les données des utilisateurs dans un fichier texte sur le serveur. Nous pouvons utiliser la fonction file_put_contents() de PHP pour écrire les données dans un fichier. Par exemple :

```
<?php
$data = "John Doe, 32, New York\nJane Doe, 30, Los Angeles";
file_put_contents('users.txt', $data, FILE_APPEND);
?>
```

Ce code écrira les données dans un fichier nommé "users.txt". La constante FILE_APPEND garantit que les données seront ajoutées à la fin du fichier, sans remplacer les données existantes.

De la même manière, nous pouvons utiliser la fonction file_get_contents() pour lire les données d'un fichier :

```
<?php
$data = file_get_contents('users.txt');
echo $data;
?>
```

Ce code affichera les données du fichier "users.txt" à l'écran.

Lecture écriture de fichier

L'ouverture et la lecture de fichiers en PHP peuvent être accomplies avec les fonctions fopen(), fread(), fgets() et file(). Pour écrire dans un fichier, vous pouvez utiliser les fonctions fwrite() et file_put_contents().

Exemple de lecture de fichier en PHP :

```
$file = fopen("file.txt", "r") or die("Unable to open file!");
while(!feof($file)) {
  echo fgets($file) . "<br>";
}
fclose($file);
```

Et un exemple d'écriture dans un fichier en PHP :

```
$file = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($file, $txt);
$txt = "Jane Doe\n";
fwrite($file, $txt);
fclose($file);
```

Vérification de login/mot de passe

La vérification de login/mot de passe peut être effectuée en comparant les informations entrées par l'utilisateur avec celles stockées dans une base de données ou un fichier. Il est important de vérifier les informations de manière sécurisée en utilisant des fonctions de hashage pour hasher le mot de passe et en utilisant une méthode sécurisée pour stocker les informations dans la base de données ou le fichier. Voici un exemple de code PHP pour vérifier le login/mot de passe :

```
if(isset($_POST['submit'])) {
  $username = $_POST['username'];
  $password = $_POST['password'];
```

```
 // Vérifier les informations en utilisant une requête SQL ou en lisant un fichier
 // ...

 if($stored_username == $username && $stored_password == hash('sha256', $password))
{
   // Login réussi
   // ...
 } else {
   // Login échoué
   // ...
 }
}
```

Cet exemple montre comment vérifier le login/mot de passe en utilisant une hashage sha256 pour sécuriser le mot de passe. Il est important de ne jamais stocker les mots de passe en clair dans une base de données ou un fichier.

Headers http & Redirection

Les en-têtes HTTP sont des informations supplémentaires qui peuvent être envoyées avec une requête ou une réponse HTTP. Ils peuvent contenir des informations telles que la date et l'heure de la requête, le type de contenu renvoyé, la localisation de la ressource demandée, etc. Les en-têtes HTTP sont importants car ils peuvent affecter le comportement de la requête ou de la réponse.

La redirection est un mécanisme qui permet de renvoyer un utilisateur ou un navigateur vers une autre URL. Cela peut être nécessaire pour plusieurs raisons, telles que l'existence d'une nouvelle URL pour une ressource ou la nécessité de restreindre l'accès à une ressource. La redirection peut être implémentée en utilisant des codes de réponse HTTP tels que "301 Moved Permanently" ou "302 Found".

Exemple de l'utilisation des en-têtes HTTP et de la redirection en PHP :

```
<?php
// Vérifier si l'utilisateur est connecté
if (!isset($_SESSION['user_id'])) {
 // Si l'utilisateur n'est pas connecté, rediriger vers la page de connexion
 header("Location: login.php");
```

```
 exit;
}

// Si l'utilisateur est connecté, afficher le contenu protégé
echo "Bienvenue sur la page protégée.";
?>
```

Dans cet exemple, si la session user_id n'a pas été définie, c'est-à-dire que l'utilisateur n'est pas connecté, l'en-tête Location est envoyé avec une valeur de login.php pour rediriger l'utilisateur vers la page de connexion. Sinon, si l'utilisateur est connecté, le message "Bienvenue sur la page protégée" est affiché.

Travaux Pratiques

Analyse d'un fichier flat-file pour génération de listes déroulantes.

Première mise en œuvre de la page catalogue, sur base des données extraites du fichier

Skipped 1 messages

Travaux Pratiques

Analyse d'un fichier flat-file pour génération de listes déroulantes.

Pour réaliser ce travail pratique, nous allons d'abord analyser le fichier flat-file pour comprendre son contenu et son format. Ce fichier peut être un fichier CSV (Comma Separated Values) ou un fichier texte simple avec une structure définie.

Ensuite, nous pouvons utiliser PHP pour lire le contenu du fichier et le stocker dans une variable, souvent un tableau. Nous pouvons parcourir ce tableau pour générer les options de la liste déroulante.

Exemple de code qui lit un fichier CSV et génère une liste déroulante à partir de ses données:

```
<select name="product_list">
 <?php
 $file = fopen("products.csv", "r");
 while (($data = fgetcsv($file, 1000, ",")) !== FALSE) {
  echo "<option value='".$data[0]."'>".$data[1]."</option>";
 }
 fclose($file);
```

```
 ?>

</select>
```

Ce code ouvre le fichier products.csv en mode lecture, parcourt les lignes du fichier avec la fonction fgetcsv et génère une option pour chaque ligne. La valeur de l'option est définie sur la première colonne de chaque ligne et le texte affiché est défini sur la seconde colonne.

Enfin, nous pouvons inclure ce code dans notre page HTML pour afficher la liste déroulante sur notre page catalogue.

Première mise en œuvre de la page catalogue, sur base des données extraites du fichier

Pour mettre en œuvre la page catalogue, vous pouvez utiliser les données extraites d'un fichier plat (flat-file) en utilisant les fonctions de lecture de fichier de PHP. Il est important de vérifier que les données sont sécurisées et protégées contre les injections de code malveillant.

Une fois que les données sont lues, vous pouvez les utiliser pour générer du HTML et afficher les produits du catalogue. Il est important d'utiliser des boucles pour parcourir les données du catalogue et générer du HTML pour chaque produit.

Enfin, vous pouvez utiliser des fonctions telles que htmlspecialchars pour sécuriser les données affichées dans la page catalogue et éviter les problèmes de sécurité tels que la faille XSS.

Exemple de code pour générer la page catalogue à partir des données extraites d'un fichier plat peut ressembler à ceci :

```
<?php


// Ouvrir le fichier en lecture
$file = fopen("catalogue.txt", "r");


// Lire les données du fichier et les stocker dans un tableau
$catalogue = array();
while (!feof($file)) {
  $ligne = fgets($file);
  $produit = explode(",", $ligne);
```

```
  $catalogue[] = $produit;
}

// Fermer le fichier
fclose($file);

// Afficher le catalogue en utilisant une boucle
echo "<table>";
foreach ($catalogue as $produit) {
  echo "<tr>";
  echo "<td>" . htmlspecialchars($produit[0]) . "</td>";
  echo "<td>" . htmlspecialchars($produit[1]) . "</td>";
  echo "<td>" . htmlspecialchars($produit[2]) . "</td>";
  echo "</tr>";
}
echo "</table>";

?>
```

Notez que ce n'est qu'un exemple de base et que vous pouvez ajouter des fonctionnalités supplémentaires en fonction de vos besoins.

Sessions utilisateurs et panier d'achat

Les sessions permettent de stocker des informations sur le serveur pour une durée limitée pour un utilisateur spécifique, tout en les associant à un identifiant unique. Cela permet de conserver l'état d'une application entre les requêtes d'un même utilisateur.

Dans le cas d'un panier d'achat, les sessions peuvent être utilisées pour stocker les articles ajoutés au panier par un utilisateur, et pour maintenir le contenu du panier en cours de navigation sur le site, même après plusieurs requêtes.

Exemple de code pour ajouter un article au panier serait:

```
<?php
```

```
session_start();


if (isset($_GET['add_to_cart'])) {
   $product_id = $_GET['add_to_cart'];


   if (!isset($_SESSION['cart'])) {
      $_SESSION['cart'] = array();
   }


   array_push($_SESSION['cart'], $product_id);
}


// Afficher le contenu du panier
print_r($_SESSION['cart']);
?>
```

Exemple simple de formulaire d'achat avec PHP et sessions utilisateurs :

```
<?php
// Démarrage de la session
session_start();


// Vérification de la présence de l'article dans le panier
if(isset($_POST['ajouter_au_panier'])) {
   // Si le panier n'existe pas encore, on le crée
   if(!isset($_SESSION['panier'])) {
      $_SESSION['panier'] = array();
   }
   // Ajout de l'article au panier
   array_push($_SESSION['panier'], $_POST['article']);
}
```

```
// Affiche le contenu du panier
echo "Contenu du panier : <br>";
print_r($_SESSION['panier']);

?>

<form action="" method="post">
 <label for="article">Article :</label>
 <input type="text" id="article" name="article">
 <input type="submit" value="Ajouter au panier" name="ajouter_au_panier">
</form>
```

Dans ce code, on utilise la fonction session_start() pour démarrer une session PHP. Puis, on vérifie si le formulaire d'ajout au panier a été soumis en utilisant la superglobale $_POST. Si le formulaire a été soumis, on vérifie si la session panier existe déjà, et si ce n'est pas le cas, on la crée en utilisant un tableau vide. Ensuite, on ajoute l'article entré par l'utilisateur à la fin du tableau en utilisant la fonction array_push(). Enfin, on affiche le contenu du panier en utilisant la fonction print_r().

Avantages et inconvénients des cookies et sessions

Les avantages des cookies sont :

Les cookies peuvent enregistrer des informations sur les préférences de l'utilisateur pour les sites web, ce qui peut améliorer l'expérience de l'utilisateur.

Les cookies peuvent être utilisés pour suivre le comportement des utilisateurs sur un site web et collecter des données d'analyse.

Les inconvénients des cookies sont :

Les cookies peuvent être utilisés pour suivre les activités des utilisateurs sur Internet, ce qui peut être considéré comme une violation de la vie privée.

Certains utilisateurs peuvent bloquer les cookies sur leur navigateur, ce qui peut rendre les fonctionnalités de certains sites web inopérantes.

Les avantages des sessions sont :

Les sessions permettent de stocker les données sur le serveur plutôt que sur l'ordinateur de l'utilisateur, ce qui est plus sécurisé.

Les sessions peuvent gérer l'identification de l'utilisateur sur un site web.

Les inconvénients des sessions sont :

Les sessions dépendent du serveur et peuvent ne pas être disponibles si le serveur tombe en panne.

Les sessions peuvent être plus difficiles à gérer que les cookies, car elles nécessitent une gestion active du serveur.

Exemple de cookies et de session à l'aide de formulaire

```
<?php

// Vérification de la soumission du formulaire
if(isset($_POST['submit'])) {
   // Récupération des données du formulaire
   $username = $_POST['username'];
   $password = $_POST['password'];

   // Vérification du nom d'utilisateur et du mot de passe
   if($username == "admin" && $password == "secret") {
      // Définition d'un cookie pour enregistrer le nom d'utilisateur
      setcookie("username", $username, time()+3600);

      // Démarrage d'une session pour enregistrer les informations de l'utilisateur
      session_start();
      $_SESSION['username'] = $username;

      // Redirection vers la page protégée
      header("Location: protected_page.php");
      exit;
   } else {
      // Affichage d'un message d'erreur
      echo "Nom d'utilisateur ou mot de passe incorrect.";
```

```
    }
}


?>


<!-- Formulaire pour demander le nom d'utilisateur et le mot de passe -->
<form action="" method="post">
    <label for="username">Nom d'utilisateur :</label>
    <input type="text" id="username" name="username" required>

    <label for="password">Mot de passe :</label>
    <input type="password" id="password" name="password" required>

    <input type="submit" name="submit" value="Se connecter">
</form>
```

Dans cet exemple, lorsque l'utilisateur soumet le formulaire, les données du nom d'utilisateur et du mot de passe sont vérifiées. Si les informations sont correctes, un cookie est défini pour enregistrer le nom d'utilisateur et une session est démarrée pour enregistrer les informations de l'utilisateur. Enfin, l'utilisateur est redirigé vers la page protégée. Si les informations sont incorrectes, un message d'erreur est affiché.

Limitations et précautions

Les limitations et précautions concernant l'utilisation des cookies et des sessions comprennent:

Durée de vie limitée: Les cookies ont une durée de vie limitée et peuvent être supprimés par l'utilisateur à tout moment. Les sessions ont également une durée de vie limitée qui se termine généralement lorsque l'utilisateur ferme le navigateur.

Sécurité: Les cookies peuvent être volés ou altérés par des tierces parties, ce qui peut entraîner des problèmes de sécurité. Les sessions sont souvent mieux sécurisées car elles sont généralement associées à un identifiant de session crypté qui n'est pas stocké sur le client.

Stockage limité: Les cookies ont une taille de stockage limitée, ce qui peut entraîner des problèmes si vous avez besoin de stocker de grandes quantités de données. Les sessions sont plus adaptées pour stocker de grandes quantités de données, mais elles peuvent également être limitées en fonction de la configuration du serveur.

Il est important de prendre en compte ces limitations et de les considérer en fonction des exigences de votre application avant de choisir d'utiliser des cookies ou des sessions. Il est également important de mettre en œuvre des mesures de sécurité adéquates pour protéger les données stockées dans les cookies ou les sessions.

Les variables de session

Les variables de session sont des variables qui peuvent être stockées sur le serveur pour une durée déterminée et qui sont associées à un utilisateur spécifique. Elles peuvent être utilisées pour stocker des informations sur l'état de l'application, tels que les préférences de l'utilisateur, les données de panier d'achat, etc.

Les variables de session sont accessibles sur toutes les pages du site où elles ont été définies, ce qui en fait un moyen pratique pour les développeurs de suivre les données de l'utilisateur à travers les pages de leur site Web. Elles sont généralement associées à un identifiant de session unique, généré par le serveur, pour garantir que les données de session associées à un utilisateur spécifique ne soient pas mélangées avec les données d'un autre utilisateur.

Pour utiliser des variables de session dans PHP, vous devez d'abord démarrer une session avec la fonction session_start(). Ensuite, vous pouvez définir des variables de session en assignant des valeurs à des variables $_SESSION. Pour accéder à ces variables sur d'autres pages, vous devez d'abord démarrer une session en utilisant session_start() à nouveau.

Exemple de l'utilisation de variables de session avec un formulaire en PHP :

```php
<?php
// Démarrage de la session
session_start();

if (isset($_POST['submit'])) {
  // Stockage des données du formulaire dans des variables de session
  $_SESSION['username'] = $_POST['username'];
  $_SESSION['email'] = $_POST['email'];

  // Redirection vers la page suivante
  header("Location: next_page.php");
```

```
  exit;
}
?>


<!DOCTYPE html>
<html>
<head>
  <title>Formulaire de connexion</title>
</head>
<body>
  <form action="" method="post">
    <input type="text" name="username" placeholder="Nom d'utilisateur">
    <input type="email" name="email" placeholder="Adresse email">
    <input type="submit" name="submit" value="Envoyer">
  </form>
</body>
</html>
```

Dans ce script, nous démarrons d'abord la session avec la fonction session_start(). Ensuite, nous vérifions si le formulaire a été soumis avec le bouton submit. Si c'est le cas, nous stockons les données du formulaire (nom d'utilisateur et adresse email) dans des variables de session. Enfin, nous redirigeons l'utilisateur vers la page suivante avec la fonction header().

Les Cookies

Les cookies sont de petits fichiers de données stockés sur l'ordinateur d'un utilisateur par un site web. Ils peuvent contenir des informations telles que les préférences de l'utilisateur, les informations de connexion ou toute autre information qui peut aider à personnaliser l'expérience de l'utilisateur sur le site. Les cookies peuvent être lus par le site web lorsque l'utilisateur y accède à nouveau, ce qui permet de se souvenir de l'utilisateur et de personnaliser son expérience. Les cookies peuvent être utilisés pour une variété de fins, telles que la mémorisation de la connexion à un compte, la personnalisation de la page d'accueil, la tenue de statistiques de site web et la livraison de publicités ciblées. Cependant, il est important de noter que les cookies peuvent également poser des problèmes de sécurité et de vie privée, il est donc important de les utiliser avec précaution et de les gérer correctement.

Exemple de script PHP permettant de créer, lire et effacer un cookie :

```php
<?php
// Création du cookie
setcookie("username", "John Doe", time() + 3600); // Expire dans 1 heure


// Lecture du cookie
if (isset($_COOKIE["username"])) {
  echo "Bienvenue " . $_COOKIE["username"];
} else {
  echo "Bonjour invité";
}


// Suppression du cookie
setcookie("username", "", time() - 3600);
?>
```

Dans cet exemple, nous créons un cookie "username" avec une durée de vie de 1 heure. Nous vérifions ensuite si ce cookie existe en utilisant la fonction isset, et affichons un message différent en conséquence. Enfin, nous effaçons le cookie en le ré-écrivant avec une date d'expiration antérieure.

Sérialisation des variables complexes

La sérialisation des variables complexes consiste à convertir une variable PHP en une chaîne de caractères pouvant être stockée et récupérée plus tard. Cette conversion peut inclure des tableaux, des objets, des ressources et d'autres types de données PHP. La fonction serialize() peut être utilisée pour sérialiser une variable et la fonction unserialize() peut être utilisée pour dé-sérialiser la chaîne sérialisée et récupérer la variable originale.

```php
<?php
  $person = array("nom" => "Dupont", "prenom" => "Jean", "age" => 30);
  $person_ser = serialize($person);
  setcookie("person", $person_ser);
?>
```

Lorsqu'une page est chargée, le cookie person peut être dé-sérialisé pour récupérer les informations originales de la variable $person:

```php
<?php
 $person = unserialize($_COOKIE["person"]);
 echo "Nom: " . $person["nom"] . "<br>";
 echo "Prénom: " . $person["prenom"] . "<br>";
 echo "Age: " . $person["age"] . "<br>";
?>
```

Exemple d'utilisation de la sérialisation de variables complexes à l'aide d'un formulaire en PHP :

Créez un formulaire HTML avec des champs pour entrer des informations sur un produit, telles que le nom, la description, le prix et la quantité.

Traitez les données soumises par le formulaire en utilisant le code PHP suivant :

```php
if (isset($_POST['submit'])) {
 // Récupérer les données soumises par le formulaire
 $product = array(
  'name' => $_POST['name'],
  'description' => $_POST['description'],
  'price' => $_POST['price'],
  'quantity' => $_POST['quantity']
 );

 // Sérialiser les données du produit en une chaîne de caractères
 $serialized_product = serialize($product);

 // Stocker les données sérialisées dans un cookie
 setcookie('product', $serialized_product, time() + 3600);
}
```

Lorsque l'utilisateur soumet le formulaire, les informations sur le produit sont stockées dans un cookie sur son ordinateur. Vous pouvez accéder à ces données en utilisant le code PHP suivant :

```
if (isset($_COOKIE['product'])) {
 // Déserialiser les données du produit à partir de la chaîne de caractères stockée dans le cookie
 $product = unserialize($_COOKIE['product']);

 // Afficher les informations sur le produit
 echo 'Nom du produit : ' . $product['name'] . '<br>';
 echo 'Description : ' . $product['description'] . '<br>';
 echo 'Prix : ' . $product['price'] . '<br>';
 echo 'Quantité : ' . $product['quantity'] . '<br>';
}
```

Cet exemple montre comment sérialiser les données complexes d'un formulaire en utilisant des cookies en PHP.

Utilisation

L'utilisation de la sérialisation de variables complexes dépend du contexte dans lequel elle est utilisée. En général, elle peut être utilisée pour stocker des données complexes telles que des tableaux associatifs ou des objets en tant que valeur d'un cookie ou d'une variable de session. Cela peut être utile pour conserver des informations de l'utilisateur à travers les requêtes, comme le contenu d'un panier d'achat dans une boutique en ligne. La sérialisation peut également être utilisée pour transmettre des données complexes entre des scripts sur le serveur ou entre le serveur et le client. Il est important de noter que la sérialisation n'est pas sécurisée par défaut et que les données sérialisées doivent être vérifiées avant d'être désérialisées pour éviter les vulnérabilités de sécurité telles que l'injection de code.

Exemple simple d'utilisation de la sérialisation des variables complexes en PHP :

```
<?php
// Déclaration d'un tableau associatif
$person = array("nom" => "Doe", "prenom" => "John", "age" => 30);
// Sérialisation du tableau associatif en une chaîne de caractères
$serialized = serialize($person);

// Stockage de la chaîne de caractères sérialisée dans un cookie
setcookie("person", $serialized);
```

```
// Récupération de la chaîne de caractères sérialisée dans un cookie
$cookie = $_COOKIE["person"];

// Désérialisation de la chaîne de caractères sérialisée en un tableau associatif
$person = unserialize($cookie);

// Affichage des valeurs du tableau associatif désérialisé
echo "Nom : " . $person["nom"] . "<br>";
echo "Prénom : " . $person["prenom"] . "<br>";
echo "Age : " . $person["age"] . "<br>";
?>
```

Dans ce code, on déclare un tableau associatif $person avec des informations sur une personne. On utilise la fonction serialize() pour convertir ce tableau en une chaîne de caractères sérialisée. Ensuite, on stocke cette chaîne sérialisée dans un cookie. Au chargement de la page, on récupère le cookie et on utilise la fonction unserialize() pour le convertir en tableau associatif à nouveau. Finalement, on affiche les valeurs du tableau associatif désérialisé.

Travaux Pratiques

Gestion de l'authentification et des autorisations pour accès au Back-Office.

Pour la gestion de l'authentification et des autorisations pour l'accès au back-office, vous pouvez créer un système de connexion en utilisant des sessions et des cookies. Vous pouvez vérifier les informations d'identification de l'utilisateur à partir d'une base de données ou d'un fichier texte. Si les informations sont correctes, vous pouvez stocker les informations d'authentification de l'utilisateur dans des variables de session ou des cookies.

Ensuite, vous pouvez ajouter une vérification de l'authentification sur chaque page du back-office en vérifiant si les variables de session ou les cookies existent. Si les variables ne sont pas trouvées, l'utilisateur sera redirigé vers la page de connexion.

Vous pouvez également implémenter des niveaux d'autorisation en stockant les informations d'autorisation de l'utilisateur dans des variables de session ou des cookies. Ensuite, vous pouvez vérifier les informations d'autorisation sur chaque page et déterminer si l'utilisateur a accès à la page en question.

Il est important de noter que vous devez prendre des précautions de sécurité telles que le hachage des mots de passe et l'utilisation de connexions sécurisées (HTTPS) pour empêcher les attaques telles que les injections SQL ou les interceptions de données.

Exemple simple de formulaire d'authentification utilisant PHP pour vérifier les informations de connexion d'un utilisateur :

```
<form action="authenticate.php" method="post">
  <label for="username">Nom d'utilisateur :</label>
  <input type="text" name="username" id="username">

  <label for="password">Mot de passe :</label>
  <input type="password" name="password" id="password">

  <input type="submit" value="Se connecter">
</form>
```

Le fichier authenticate.php utiliserait les informations soumises pour vérifier si le nom d'utilisateur et le mot de passe correspondent à ceux dans la base de données. Si les informations sont valides, une session est démarrée pour l'utilisateur et il est redirigé vers le back-office. Sinon, un message d'erreur serait affiché.

Exemple simple de fichier authenticate.php qui gère l'authentification et les autorisations pour accéder à un back-office.

```
<?php
session_start();

// Vérifie si le nom d'utilisateur et le mot de passe sont envoyés à partir du formulaire
if (isset($_POST['username']) && isset($_POST['password'])) {
  $username = $_POST['username'];
  $password = $_POST['password'];

  // Vérifie si le nom d'utilisateur et le mot de passe sont corrects
  if ($username == 'admin' && $password == 'secret') {
    // Stocker le nom d'utilisateur dans une variable de session
    $_SESSION['username'] = $username;
```

```
    // Rediriger vers la page de back-office
    header('Location: backoffice.php');
    exit;
  } else {
    // Enregistrer un message d'erreur
    $_SESSION['error'] = 'Nom d\'utilisateur ou mot de passe incorrect';

    // Rediriger vers la page de connexion
    header('Location: login.php');
    exit;
  }
}

// Vérifie si l'utilisateur est déjà connecté
if (!isset($_SESSION['username'])) {
  // Rediriger vers la page de connexion
  header('Location: login.php');
  exit;
}
```

Ce code vérifie si un nom d'utilisateur et un mot de passe ont été envoyés à partir du formulaire de connexion, et vérifie si les données sont correctes en comparant les données envoyées avec un nom d'utilisateur et un mot de passe prédéfinis. Si les données sont correctes, le nom d'utilisateur est stocké dans une variable de session et l'utilisateur est redirigé vers la page de back-office. Si les données ne sont pas correctes, un message d'erreur est enregistré et l'utilisateur est redirigé vers la page de connexion. Si l'utilisateur n'est pas déjà connecté, il est redirigé vers la page de connexion.

Utilisation d'une base de données MySql

L'utilisation d'une base de données MySql dans un projet de développement web consiste à stocker les données de l'application dans une base de données relationnelle. Cela permet de centraliser les informations, de les organiser de manière cohérente et de les rendre accessibles rapidement et efficacement. Les données sont stockées dans des tables et peuvent être manipulées à l'aide de requêtes SQL. MySql est un système de gestion de bases de

données largement utilisé pour les applications web en raison de sa rapidité, de sa fiabilité et de sa facilité d'utilisation.

Gérer les bases MySQL avec phpMyAdmin

phpMyAdmin est un outil en ligne de commande open source pour gérer les bases de données MySQL. Il permet de créer, supprimer et modifier des bases de données et des tables, ainsi que d'exécuter des requêtes SQL. phpMyAdmin propose également une interface graphique pour les utilisateurs qui n'ont pas de connaissances en SQL, offrant ainsi une alternative simple et facile à l'utilisation de la ligne de commande. Les fonctionnalités incluent la gestion de la structure de la base de données, la gestion des données, l'exportation et l'importation de données, la gestion des utilisateurs et des privilèges, et plus encore.

Concepts fondamentaux: Bases, tables, champs, enregistrements

Les concepts fondamentaux en base de données sont les suivants:

Bases: une base de données est un système de stockage qui peut contenir de nombreuses tables de données reliées entre elles.

Tables: une table est une structure de données qui peut contenir plusieurs enregistrements et plusieurs champs. Chaque table peut être considérée comme une liste d'objets similaires tels que les clients, les produits ou les commandes.

Champs: un champ est une colonne dans une table qui représente une information spécifique à propos d'un enregistrement, comme le nom ou l'adresse email.

Enregistrements: un enregistrement est une ligne dans une table qui représente une instance spécifique d'un objet, comme un client particulier ou une commande spécifique. Chaque enregistrement comporte plusieurs champs qui décrivent les détails de cet objet.

Exemple simple pour expliquer les concepts fondamentaux de bases de données MySQL :

Supposons que vous souhaitez créer une base de données pour enregistrer les informations sur les employés d'une entreprise.

La base de données serait la "base de données employés", qui contiendrait toutes les informations sur les employés.

Une table dans la base de données serait la "table employés", qui contiendrait les informations détaillées sur chaque employé, telles que le nom, l'adresse, le numéro de téléphone, etc.

Les champs seraient les différentes colonnes dans la table, telles que "nom", "adresse", "numéro de téléphone".

Les enregistrements seraient les différentes lignes dans la table, contenant les informations sur chaque employé, telles que "John Doe, 123 Main St, 555-555-5555".

C'est un exemple simple mais il montre comment les bases de données, les tables, les champs et les enregistrements peuvent être utilisés pour organiser et stocker des informations dans une base de données MySQL.

Fonctions PHP Mysql

Les fonctions PHP MySQL permettent de gérer les bases de données MySQL à partir d'un script PHP. Il existe un certain nombre de fonctions prédéfinies qui couvrent des tâches courantes telles que la connexion à une base de données, la récupération de données, la mise à jour et la suppression de données. Les fonctions les plus couramment utilisées incluent :

mysql_connect() : pour établir une connexion à la base de données MySQL

mysql_select_db() : pour sélectionner une base de données à utiliser

mysql_query() : pour exécuter une requête SQL sur la base de données

mysql_fetch_array() : pour obtenir des enregistrements sous forme de tableau associatif

mysql_num_rows() : pour obtenir le nombre d'enregistrements dans un jeu de résultats

mysql_close() : pour fermer la connexion à la base de données.

Cependant, il est important de noter que ces fonctions sont considérées comme obsolètes et peu sûres en raison de problèmes de sécurité, il est donc recommandé d'utiliser des extensions telles que mysqli ou PDO pour gérer les bases de données.

Exemple de formulaire en PHP qui utilise les fonctions MySQL pour interagir avec une base de données pourrait être :

```
<form action="submit.php" method="post">
  <label for="username">Nom d'utilisateur :</label>
  <input type="text" id="username" name="username"><br><br>

  <label for="email">Adresse email :</label>
  <input type="email" id="email" name="email"><br><br>

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

submit.php :

```
<?php
```

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "database_name";

// Connexion à la base de données
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Vérification de la connexion
if (!$conn) {
  die("Erreur de connexion : " . mysqli_connect_error());
}

// Récupération des données du formulaire
$username = $_POST['username'];
$email = $_POST['email'];

// Création de la requête SQL
$sql = "INSERT INTO users (username, email)
VALUES ('$username', '$email')";

// Exécution de la requête SQL
if (mysqli_query($conn, $sql)) {
  echo "Nouvel enregistrement créé avec succès";
} else {
  echo "Erreur : " . $sql . "<br>" . mysqli_error($conn);
}

// Fermeture de la connexion
```

```
mysqli_close($conn);

?>
```

Ce script permet de récupérer les données d'un formulaire, de les enregistrer dans une base de données MySQL en utilisant les fonctions PHP mysqli, et de renvoyer un message de confirmation ou d'erreur en fonction du résultat de l'opération d'enregistrement.

Exemple simple d'une connexion à une base de données MySQL en utilisant l'approche PDO (PHP Data Objects) :

```
<?php


// Paramètres de connexion à la base de données
$host = 'localhost';
$dbname = 'ma_base_de_donnees';
$username = 'utilisateur';
$password = 'mot_de_passe';


// Connexion à la base de données
try {
  $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
  $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
  die("Erreur lors de la connexion à la base de données : " . $e->getMessage());
}


// Exécution d'une requête SQL
try {
  $stmt = $pdo->prepare('SELECT * FROM ma_table');
  $stmt->execute();
  $resultats = $stmt->fetchAll();
} catch (PDOException $e) {
  die("Erreur lors de l'exécution de la requête : " . $e->getMessage());
```

```
}


// Affichage des résultats
foreach ($resultats as $ligne) {
  echo '<pre>';
  print_r($ligne);
  echo '</pre>';
}


// Fermeture de la connexion
$pdo = null;


?>
```

Cet exemple montre comment établir une connexion à une base de données en utilisant les paramètres de connexion, puis comment exécuter une requête SQL pour récupérer des données dans un tableau. Les résultats sont ensuite affichés à l'aide d'une boucle foreach. Enfin, la connexion est fermée pour libérer les ressources.

Exemple de code PHP utilisant une approche PDO pour gérer une connexion à une base de données MySQL avec un formulaire de connexion :

```
<?php
if(isset($_POST['submit'])){
  $host = 'localhost';
  $user = 'username';
  $password = 'password';
  $dbname = 'database_name';


  try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $user, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);


    $username = $_POST['username'];
```

```
    $password = $_POST['password'];


    $stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username AND password = :password");
    $stmt->execute(array(':username' => $username, ':password' => $password));
    $result = $stmt->fetch(PDO::FETCH_ASSOC);


    if ($result) {
      // Start session
      session_start();


      // Store data in session variables
      $_SESSION["loggedin"] = true;
      $_SESSION["username"] = $username;


      // Redirect user to welcome page
      header("location: welcome.php");
    } else {
      echo "Incorrect username or password.";
    }


  } catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
  }


  $pdo = null;
}
?>
```

```
<!DOCTYPE html>
```

```
<html>
<head>
 <title>Login Form</title>
</head>
<body>
 <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required><br><br>
  <label for="password">Password:</label>
  <input type="password" id="password" name="password" required><br><br>
  <input type="submit" name="submit" value="Submit">
 </form>
</body>
</html>
```

Ce code effectue les actions suivantes :

Vérifie si le formulaire a été soumis en utilisant isset($_POST['submit'])

Initialise les informations de connexion à la base de données

Utilise PDO pour établir une connexion à la base de données et préparer une requête SQL

Exécute la requête avec les données du formulaire en utilisant les placeholders pour la sécurité

Vérifie si les informations de connexion sont correctes en comparant les données renvoyées par la requête à celles du formulaire

Si les informations sont correctes, démarre la session et stocke les variables de session

Si les informations sont incorrectes

Introduction au langage SQL (sélection, modification, suppression)

SQL (Structured Query Language) est un langage standard de manipulation de données pour les bases de données relationnelles. Il permet de travailler avec les bases de données en effectuant des opérations telles que la sélection, la modification et la suppression de données. Les opérations les plus courantes dans SQL sont les suivantes :

Sélection : utilisée pour récupérer des données d'une table. Exemple : "SELECT * FROM users"

Modification : utilisée pour mettre à jour des données existantes dans une table. Exemple : "UPDATE users SET email = 'new_email@example.com' WHERE id = 2"

Suppression : utilisée pour supprimer des données d'une table. Exemple : "DELETE FROM users WHERE id = 3".

Il existe également d'autres opérations telles que l'insertion de données, la jointure de tables, l'agrégation de données, etc. Il est important de comprendre les concepts fondamentaux de SQL pour travailler avec des bases de données relationnelles.

Exemple, nous allons utiliser un formulaire pour effectuer des opérations sur une base de données MySQL.

Créer une base de données MySQL nommée "test_db" avec une table nommée "utilisateurs". La table utilisateurs aura les champs suivants : "id", "nom_utilisateur", "mot_de_passe".

Créer un formulaire HTML pour se connecter à la base de données et effectuer des opérations. Le formulaire aura deux champs : "nom_utilisateur" et "mot_de_passe".

Écrire un script PHP pour traiter les données du formulaire. Ce script vérifiera les informations de connexion et effectuera les opérations correspondantes : sélection, modification ou suppression.

Exemple de code PHP pour la vérification des informations de connexion et l'exécution d'une requête SQL :

```
<?php

// Connexion à la base de données
$serveur = "localhost";
$username = "root";
$password = "";
$database = "test_db";

$connexion = mysqli_connect($serveur, $username, $password, $database);

// Vérification de la connexion
if (!$connexion) {
```

```
    die("Erreur de connexion : " . mysqli_connect_error());
}

// Récupération des données du formulaire
$nom_utilisateur = $_POST["nom_utilisateur"];
$mot_de_passe = $_POST["mot_de_passe"];

// Requête SQL pour sélectionner l'enregistrement correspondant au nom d'utilisateur et au mot de passe
$requete = "SELECT * FROM utilisateurs WHERE nom_utilisateur = '$nom_utilisateur' AND mot_de_passe = '$mot_de_passe'";
$resultat = mysqli_query($connexion, $requete);

// Vérification du résultat
if (mysqli_num_rows($resultat) == 1) {
    // Connexion réussie
    // ...
} else {
    // Connexion échouée
    // ...
}

// Fermeture de la connexion
mysqli_close($connexion);
?>
```

Ce n'est qu'un exemple de base, et il est fortement recommandé de sécuriser les informations de connexion et de protéger contre les attaques telles que les injections SQL.

Traitement des résultats des requêtes

Le traitement des résultats des requêtes SQL consiste à gérer les données retournées par la base de données pour les utiliser dans votre application PHP. Pour ce faire, vous pouvez utiliser les fonctions PHP telles que mysqli_query() ou PDO::query(). Une fois la requête

exécutée, le résultat peut être parcouru à l'aide de fonctions telles que mysqli_fetch_assoc() ou PDO::fetch(). Vous pouvez alors utiliser les données retournées pour peupler une table, alimenter un graphique, afficher une liste déroulante, etc. Il est important de vérifier que la requête a réussi et que des données ont été retournées avant d'essayer de les utiliser pour éviter des erreurs.

Exemple d'utilisation d'un formulaire pour traiter les résultats des requêtes :

```
<form action="traitement.php" method="post">
  <label for="nom">Entrez votre nom :</label>
  <input type="text" id="nom" name="nom">
  <input type="submit" value="Envoyer">
</form>
```

PHP :

```
<?php
if (isset($_POST['nom'])) {
  $nom = $_POST['nom'];

  // Connexion à la base de données
  $host = "localhost";
  $user = "utilisateur";
  $password = "motdepasse";
  $dbname = "basededonnees";
  $connexion = mysqli_connect($host, $user, $password, $dbname);

  // Requête SQL pour récupérer les informations
  $requete = "SELECT * FROM table WHERE nom='$nom'";
  $resultat = mysqli_query($connexion, $requete);

  // Traitement des résultats
  while ($ligne = mysqli_fetch_assoc($resultat)) {
    echo "Nom : " . $ligne['nom'] . "<br>";
```

```
  echo "Prénom : " . $ligne['prenom'] . "<br>";
  echo "Adresse : " . $ligne['adresse'] . "<br>";
 }

 // Fermeture de la connexion
 mysqli_close($connexion);
}
?>
```

Dans cet exemple, nous avons un formulaire HTML qui permet à l'utilisateur d'entrer son nom. Lorsque l'utilisateur soumet le formulaire, les données sont envoyées à la page "traitement.php".

Dans la page PHP, nous vérifions d'abord si les données du formulaire sont disponibles en utilisant la fonction isset(). Si les données sont présentes, nous les enregistrons dans une variable et nous effectuons une connexion à notre base de données en utilisant les fonctions mysqli_connect() et mysqli_query().

Ensuite, nous exécutons une requête SQL pour récupérer les informations associées au nom entré par l'utilisateur. Nous utilisons la fonction mysqli_fetch_assoc() pour traiter les résultats de la requête et afficher les informations correspondantes.

Enfin, nous fermons la connexion à la base de données en utilisant la fonction mysqli_close().

Travaux Pratiques

Création d'une base Mysql

Pour créer une base de données MySQL, vous pouvez utiliser le logiciel phpMyAdmin ou un terminal de commande.

Exemple avec phpMyAdmin :

Connectez-vous à votre compte phpMyAdmin.

Cliquez sur l'onglet "Bases de données" dans le menu de gauche.

Saisissez le nom de votre nouvelle base de données dans le champ "Créer une nouvelle base de données".

Sélectionnez le type de codage que vous souhaitez utiliser et cliquez sur le bouton "Créer".

Exemple en utilisant le terminal de commande :

Connectez-vous à votre serveur MySQL en utilisant le client mysql.

Exécutez la commande suivante pour créer une nouvelle base de données :

CREATE DATABASE nom_de_la_base_de_données;

Vérifiez si la base de données a été créée en exécutant la commande suivante :

SHOW DATABASES;

Remplissage de la base à partir d'une base texte

Pour remplir une base de données MySQL à partir d'une base de données texte, vous pouvez suivre les étapes suivantes :

Préparer le fichier texte : Assurez-vous que votre fichier texte est formaté correctement pour être importé dans la base de données. Cela signifie que les données doivent être séparées par des virgules ou des tabulations, et qu'il doit y avoir une ligne d'en-tête qui définit les noms de colonne.

Créer une table dans votre base de données : Pour commencer, vous devez créer une table dans votre base de données qui sera utilisée pour stocker les données importées à partir du fichier texte. Vous pouvez le faire en utilisant l'interface web de phpMyAdmin, en exécutant une requête SQL, ou en utilisant un outil de développement de base de données.

Importation des données : Une fois la table créée, vous pouvez importer les données à partir du fichier texte en utilisant la commande SQL "LOAD DATA INFILE". Cette commande vous permet de spécifier le chemin vers le fichier texte, et comment les données doivent être séparées.

Vérifier les données importées : Une fois les données importées, vous pouvez exécuter une requête SELECT pour vérifier que les données sont correctement importées dans la base de données.

Il est important de noter que ces étapes peuvent varier en fonction de la structure de votre fichier texte et de la base de données que vous utilisez. Il est donc toujours bon de consulter la documentation de votre base de données pour obtenir les instructions spécifiques à votre situation.

Exemple de remplissage de base de données MySQL à partir d'une base de données texte :

Importation du fichier texte dans la base de données :

```
CREATE TABLE customers (
```

```
 id INT AUTO_INCREMENT PRIMARY KEY,
 name VARCHAR(255),
 email VARCHAR(255),
 age INT
);
```

LOAD DATA INFILE '/path/to/customers.txt'

INTO TABLE customers

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(name, email, age);

Vérification du remplissage :

SELECT * FROM customers;

Ce code permet de créer une table customers avec les champs id, name, email et age, et de charger les données du fichier texte /path/to/customers.txt dans la table en utilisant la commande LOAD DATA INFILE. Les données sont séparées par des virgules et chaque ligne est terminée par un retour à la ligne. La première ligne du fichier texte est ignorée car elle contient les en-têtes de colonne. Enfin, la requête SELECT * FROM customers permet de vérifier que les données ont bien été chargées dans la table.

Les tableaux complexes

Les tableaux complexes en PHP peuvent être utilisés pour stocker des informations complexes dans un seul objet. Les tableaux complexes peuvent inclure des informations de différents types, telles que des valeurs numériques, des chaînes de caractères, des tableaux et des objets. Par exemple, un tableau complexe peut être utilisé pour stocker les informations d'un utilisateur, telles que son nom d'utilisateur, son adresse e-mail, son numéro de téléphone et ses informations de connexion. Les tableaux complexes peuvent être itérés, triés et modifiés en utilisant différentes fonctions PHP.

Exemple de tableau complexe peut être un tableau associatif qui associe une clé à une valeur.

```
$personnes = array(
"nom" => "Dupont",
"prenom" => "Jean",
```

```
"age" => 35,
"ville" => "Paris"
);
```

Dans ce cas, la clé "nom" est associée à la valeur "Dupont", la clé "prenom" est associée à la valeur "Jean", etc. On peut accéder à ces valeurs en utilisant leur clé comme index :

```
echo $personnes["nom"];

// Affiche "Dupont"
```

Constructeur array

Le constructeur array en PHP est une fonction qui permet de créer un tableau. La syntaxe pour créer un tableau est la suivante :

```
$tableau = array(
  'clé1' => 'valeur1',
  'clé2' => 'valeur2',
  ...
);
```

Vous pouvez également utiliser la syntaxe suivante :

```
$tableau = [
  'clé1' => 'valeur1',
  'clé2' => 'valeur2',
  ...
];
```

Le constructeur array peut être utilisé pour stocker une série de valeurs associées à des clés. Les clés peuvent être des nombres entiers ou des chaînes de caractères, et les valeurs peuvent être de n'importe quel type, y compris d'autres tableaux. Les tableaux sont souvent utilisés pour stocker des ensembles de données liées telles que des listes de produits, des données de formulaire, etc.

Fonctions associées aux tableaux

Il existe plusieurs fonctions associées aux tableaux en PHP qui permettent de gérer et manipuler les données contenues dans les tableaux. Quelques exemples incluent :

sort() - Trie les éléments d'un tableau dans l'ordre croissant ou décroissant.

count() - Retourne le nombre d'éléments dans un tableau.

array_pop() - Supprime le dernier élément d'un tableau.

array_push() - Ajoute un ou plusieurs éléments à la fin d'un tableau.

array_keys() - Retourne les clés d'un tableau.

array_values() - Retourne les valeurs d'un tableau.

array_merge() - Fusionne plusieurs tableaux en un seul tableau.

array_slice() - Extrait une portion d'un tableau.

Ces fonctions peuvent être utilisées pour effectuer des tâches telles que la trier, la recherche, la modification, la suppression et l'ajout d'éléments dans un tableau.

Exemple simple d'utilisation de fonctions associées aux tableaux en PHP :

```
$fruits = array("banane", "pomme", "poire", "cerise");

// La fonction count retourne le nombre d'éléments dans un tableau
echo count($fruits); // affiche "4"

// La fonction sort trie les éléments d'un tableau dans l'ordre alphabétique
sort($fruits);

// La fonction in_array vérifie si un élément est présent dans un tableau
if(in_array("pomme", $fruits)) {
   echo "La pomme est présente dans le tableau de fruits";
} else {
   echo "La pomme n'est pas présente dans le tableau de fruits";
```

```
}


// La fonction implode transforme un tableau en chaîne de caractères séparées par un séparateur

$fruits_string = implode(", ", $fruits);

echo $fruits_string; // affiche "banane, cerise, poire, pomme"
```

Ce sont là quelques exemples parmi les nombreuses fonctions associées aux tableaux disponibles en PHP.

Fonctions d'extraction

Les fonctions d'extraction sont des fonctions qui permettent de récupérer des informations spécifiques à partir d'un tableau. Certaines des fonctions les plus couramment utilisées sont array_pop, array_shift, array_slice, array_splice, array_values, array_keys et array_unique. Les fonctions d'extraction sont utiles pour travailler avec des tableaux et manipuler leurs données en fonction de vos besoins. Par exemple, vous pouvez utiliser array_slice pour extraire une plage de valeurs d'un tableau, ou array_splice pour ajouter ou supprimer des éléments d'un tableau existant.

Exemple d'utilisation de fonctions d'extraction sur un tableau en PHP :

```
<?php


$array = array("red", "blue", "green", "yellow");


// Extraction de la première valeur du tableau

$first = array_shift($array);

echo "La première valeur est : $first <br>";

// Extraction de la dernière valeur du tableau

$last = array_pop($array);

echo "La dernière valeur est : $last <br>";

// Affichage du tableau restant

print_r($array);


?>
```

Ce code affichera :

La première valeur est : red

La dernière valeur est : green

Array ([0] => blue [1] => yellow)

Les formulaires complexes

Les formulaires complexes sont des formulaires qui contiennent plusieurs champs et sont utilisés pour collecter des informations plus détaillées. Ils peuvent inclure des champs tels que des sélections multiples, des cases à cocher, des listes déroulantes, des zones de texte, des fichiers, etc. La mise en place de formulaires complexes peut être plus complexe que les formulaires simples, car il peut être nécessaire de valider les données soumises et de les traiter avant de les stocker dans une base de données ou de les utiliser pour effectuer d'autres tâches.

Moteur de recherche: formulaire en relation avec une base de données

Un moteur de recherche peut être implémenté en utilisant un formulaire HTML qui envoie les requêtes à une base de données via PHP. Lorsque l'utilisateur soumet le formulaire, PHP envoie une requête SQL à la base de données pour rechercher les informations correspondant à la demande de l'utilisateur. Les résultats de la requête sont alors retournés à l'utilisateur sous forme de liste, tableau ou autre format souhaité.

Exemple simple de code pour un formulaire de recherche qui envoie les requêtes à une base de données :

```
<form action="search.php" method="post">
 <input type="text" name="search_term">
 <input type="submit" value="Rechercher">
</form>
```

Et voici un exemple de code pour le script PHP qui traite les requêtes soumises à partir du formulaire :

```
<?php
 $search_term = $_POST['search_term'];

 $pdo = new PDO('mysql:host=localhost;dbname=database_name', 'username', 'password');
 $query = "SELECT * FROM table_name WHERE field_name LIKE '%$search_term%'";
 $statement = $pdo->prepare($query);
 $statement->execute();
```

```
 $results = $statement->fetchAll();


 foreach ($results as $result) {
   echo $result['field_name'] . '<br>';
 }
?>
```

Ce code utilise une approche PDO pour se connecter à la base de données, préparer la requête SQL et exécuter la requête. Les résultats de la requête sont stockés dans une variable, qui est ensuite bouclée pour afficher les informations correspondantes à la demande de l'utilisateur.

Fonctions avancées de sélection: modification de la base, tris (ORDER BY), recherches (WHERE)

Les fonctions avancées de sélection permettent de modifier une base de données, d'effectuer des tris sur les données et de faire des recherches plus complexes. La clause "ORDER BY" permet de trier les données selon un ou plusieurs critères spécifiés. La clause "WHERE" permet de sélectionner des enregistrements en fonction de conditions spécifiées. Par exemple, on peut utiliser la clause "WHERE" pour sélectionner uniquement les enregistrements dont la valeur d'une colonne est supérieure à une certaine valeur. Les fonctions avancées de sélection peuvent être utilisées avec des requêtes SQL telles que SELECT, UPDATE, DELETE, etc.

Exemple de comment utiliser les fonctions avancées de sélection avec une base de données MySQL et PHP :

Connexion à la base de données :

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "database_name";


// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);


// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
```

```
}
echo "Connected successfully";
?>
```

Exemple de requête pour trier les enregistrements selon un champ de la table (ORDER BY) :

```
<?php
$sql = "SELECT * FROM users ORDER BY age DESC";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
   // output data of each row
   while($row = $result->fetch_assoc()) {
      echo "id: " . $row["id"]. " - Name: " . $row["username"]. " - Age: " . $row["age"]. "<br>";
   }
} else {
   echo "0 results";
}
$conn->close();
?>
```

Exemple de requête pour effectuer une recherche dans la base de données en utilisant la clause WHERE :

```
<?php
$sql = "SELECT * FROM users WHERE username='John'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
   // output data of each row
   while($row = $result->fetch_assoc()) {
      echo "id: " . $row["id"]. " - Name: " . $row["username"]. " - Age: " . $row["age"]. "<br>";
   }
} else {
```

```
   echo "0 results";
}
$conn->close();
?>
```

Ceci est un exemple simple pour montrer comment utiliser les fonctions avancées de sélection avec MySQL et PHP. Il peut être étendu ou personnalisé en fonction des besoins de votre projet.

Le graphisme

Le graphisme est une composante importante de la conception et de la présentation visuelle d'un site web. Il inclut la sélection des couleurs, des images, des polices et des images pour la mise en page et la mise en forme du contenu. Le graphisme peut également inclure des animations et des graphiques pour aider à transmettre les informations de manière claire et attrayante. L'objectif principal du graphisme est de rendre un site web attractif et convivial pour les utilisateurs, tout en transmettant efficacement les informations nécessaires.

Présentation de la librairie GD

La librairie GD est une bibliothèque logicielle pour la création dynamique d'images. Elle offre une variété de fonctions pour manipuler et travailler avec des images telles que la création d'images, la modification d'images existantes, la redimensionnement d'images, la rotation d'images, l'ajout de textes, de formes et de couleurs à des images, etc. La librairie GD est largement utilisée en conjonction avec le langage de programmation PHP pour créer et générer des images à la volée pour des sites Web, des applications et des systèmes.

Un exemple d'utilisation de la librairie GD peut être la création d'une image simple, comme un cercle. Voici un code PHP qui utilise GD pour créer une image de cercle rouge de 500 pixels de diamètre :

```
<?php

// Créer une nouvelle image de 500 x 500 pixels
$image = imagecreatetruecolor(500, 500);

// Allouer une couleur rouge pour remplir le cercle
$red = imagecolorallocate($image, 255, 0, 0);

// Dessiner un cercle rempli avec la couleur rouge
```

```
imagefilledellipse($image, 250, 250, 500, 500, $red);


// En-tête HTTP pour indiquer le type de fichier
header('Content-Type: image/jpeg');


// Enregistrer l'image dans un fichier jpeg
imagejpeg($image);


// Libérer la mémoire
imagedestroy($image);


?>
```

Ce code peut être enregistré dans un fichier PHP et exécuté pour voir le résultat. L'image créée peut également être enregistrée sur le disque en utilisant la fonction imagejpeg ou imagepng selon le format souhaité.

Création d'image, réutilisation

Superposition de texte pour protection de droits

Intégration au site

L'utilisation de la bibliothèque GD en PHP permet de créer, manipuler et générer des images à partir de données dynamiques. Par exemple, vous pouvez utiliser GD pour générer des images en fonction d'entrées utilisateur, pour superposer du texte sur des images pour la protection des droits d'auteur, ou pour créer des images de graphs pour la visualisation des données.

Exemple simple de la création d'une image à l'aide de GD en PHP :

```
<?php
header("Content-Type: image/png");
$im = imagecreate(200, 50);
$blanc = imagecolorallocate($im, 255, 255, 255);
$noir = imagecolorallocate($im, 0, 0, 0);
imagestring($im, 5, 50, 25, "Hello World!", $noir);
imagepng($im);
```

```
imagedestroy($im);
?>
```

Ce code crée une image PNG de 200x50 pixels, avec le fond blanc et le texte "Hello World!" en noir.

Une fois la création de l'image terminée, vous pouvez l'intégrer à votre site en utilisant une URL qui appelle ce script PHP. Vous pouvez également sauvegarder l'image sur le disque dur pour une utilisation future.

Il est important de noter que les fonctions GD consomment beaucoup de ressources système, il est donc important d'être prudent dans leur utilisation pour éviter les problèmes de performance sur votre site.

Exemple de création d'image avec la librairie GD en PHP :

```
<?php
// Création d'une image avec une largeur de 300px et une hauteur de 200px
$image = imagecreatetruecolor(300, 200);

// Définition de la couleur de fond en blanc
$white = imagecolorallocate($image, 255, 255, 255);
imagefill($image, 0, 0, $white);

// Définition de la couleur du texte en noir
$black = imagecolorallocate($image, 0, 0, 0);
// Ajout du texte "Hello World!" en noir sur l'image
imagestring($image, 5, 75, 100, "Hello World!", $black);

// Enregistrement de l'image au format PNG
imagepng($image, "example.png");

// Libération de la mémoire associée à l'image
imagedestroy($image);
?>
```

Cet exemple crée une image blanche avec la largeur de 300px et la hauteur de 200px, puis ajoute le texte "Hello World!" en noir au milieu de l'image. Finalement, l'image est enregistrée au format PNG sous le nom de "example.png".