# Cours de PHP : Du Début jusqu'au CRUD

---

## Chapitre 1 : Introduction à PHP

### 1.1. Qu'est-ce que PHP ?

PHP (Hypertext Preprocessor) est un langage de script côté serveur. Il est utilisé pour développer des applications web dynamiques qui peuvent interagir avec des bases de données. PHP est souvent intégré dans le code HTML, ce qui permet de créer des pages web dynamiques. Voici un exemple d'une page PHP très simple :

```
<?php
    echo "Bonjour, monde!";
?>
```

Lorsque ce script est exécuté sur un serveur web, il affiche "Bonjour, monde!" dans le navigateur.

### 1.2. Installation de PHP

Pour exécuter des scripts PHP, tu dois installer un serveur web comme Apache, et PHP lui-même. La méthode la plus simple consiste à installer un package tel que :

- **XAMPP** (pour Windows, Linux, Mac)
- **WAMP** (pour Windows)
- **MAMP** (pour Mac)

Ces packages incluent tout ce dont tu as besoin pour commencer à coder en PHP.

### 1.3. Exécution de ton premier script PHP

Une fois PHP installé, crée un fichier nommé `index.php` dans le répertoire de ton serveur web (généralement `htdocs` pour XAMPP). Place-y le code suivant :

```
<?php
    echo "Ceci est mon premier script PHP!";
?>
```

Ensuite, ouvre ton navigateur et tape `http://localhost/index.php`. Tu devrais voir le message "Ceci est mon premier script PHP!" s'afficher.

**Exercice**: Installe un environnement de développement PHP (XAMPP, WAMP, ou MAMP) et crée un fichier PHP qui affiche ton nom.

---

## Chapitre 2 : Les Variables en PHP

### 2.1. Qu'est-ce qu'une Variable ?

Une variable en PHP est un conteneur qui permet de stocker des informations, comme un texte ou un nombre. Les variables en PHP commencent par un symbole dollar `$` suivi du nom de la variable.

```
<?php
    $nom = "Alice";
    $age = 30;
    echo "Nom: " . $nom . ", Age: " . $age;
?>
```

Dans cet exemple, `$nom` et `$age` sont des variables qui stockent respectivement une chaîne de caractères ("Alice") et un nombre entier (30).

### 2.2. Les Types de Données

Les variables en PHP peuvent stocker différents types de données, dont les principaux sont :

- **String** (Chaîne de caractères) : `"Bonjour"`
- **Integer** (Entier) : `25`
- **Float** (Nombre à virgule flottante) : `12.3`
- **Boolean** (Booléen) : `true` ou `false`
- **Array** (Tableau) : `["Pomme", "Banane"]`

**Exemple** :

```
<?php
    $prix = 19.99;
    $disponible = true;
    $fruits = ["Pomme", "Banane", "Orange"];
    echo "Prix: " . $prix . ", Disponible: " . $disponible;
?>
```

### 2.3. Comment Utiliser les Variables

Les variables peuvent être utilisées dans des opérations, concaténées dans des chaînes de caractères, et bien plus encore. Par exemple :

```
<?php
    $a = 10;
    $b = 20;
    $somme = $a + $b;
    echo "La somme de a et b est : " . $somme;
?>
```

**Exercice**: Crée un script PHP qui déclare trois variables : un texte, un nombre, et un tableau. Affiche chacune de ces variables.

---

## Chapitre 3 : Les Structures de Contrôle

### 3.1. Conditions `if`, `else`, et `elseif`

Les conditions permettent d'exécuter du code en fonction de certaines situations.

```
<?php

    $age = 20;
    if ($age >= 18) {
        echo "Vous êtes majeur.";
    } else {
        echo "Vous êtes mineur.";
    }
?>
```

### 3.2. Les Boucles `for`, `while`, et `foreach`

Les boucles permettent de répéter un bloc de code plusieurs fois.

- **for** : Pour un nombre fixe d'itérations.
- **while** : Tant qu'une condition est vraie.
- **foreach** : Pour parcourir les éléments d'un tableau.

**Exemple de boucle `for`**:

```
<?php
    for ($i = 0; $i < 5; $i++) {
        echo "Numéro: " . $i;
    }
?>
```

**Exercice**: Crée un script qui utilise une boucle `while` pour afficher les nombres de 1 à 10.

---

## Chapitre 4 : Les Fonctions en PHP

### 4.1. Qu'est-ce qu'une Fonction ?

Une fonction est un bloc de code qui effectue une tâche précise. Les fonctions peuvent recevoir des paramètres et retourner une valeur.

```php
<?php
    function saluer($nom) {
        return "Bonjour, " . $nom;
    }
    echo saluer("Alice");
?>
```

Cette fonction `saluer` prend un nom en paramètre et retourne un message de salutation.

### 4.2. Fonctions Prédéfinies en PHP

PHP propose de nombreuses fonctions intégrées pour manipuler des chaînes de caractères, des nombres, des tableaux, etc.

**Exemple avec `strlen()`** :

```php
<?php
    $texte = "Hello World";
    echo "La longueur du texte est: " . strlen($texte);
?>
```

**Exercice**: Crée une fonction qui calcule et retourne le carré d'un nombre.

---

## Chapitre 5 : Les Tableaux en PHP

### 5.1. Qu'est-ce qu'un Tableau ?

Un tableau est une collection de valeurs. Les valeurs peuvent être de tout type, et les tableaux peuvent être indexés (numériquement) ou associatifs (avec des clés).

- **Tableau Indexé** :

```php
<?php
    $fruits = ["Pomme", "Banane", "Orange"];
    echo $fruits[1]; // Affiche "Banane"
?>
```

- **Tableau Associatif** :

```php
<?php
    $personne = ["nom" => "Jean", "age" => 25];
    echo $personne["nom"]; // Affiche "Jean"
?>
```

### 5.2. Manipuler les Tableaux

Tu peux ajouter ou supprimer des éléments dans un tableau :

```php
<?php
    $fruits[] = "Cerise"; // Ajouter un élément
    unset($fruits[1]);    // Supprimer un élément
?>
```

**Exercice**: Crée un tableau associatif pour stocker les informations d'un livre (titre, auteur, prix) et affiche-les.

---

## Chapitre 6 : Introduction aux Bases de Données avec PHP

### 6.1. Qu'est-ce qu'une Base de Données ?

Une base de données est une collection organisée de données. PHP peut interagir avec des bases de données comme MySQL pour stocker, récupérer, mettre à jour, et supprimer des données.

### 6.2. Connexion à une Base de Données MySQL

Pour te connecter à une base de données MySQL, utilise l'objet `mysqli`.

```php
<?php
    $conn = new mysqli("localhost", "utilisateur", "mot_de_passe",
"base_de_donnees");
    if ($conn->connect_error) {
        die("Connexion échouée: " . $conn->connect_error);
    }
    echo "Connexion réussie!";
?>
```

### 6.3. Exécution de Requêtes SQL

Les requêtes SQL permettent d'interagir avec la base de données. Voici les types de requêtes courantes :

- **INSERT INTO** : Ajouter des données
- **SELECT** : Récupérer des données
- **UPDATE** : Mettre à jour des données
- **DELETE** : Supprimer des données

**Exemple de requête SELECT** :

```php
<?php
    $sql = "SELECT * FROM utilisateurs";
    $resultat = $conn->query($sql);
    while($row = $resultat->fetch_assoc()) {
        echo "Nom: " . $row["nom"];
    }
?>
```

**Exercice**: Connecte-toi à une base de données MySQL et récupère les informations d'une table appelée `etudiants`.

---

## Chapitre 7 : Le CRUD en PHP

### 7.1. Créer (Create)

Le CRUD représente les quatre opérations de base pour gérer les données : Create (créer), Read (lire), Update (mettre à jour), Delete (supprimer).

- **Exemple de Création** :

```
<?php
    $sql = "INSERT INTO utilisateurs (nom, email) VALUES ('Jean',
'jean@example.com')";
    if ($conn->query($sql) === TRUE) {
        echo "Nouvel enregistrement créé avec succès";
    } else {
        echo "Erreur: " . $sql . "<br>" . $conn->error;
    }
?>
```

### 7.2. Lire (Read)

```
<?php
    $sql = "SELECT * FROM utilisateurs";
    $resultat = $conn->query($sql);
    while($row = $resultat->fetch_assoc()) {
        echo "Nom: " . $row["nom"] . ", Email: " . $row["email"];
    }
?>
```

### 7.3. Mettre à Jour (Update)

```
<?php
    $sql = "UPDATE utilisateurs SET email='nouveau@example.com' WHERE
nom='Jean'";
    if ($conn->query($sql) === TRUE) {
        echo "Enregistrement mis à jour avec succès";
    } else {
        echo "Erreur de mise à jour: " . $conn->error;
    }
?>
```

### 7.4. Supprimer (Delete)

```
<?php
    $sql = "DELETE FROM utilisateurs WHERE nom='Jean'";
    if ($conn->query($sql) === TRUE) {
```

```
        echo "Enregistrement supprimé avec succès";
    } else {
        echo "Erreur de suppression: " . $conn->error;
    }
?>
```

**Exercice**: Crée une petite application PHP qui permet d'ajouter, lire, mettre à jour, et supprimer des utilisateurs dans une base de données.

## Chapitre 8 : Utilisation des Formulaires en PHP

### 8.1. Introduction aux Formulaires

Les formulaires HTML permettent aux utilisateurs de soumettre des données qui peuvent ensuite être traitées par un script PHP. Les formulaires sont souvent utilisés pour des tâches comme l'inscription d'utilisateurs, la soumission de commentaires, etc.

### 8.2. Création d'un Formulaire de Base

Un formulaire HTML typique inclut des champs de texte, des boutons radio, des cases à cocher, etc. Voici un exemple de formulaire simple :

```
<form action="traitement.php" method="post">
    Nom : <input type="text" name="nom">
    Email : <input type="email" name="email">
    <input type="submit" value="Soumettre">
</form>
```

- **`action`** : Spécifie le fichier PHP qui traitera les données soumises.
- **`method`** : Définit comment les données seront envoyées (`GET` ou `POST`).

### 8.3. Récupération des Données du Formulaire en PHP

Lorsque l'utilisateur soumet le formulaire, les données peuvent être récupérées en PHP via les superglobales `$_POST` ou `$_GET`, selon la méthode utilisée.

**Exemple de traitement (`traitement.php`)** :

```
<?php
    $nom = $_POST['nom'];
    $email = $_POST['email'];
    echo "Nom : " . $nom . "<br>Email : " . $email;
?>
```

### 8.4. Validation des Données du Formulaire

Avant de traiter les données, il est important de valider et de sécuriser les entrées utilisateur pour éviter les erreurs et les failles de sécurité.

**Validation de base** :

```
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        if (empty($_POST["nom"])) {
            echo "Le nom est requis.";
        } else {
            $nom = htmlspecialchars($_POST["nom"]);
            echo "Nom validé : " . $nom;
        }
    }
?>
```

**Exercice**: Crée un formulaire de contact avec les champs `Nom`, `Email`, et `Message`. Valide les champs pour s'assurer qu'ils ne sont pas vides avant de les afficher.

---

## Chapitre 9 : Affichage des Données dans un Tableau

### 9.1. Introduction à l'Affichage des Données

L'affichage des données dans un tableau HTML est une manière courante de présenter des informations structurées, comme les résultats d'une requête SQL.

### 9.2. Récupération et Affichage des Données

Imaginons que tu aies une table `utilisateurs` dans ta base de données. Tu peux récupérer ces données et les afficher dans un tableau HTML comme ceci :

**Exemple d'affichage** :

```
<?php
    // Connexion à la base de données
    $conn = new mysqli("localhost", "utilisateur", "mot_de_passe",
"base_de_donnees");

    // Requête SQL
    $sql = "SELECT id, nom, email FROM utilisateurs";
    $resultat = $conn->query($sql);

    // Affichage des données dans un tableau
    if ($resultat->num_rows > 0) {
        echo "<table
border='1'><tr><th>ID</th><th>Nom</th><th>Email</th></tr>";
        while($row = $resultat->fetch_assoc()) {
            echo "<tr><td>" . $row["id"]. "</td><td>" . $row["nom"].
"</td><td>" . $row["email"]. "</td></tr>";
```

```
        }
        echo "</table>";
    } else {
        echo "0 résultats";
    }
    $conn->close();
?>
```

### 9.3. Stylisation du Tableau

Tu peux également styliser ton tableau avec du CSS pour le rendre plus attrayant :

```
<style>
    table {
        width: 100%;
        border-collapse: collapse;
    }
    table, th, td {
        border: 1px solid black;
    }
    th, td {
        padding: 10px;
        text-align: left;
    }
    th {
        background-color: #f2f2f2;
    }
</style>
```

**Exercice**: Crée un script PHP qui affiche les informations de tous les utilisateurs dans un tableau HTML. Stylise le tableau pour le rendre plus lisible.

---

## Chapitre 10 : Gestion des Sessions en PHP

### 10.1. Qu'est-ce qu'une Session ?

Une session en PHP permet de stocker des informations sur l'utilisateur, qui peuvent être utilisées sur plusieurs pages. Contrairement aux cookies, les sessions stockent les informations sur le serveur et non sur l'ordinateur de l'utilisateur.

### 10.2. Démarrer une Session

Pour utiliser une session en PHP, il faut d'abord la démarrer avec `session_start()` :

```
<?php
    session_start();
    $_SESSION['nom_utilisateur'] = "Alice";
    echo "Session démarrée. Nom de l'utilisateur : " .
$_SESSION['nom_utilisateur'];
?>
```

### 10.3. Utilisation des Sessions

Les sessions sont particulièrement utiles pour maintenir des données utilisateur entre les pages, comme les informations de connexion, le contenu du panier d'achat, etc.

**Exemple** :

```
<?php
    session_start();
    if (!isset($_SESSION['visites'])) {
        $_SESSION['visites'] = 0;
    }
    $_SESSION['visites']++;
    echo "Nombre de visites : " . $_SESSION['visites'];
?>
```

- **`<?php`**
    - Cette ligne ouvre le bloc PHP. Tout le code PHP doit être inclus entre `<?php` et `?>`.
- **`session_start();`**
    - **Fonction** : `session_start()` démarre une session ou reprend une session existante. Elle est nécessaire pour accéder aux variables de session.
    - **Explication** : Si aucune session n'a encore été démarrée pour cet utilisateur, cette fonction en crée une nouvelle. Sinon, elle reprend la session existante en associant les données stockées pour cet utilisateur (basées sur un identifiant de session stocké généralement dans un cookie).
- **`if (!isset($_SESSION['visites'])) {`**
    - **Fonction** : `isset()` vérifie si une variable est définie et non nulle.
    - **Explication** : Cette ligne vérifie si la variable de session `$_SESSION['visites']` est définie. Si elle n'existe pas encore (c'est-à-dire si l'utilisateur visite le site pour la première fois), le code à l'intérieur des accolades sera exécuté.
- **`$_SESSION['visites'] = 0;`**
    - **Fonction** : Affectation d'une valeur à une variable de session.
    - **Explication** : Si `$_SESSION['visites']` n'est pas définie, elle est initialisée à 0. Cela signifie que l'utilisateur n'a pas encore visité la page ou que la session est nouvelle.
- **`}`**

- **Fonction** : Fin de la structure conditionnelle `if`.
- **Explication** : Cette accolade ferme le bloc de code conditionnel, indiquant la fin de l'exécution de la condition `if`.

### 10.4. Détruire une Session

Pour déconnecter un utilisateur ou effacer les données de session, utilise `session_unset()` et `session_destroy()` :

```
<?php
    session_start();
    session_unset();
    session_destroy();
    echo "Session détruite.";
?>
```

### 10.5. Sécurité des Sessions

Les sessions sont sensibles aux attaques telles que le vol de session. Pour sécuriser les sessions, il est recommandé de :

- Utiliser `session_regenerate_id()` pour changer l'ID de session après la connexion.
- Utiliser HTTPS pour protéger les données de session en transit.
- Mettre en place une durée d'expiration pour les sessions inactives.

un exemple de système de connexion simple en PHP utilisant les sessions. Ce système comporte trois fichiers principaux : un formulaire de connexion, un script de traitement pour la connexion, et une page de bienvenue qui affiche un message une fois l'utilisateur connecté, avec une option de déconnexion.

## 1. Formulaire de Connexion (`login.php`)

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Connexion</title>
</head>
<body>
    <h2>Connexion</h2>
    <form action="login_process.php" method="post">
        Nom d'utilisateur: <input type="text" name="username"
required><br><br>
        Mot de passe: <input type="password" name="password" required><br><br>
```

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

## 2. Script de Traitement de la Connexion (`login_process.php`)

```
<?php
session_start();

// Données d'utilisateur fictives (pour simplifier, en production, ces données
devraient provenir d'une base de données)
$utilisateur_valide = "Alice";
$motdepasse_valide = "12345";

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

// Vérification des informations d'identification
if ($username === $utilisateur_valide && $password === $motdepasse_valide) {
    // Si les informations sont correctes, démarre la session
    $_SESSION['username'] = $username;
    header("Location: welcome.php");
    exit();
} else {
    // Si les informations sont incorrectes, retourne au formulaire de
connexion
    echo "Nom d'utilisateur ou mot de passe incorrect.";
    echo "<br><a href='login.php'>Réessayer</a>";
}
?>
```

## 3. Page de Bienvenue (`welcome.php`)

```
<?php
session_start();

// Vérifie si l'utilisateur est connecté
if (!isset($_SESSION['username'])) {
    header("Location: login.php");
    exit();
}
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Bienvenue</title>
</head>
<body>
    <h2>Bienvenue, <?php echo $_SESSION['username']; ?>!</h2>
    <p>Vous êtes connecté.</p>
    <a href="logout.php">Se déconnecter</a>
```

```
</body>
</html>
```

## 4. Script de Déconnexion (`logout.php`)

```
<?php
session_start();

// Détruit la session et redirige vers le formulaire de connexion
session_unset();
session_destroy();
header("Location: login.php");
exit();
?>
```

## Explication :

1. **Formulaire de Connexion (`login.php`)** : L'utilisateur entre son nom d'utilisateur et son mot de passe. Ces informations sont envoyées à `login_process.php` pour être vérifiées.
2. **Script de Traitement de la Connexion (`login_process.php`)** : Ce script vérifie les informations d'identification contre les valeurs définies (ici, elles sont codées en dur pour simplifier). Si les informations sont correctes, la session démarre et l'utilisateur est redirigé vers la page de bienvenue. Sinon, un message d'erreur est affiché.
3. **Page de Bienvenue (`welcome.php`)** : Si l'utilisateur est connecté, un message de bienvenue avec son nom d'utilisateur est affiché, ainsi qu'un lien pour se déconnecter.
4. **Script de Déconnexion (`logout.php`)** : Ce script détruit la session de l'utilisateur, ce qui le déconnecte, puis il redirige vers le formulaire de connexion.