

# **PHP**

**Développement web avec PHP**



FORMATEUR : MAHAMANE SALISSOU YAHAYA : [ysalissou@gmail.com](mailto:ysalissou@gmail.com)

(TechnoLAB -ISTA)  
Master I GL

Décembre 2024

## A propos du formateur

- Mahamane Salissou YAHAYA
- Email : [ysalissou@gmail.com](mailto:ysalissou@gmail.com)
- Tél : +223 74 46 63 17
- Consultant en système d'information
- Résumé des compétences:
  - Développement JAVA/J2EE, PHP, DOTNET, ANDROID
  - Administrateur base de données
  - Mise en place et audit de système d'information de gestion
  - Conduite de projet informatique
  - Formateur vacataire
- Actuellement directeur technique chez SITAN INFORMATIQUE.
- Mes références :  
    Linkedin : <https://www.linkedin.com/in/salissou-yahaya-2a1b7071/>

# A propos des étudiants

- **Nom**
- **Expériences en développement**
- **Objectif de ce cours**

## Horaires de la formation

- **Durée : 24 heures**
- **Horaire : de 17h00 à 20h00**
- **15 minutes de pause au milieu de chaque séance**

# Organisation de l'enseignement

- **Objectifs**

- Apprenez le PHP sur sa version la plus récente avec la version 8
- Familiarisez vous avec les fonctionnalités natives du langage
- Apprenez les syntaxes modernes du langage
- Maitrisez la programmation orientée objet avec les classes, les traits et les interfaces
- Maitrisez toutes les bases du langage avec les types, les tableaux et les fonctions
- Apprenez à mettre vos applications en production avec Nginx, PHP-FPM et HTTPS
- Découvrez comment utiliser MySQL dans vos applications
- Découvrez Composer et l'autoloading

- **Prérequis**

- Des connaissance en HTML & CSS sont nécessaires
- Et ... du courage comme toujours !

- **Évaluations**

- Un QCM sur les notions importantes et projet de fin de module
- Il est important de venir à tous les TP
- Les notes sont individuelles

# Projet réalisé pendant la formation

Nous utiliserons l'ensemble des fonctionnalités de PHP pour mettre en place un projet complet dans lequel nous créerons un blog.

Nous utiliserons également les éléments suivants :

### MySQL et PDO



Toutes les applications backend, comme celles créées avec PHP, nécessitent d'utiliser une base de données. En PHP, on utilise le plus souvent des bases de données SQL. La base de données la plus utilisée est MySQL. Durant la formation vous apprendrez toutes les bases du SQL et vous apprendrez à faire communiquer vos applications PHP avec MySQL.

### Nginx



Pour que vos utilisateurs puissent utiliser votre application PHP, il faut pouvoir leur délivrer via un serveur Web. Le plus utilisé aujourd'hui et le plus performant est NGINX. C'est pourquoi durant la formation nous vous montrerons comment mettre en place une architecture moderne avec Nginx et PHP-FPM pour donner à vos utilisateurs la meilleure expérience possible avec des vitesses de chargement extrêmement rapides.

### Programmation orientée objet



La programmation orientée objet est permise en PHP grâce à l'utilisation d'un système de classe. Les classes permettent de regrouper des fonctionnalités pour permettre une meilleure lisibilité de votre code. Dans la formation vous apprendrez toutes les bases de la POO et comment la mettre en pratique.

## Description

Le PHP est le langage backend le plus utilisé au monde pour la réalisation d'applications Web.

Le langage s'utilise en complément de HTML, CSS ainsi que de JavaScript.

Le rôle du PHP est de vous permettre de créer toute la partie backend de vos applications, c'est grâce à lui que vous allez pouvoir gérer l'authentification de vos utilisateurs et sauvegarder des données dans une base de données.

80% des sites Web actuels ont été réalisés avec PHP.

C'est un langage facile d'accès mais avec beaucoup de profondeur qui prendra des années à maîtriser parfaitement.

Vous serez à jour de toutes les nouveautés qui se sont ajoutées au fil des années (PHP 5, PHP 6, PHP 7 et PHP 8) et qui font du PHP un langage riche et performant.

PHP est un langage ancien mais qui évolue avec son temps. Vous apprendrez toutes les syntaxes récentes du langage apparues avec la version 8.

Le PHP permet également de faire de la programmation orientée objet avec un système de classes poussé qui vous permettra de réaliser des applications complexes.

Apprendre le langage PHP vous sera très utile pour l'apprentissage de frameworks basés sur PHP comme Symfony ou encore Laravel.

PHP est beaucoup utilisé en entreprise et très souvent il est même obligatoire de le maîtriser. Le connaître vous donnera de l'assurance dans votre métier de développeur Web.

# Plan du module

## Pour débuté

- Chapitre 1 : Introduction au développement Web
- Chapitre 2 : Bases de PHP
- Chapitre 3 : Manipulation des données
- Chapitre 4 : Bases de données avec PHP
- Chapitre 5 : Gestion des sessions et cookies

## Thèmes avancés

- Chapitre 6 : PHP avancé
- Chapitre 7 : Travail avec les frameworks PHP
- Chapitre 8 : Projet final

## Ressources Recommandées

- Documentation PHP officielle : [php.net](https://www.php.net)
- Laravel : [laravel.com](https://laravel.com)
- Tutoriels vidéos : FreeCodeCamp, OpenClassrooms, YouTube
- Livres : PHP & MySQL: Server-side Web Development de Jon Duckett

**LET' S**

**GO**

# **PHP**

**Introduction au développement Web**

## Comprendre le Web

***Dans cette partie, nous allons explorer les fondamentaux du fonctionnement du web. Cela inclut les protocoles, les rôles des serveurs et des clients, ainsi que les types de contenus échangés***

## Comprendre le Web

### Le fonctionnement de base du web

Le web repose sur l'interaction entre un **client** (par exemple, un navigateur) et un **serveur** (par exemple, un serveur Apache). Cette interaction se fait via le **protocole HTTP** (Hypertext Transfer Protocol).

Étapes d'une interaction typique :

1. **Requête HTTP** :

- Le client envoie une requête au serveur pour demander une ressource (une page web, une image, etc.).
- Exemple de requête : GET /index.html HTTP/1.1.

2. **Réponse HTTP** :

- Le serveur traite la requête et retourne une réponse contenant les données demandées (HTML, JSON, etc.) et un code de statut (par exemple, 200 OK pour indiquer que la requête a été satisfaite).

3. **Rendu** :

- Le navigateur interprète la réponse (souvent en HTML) et affiche la page web.

Types de requêtes HTTP :

- **GET** : Demande de récupérer une ressource (par exemple, afficher une page web).
- **POST** : Envoi de données au serveur (par exemple, soumettre un formulaire).
- **PUT** : Mise à jour d'une ressource existante.
- **DELETE** : Suppression d'une ressource.

# Comprendre le Web

## Rôles des Clients et des Serveurs

- **Client :**
  - Logiciel qui fait une demande de ressource au serveur.
  - Les clients courants incluent :
    - Les navigateurs web (Chrome, Firefox).
    - Les applications mobiles et de bureau.
- **Serveur :**
  - Logiciel (ou matériel) qui reçoit, traite et répond aux demandes du client.
  - Exemples de serveurs :
    - Serveur web : Apache, Nginx.
    - Serveur d'application : Node.js, PHP.
    - Serveur de base de données : MySQL, PostgreSQL.
- **Schéma d'Interaction Client-Serveur :**

```
Client (navigateur) -----> Requête HTTP -----> Serveur Web  
----- Réponse HTTP (HTML, CSS, JS) -----
```

# Comprendre le Web

## Types de contenus échangés

Le contenu échangé entre le client et le serveur peut inclure :

- **HTML** : Structure de la page web.
- **CSS** : Apparence (couleurs, polices).
- **JavaScript** : Interaction et dynamisme.
- **Images** : Fichiers (PNG, JPEG, SVG).
- **Données** : JSON, XML pour les API.

# Comprendre le Web

## HTTP et HTTPS

- **HTTP (HyperText Transfer Protocol) :**
  - Protocol de communication non sécurisé.
  - Les données circulent en clair.
- **HTTPS (HTTP Secure) :**
  - HTTP + chiffrement SSL/TLS.
  - Sécurise la communication et protège contre les attaques de type écoute.
- **Exemple de différence :**

HTTP : `http://example.com`

HTTPS : `https://example.com`

# Comprendre le Web

## Outils et pratiques

- **Navigateur Web :**

- Les navigateurs modernes comme **Chrome**, Firefox et **Edge** possèdent des outils intégrés pour inspecter les pages web.
  - Appuyez sur **F12** ou faites un **clic droit > Inspecter**.

- **Serveurs locaux :**

- Pour apprendre PHP, vous pouvez utiliser un serveur local :
- XAMPP ou WAMP : Fournissent PHP, MySQL et Apache/Nginx.

# Comprendre le Web

## Exemples pratiques

### 1. Premier site HTML

Créez un fichier nommé **index.html** :

```
<!DOCTYPE html>
<html>
<head>
    <title>Mon Premier Site</title>
</head>
<body>
    <h1>Bienvenue sur mon site</h1>
    <p>Ce site est généré grâce à HTML.</p>
</body>
</html>
```

# Comprendre le Web

## Exemples pratiques

### 2. Ajout de PHP

Créez un fichier nommé **index.php** dans un serveur local :

```
<?php  
echo "<h1>Bienvenue sur mon site en PHP</h1>";  
?>
```

### 3. Test des requêtes

Utilisez un outil comme Postman ou curl pour envoyer des requêtes à un serveur local.

Exemple avec curl dans le terminal

```
curl -X GET http://localhost/index.php
```

# Comprendre le Web

## Exercice

1. Expliquez en vos mots ce qu'est un protocole HTTP.
2. Identifiez les composants suivants dans une URL :

```
https://www.example.com:8080/page?param=valeur#section
```

- **Protocole** : https
- **Nom de domaine** : www.example.com
- **Port** : 8080
- **Chemin** : /page
- **Paramètres** : param=valeur
- **Fragment** : #section

1. Créez un fichier PHP qui affiche la date et l'heure actuelles :

```
<?php  
echo "La date actuelle est : " . date("d/m/Y H:i:s");  
?>
```

# Comprendre le Web

## Quiz rapide

1. Que signifie HTTP ?

- a) HyperText Transfer Protocol
- b) HyperText Translation Protocol
- c) Hyper Transfer Text Protocol

2. Quelle est la principale différence entre HTTP et HTTPS ?

- a) HTTPS est plus rapide.
- b) HTTPS est chiffré.
- c) HTTPS utilise un autre langage.

## Les outils essentiels

*Pour développer en PHP et créer des applications web, vous aurez besoin de plusieurs outils.  
Ces outils vous permettront d'écrire, tester et déployer votre code efficacement.*

## Les outils essentiels

### Serveurs locaux

Un serveur local permet de simuler un environnement serveur sur votre machine. Ces serveurs incluent un interpréteur PHP, une base de données, et un serveur HTTP (comme Apache ou Nginx).

#### Outils recommandés :

- **XAMPP :**
  - Disponible sur Windows, macOS et Linux.
  - Inclut Apache, MySQL, PHP, et PhpMyAdmin.
  - [Télécharger XAMPP](#).
- **WAMP (Windows) :**
  - Similaire à XAMPP, mais spécifique à Windows.
  - Inclut Apache, MySQL et PHP.
  - [Télécharger WAMP](#).
- **MAMP (macOS et Windows) :**
  - Spécialement conçu pour macOS et Windows.
  - Inclut Apache, Nginx, MySQL et PHP.
  - [Télécharger MAMP](#).
- **Laragon :**
  - Léger et rapide pour Windows.
  - Permet de créer des environnements de développement modulaires.
  - [Télécharger Laragon](#).

#### Installation de XAMPP :

1. Téléchargez et installez XAMPP depuis le site officiel.
2. Lancez XAMPP et démarrez les modules Apache et MySQL.
3. Placez vos fichiers PHP dans le dossier **htdocs** (généralement situé dans **C:\xampp\htdocs**).
4. Accédez à vos fichiers via le navigateur en entrant [http://localhost/nom\\_du\\_fichier.php](http://localhost/nom_du_fichier.php).

## Les outils essentiels

### Éditeurs de texte et environnements de développement

Un bon éditeur de texte rend le développement plus fluide grâce à des fonctionnalités comme la coloration syntaxique, l'autocomplétion, et l'intégration avec des outils tiers.

#### Éditeurs de texte recommandés:

- **Visual Studio Code :**
  - Gratuit, léger et extensible avec des extensions.
  - Extensions utiles pour PHP :
  - PHP IntelliSense : Autocomplétion et documentation.
  - PHP Intelephense : Analyse avancée de code.
  - Xdebug : Pour le débogage.
  - [Télécharger VS Code.](#)
- **Sublime Text :**
  - Éditeur léger et rapide.
  - Support des extensions via Package Control.
  - [Télécharger Sublime Text.](#)
- **PhpStorm :**
  - IDE complet pour PHP.
  - Support avancé de PHP, bases de données et frameworks.
  - Payant, mais offre une version d'essai.
  - [Télécharger PhpStorm.](#)

#### Configurer Visual Studio Code pour PHP :

1. Installez PHP sur votre système :
2. Télécharger PHP.
3. Configurez la variable d'environnement pour que PHP soit accessible via le terminal.
4. Installez les extensions mentionnées ci-dessus dans VS Code.
5. Configurez un débogueur PHP avec Xdebug.

## Les outils essentiels

### Navigateur Web et outils de développement

Un navigateur web est essentiel pour tester vos applications. Les navigateurs modernes offrent des outils de développement pour déboguer, analyser et optimiser vos pages web.

**Navigateurs recommandés :**

- **Google Chrome :**
  - Rapide et fiable.
  - Inclut les DevTools (F12).
  - Extensions utiles :
    - ✓ Postman : Tester les API.
    - ✓ JSON Viewer : Visualiser les réponses JSON.
- **Mozilla Firefox :**
  - Open source et axé sur la confidentialité.
  - Les outils de développement sont puissants, surtout pour CSS et JavaScript.
- **Microsoft Edge :**
  - Basé sur Chromium, offre des DevTools similaires à Chrome.

# Les outils essentiels

## Outils pour la gestion des bases de données

Pour manipuler des bases de données comme MySQL, il existe des outils graphiques qui simplifient la gestion.

Outils recommandés :

- **PhpMyAdmin :**
  - Inclus avec XAMPP/WAMP.
  - Permet de gérer des bases de données MySQL via une interface web.
  - URL par défaut : <http://localhost/phpmyadmin>.
- **Adminer :**
  - Plus léger que PhpMyAdmin.
  - [Télécharger Adminer](#).
- **MySQL Workbench :**
  - Outil avancé pour concevoir et administrer des bases de données.
  - [Télécharger MySQL Workbench](#).

# Les outils essentiels

## Contrôle de version avec Git

Git est un système de contrôle de version qui permet de suivre les modifications de votre code.

**Installation de Git :**

Téléchargez et installez Git depuis [git-scm.com](https://git-scm.com).

**Configurez Git :**

```
git config --global user.name "VotreNom"  
git config --global user.email "VotreEmail"
```

**Utilisation Basique de Git :**

Initialisez un dépôt :

```
git init
```

Ajoutez des fichiers au suivi :

```
git add fichier.php
```

Faites un commit :

```
git commit -m "Premier commit"
```

Connectez-vous à un dépôt distant (GitHub, GitLab, etc.) :

```
git remote add origin URL_DU_DEPOT  
git push -u origin master
```

# Les outils essentiels

## Exemples pratiques

1. Installez XAMPP
2. Créez un fichier nommé **test.php** dans le dossier **htdocs** :

```
<?php  
echo "Serveur local opérationnel!";  
?>
```

3. Ouvrez **http://localhost/test.php** dans un navigateur. Vous devriez voir le message "**Serveur local opérationnel!**".

## Les outils essentiels

### Quiz

1. Quel outil est inclus avec XAMPP pour gérer les bases de données ?

- a) MySQL Workbench
- b) PhpMyAdmin
- c) HeidiSQL

2. Quelle commande Git est utilisée pour enregistrer des modifications localement ?

- a) git push
- b) git commit
- c) git clone

# **PHP**

**Bases de PHP**

## Qu'est-ce que PHP ?

***PHP (Hypertext Preprocessor) est un langage de programmation côté serveur conçu pour créer des pages web dynamiques. Il est particulièrement apprécié pour sa simplicité, sa flexibilité, et sa compatibilité avec la majorité des systèmes d'exploitation et serveurs web.***

# Qu'est-ce que PHP ?

## Historique de PHP

- **Création** : PHP a été développé en 1994 par Rasmus Lerdorf pour gérer son site personnel.
- **Évolution** : Initialement appelé "Personal Home Page Tools", PHP est devenu un langage complet avec l'introduction de PHP 3.0 en 1998.
- **Version actuelle** : La dernière version de PHP 8. x est PHP 8.3 , avec PHP 8.4 prévu pour une sortie en novembre 2024.

## Définition de PHP

- PHP est un langage de script interprété qui s'exécute sur le serveur. Contrairement à JavaScript, qui s'exécute côté client, PHP génère du contenu (HTML, JSON, XML, etc.) envoyé au client (navigateur).

# Qu'est-ce que PHP ?

## Caractéristiques principales

### Dynamisme :

PHP peut générer des pages web personnalisées selon les données saisies par l'utilisateur, les informations de la base de données ou d'autres facteurs.

### Exemple :

```
<?php  
$nom = "Mahamane";  
echo "Bonjour, $nom ! Bienvenue sur notre site.";  
?>
```

# Qu'est-ce que PHP ?

## Caractéristiques principales

### Intégration facile avec HTML :

PHP s'insère directement dans du code HTML, ce qui facilite la création de pages dynamiques.

Exemple :

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemple PHP</title>
</head>
<body>
    <h1><?php echo "Bonjour, le monde !"; ?></h1>
</body>
</html>
```

# Qu'est-ce que PHP ?

## Caractéristiques principales

### Compatibilité avec les bases de données:

PHP prend en charge une large gamme de bases de données, notamment :

MySQL (le plus couramment utilisé).  
PostgreSQL, SQLite, MongoDB, etc.

# Qu'est-ce que PHP ?

## Avantages de PHP

- **Open Source :**
  - Gratuit et soutenu par une grande communauté.
  - Facilité d'apprentissage :
  - Syntaxe simple, idéal pour les débutants.
- **Multiplateforme :**
  - Fonctionne sur Windows, Linux, macOS, etc.
- **Performances :**
  - Capable de gérer des projets à grande échelle comme Facebook (PHP modifié via HHVM).
- **Large Écosystème :**
  - Cadres (frameworks) comme Laravel, Symfony.
  - CMS comme WordPress, Joomla, Drupal.

# Qu'est-ce que PHP ?

## Les limites de PHP

- **Performance côté serveur :**
  - Moins rapide que des langages compilés comme C++.
  - Cependant, PHP 7 et PHP 8 ont grandement amélioré cet aspect.
- **Sécurité :**
  - Les mauvaises pratiques (par exemple, ne pas valider les entrées utilisateur) peuvent mener à des vulnérabilités comme l'injection SQL.

## Fonctionnement de PHP

Lorsque vous accédez à une page .php via un navigateur :

- Le serveur web (Apache, Nginx) identifie le fichier comme un script PHP.
- L'interpréteur PHP traite le fichier.
- Le serveur envoie une réponse au navigateur sous forme de HTML.

# Qu'est-ce que PHP ?

## Les applications courantes de PHP

- **Sites dynamiques :**
  - Génération de pages personnalisées selon les utilisateurs.
- **Gestion des bases de données :**
  - Affichage et manipulation des données stockées dans MySQL.
- **API Web :**
  - PHP peut servir à créer des API RESTful qui renvoient des données JSON.
- **CMS (Systèmes de Gestion de Contenu) :**
  - PHP est le moteur derrière des systèmes comme WordPress.
- **Applications Web Complexes :**
  - E-commerce (Magento, WooCommerce).
  - Réseaux sociaux (Facebook à ses débuts).

# Qu'est-ce que PHP ?

## Exercice : Tester PHP

- **But : Créez un fichier PHP qui affiche une phrase dynamique basée sur l'heure actuelle.**

Code suggéré :

```
<?php
$heure = date("H");
if ($heure < 12) {
    echo "Bonjour, il est matin !";
} elseif ($heure < 18) {
    echo "Bon après-midi !";
} else {
    echo "Bonsoir !";
}
?>
```

Instructions :

- Placez le fichier dans votre serveur local.
- Accédez-y via le navigateur et observez les résultats à différents moments de la journée.

# Qu'est-ce que PHP ?

## Quiz

- **Que signifie l'acronyme PHP ?**
  - a) Personal Home Page
  - b) Professional Hypertext Preprocessor
  - c) Public HTML Parser
  
- **PHP est-il un langage côté serveur ou côté client ?**
  - a) Serveur
  - b) Client
  - c) Les deux

## Structure du code

*PHP possède une structure de code simple et facile à apprendre, ce qui en fait un excellent langage pour les débutants et les développeurs expérimentés. Cette section explique comment organiser et structurer le code PHP, avec des exemples pratiques.*

# Structure du code

## Les balises PHP

Le code PHP est encapsulé dans des balises spéciales :

```
<?php ... ?>
```

Exemple :

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

Autres styles de balises (moins courants) :

- Balises courtes : <? ... ?>  
*Remarque : Elles doivent être activées dans le fichier de configuration PHP.*
- Balises ASP : <% ... %>  
*Obsolètes et non recommandées.*

## Structure du code

### Organisation de base d'un script PHP

Un script PHP peut inclure :

1. Déclarations de variables
2. Structures conditionnelles
3. Boucles
4. Fonctions
5. Interactions avec une base de données

# Structure du code

## Les commentaires

Les commentaires sont essentiels pour rendre le code lisible et documenté.

Types de commentaires :

### Commentaire sur une ligne :

```
// Ceci est un commentaire
```

### Commentaire sur une ligne avec dièse :

```
# Ceci est aussi un commentaire
```

### Commentaire multi-lignes :

```
/*
  Ceci est un commentaire
  sur plusieurs lignes
*/
```

# Structure du code

## Les commentaires

Exemple pratique :

```
<?php
// Déclarer une variable
$nom = "Mahamane";

/*
   Afficher le contenu
   de la variable $nom
*/
echo "Bonjour, $nom !";
?>
```

# Structure du code

## Variables et constantes

### 1. Variables

- Les variables commencent par un signe \$ suivi d'un nom.
- Sensible à la casse (\$Nom et \$nom sont différents).
- Types dynamiques (pas besoin de déclarer leur type).

Exemple :

```
<?php
$nom = "Ali";
$age = 25;

echo "Nom : $nom, Âge : $age";
?>
```

# Structure du code

## Variables et constantes

### 2. Constantes

- Les constantes ne changent jamais de valeur après leur définition.
- Déclarées avec **define()** ou **const**.

Exemple :

```
<?php
define("SITE_WEB", "example.com");
echo "Bienvenue sur " . SITE_WEB;

// Avec const
const VILLE = "Niamey";
echo "Vous êtes à " . VILLE;
?>
```

# Structure du code

## Les structures de contrôle

### 1. Conditionnelles

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

Syntaxe **if**:

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

# Structure du code

## Les structures de contrôle

Syntaxe **switch** :

```
<?php  
$jour = "Lundi";  
  
switch ($jour) {  
    case "Lundi":  
        echo "Début de semaine";  
        break;  
    case "Vendredi":  
        echo "Fin de semaine";  
        break;  
    default:  
        echo "Jour ordinaire";  
}  
?>
```

# Structure du code

## Les structures de contrôle

### 2. Boucles

Boucle **for** :

```
<?php
for ($i = 1; $i <= 5; $i++) {
    echo "Nombre : $i <br>";
}
?>
```

Boucle **while** :

```
<?php
$i = 1;
while ($i <= 5) {
    echo "Nombre : $i <br>";
    $i++;
}
?>
```

# Structure du code

## Les structures de contrôle

### 2. Boucles

Boucle **foreach** :

```
<?php
$fruits = ["Pomme", "Banane", "Orange"];
foreach ($fruits as $fruit) {
    echo "Fruit : $fruit <br>";
}
?>
```

# Fonctions et inclusion de fichiers

*Les fonctions permettent de structurer le code en blocs réutilisables.*

# Fonctions et inclusion de fichiers

## Définir et appeler une fonction

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

## Fonctions avec paramètres par défaut

```
<?php  
function addition($a = 5, $b = 10) {  
    return $a + $b;  
}  
  
echo addition();          // Affiche 15  
echo addition(20, 30);   // Affiche 50  
?>
```

# Fonctions et inclusion de fichiers

*Pour organiser le code, PHP permet d'inclure d'autres fichiers..*

# Fonctions et inclusion de fichiers

## include

Génère un **avertissement** si le fichier n'est pas trouvé.

```
<?php  
include "header.php";  
?>
```

## Require

Génère une **erreur fatale** si le fichier n'est pas trouvé

```
<?php  
require "config.php";  
?>
```

## include\_once / require\_once

Garantit que le fichier n'est inclus qu'une seule fois.

```
<?php  
include_once "header.php";  
?>
```

# Fonctions et inclusion de fichiers

## Organisation du code en PHP moderne

- Séparation du code
  - Code PHP dans des fichiers .php.
  - HTML dans des fichiers séparés ou intégrés via des templates.
- Respect des standards PSR (PHP Standards Recommendations)
  - PSR-1 : Standards de base pour le code PHP.
  - PSR-4 : Chargement automatique des classes.
- Utilisation des frameworks
  - Pour les projets complexes, utilisez des frameworks comme Laravel ou Symfony pour une meilleure structure.

# Fonctions et inclusion de fichiers

## Exemple pratique : Application simple

- But :
  - Créer une page affichant un message en fonction de l'heure.
- Code :
  - Fichier **functions.php** :

```
<?php
function salutation() {
    $heure = date("H");
    if ($heure < 12) {
        return "Bonjour !";
    } elseif ($heure < 18) {
        return "Bon après-midi !";
    } else {
        return "Bonsoir !";
    }
?>
```

- Fichier **index.php** :

```
<?php
include "functions.php";
?>
<!DOCTYPE html>
<html>
<head>
    <title>Page PHP Structurée</title>
</head>
<body>
    <h1><?php echo salutation(); ?></h1>
</body>
</html>
```

# Fonctions et inclusion de fichiers

## Exercice

- Créez une fonction qui prend deux nombres et retourne leur produit. Testez-la.
- Utilisez une boucle pour afficher les 10 premières tables de multiplication.
- Divisez le code suivant en deux fichiers : un pour les fonctions et un autre pour l'affichage.

# Fonctions et inclusion de fichiers

## Quiz

- Quelle fonction est utilisée pour inclure un fichier tout en générant une erreur fatale s'il est absent ?
  - include
  - require
  - require\_once
- Quelle syntaxe est correcte pour déclarer une constante en PHP ?
  - const NOM = "Valeur";
  - \$NOM = "Valeur";
  - constant("NOM", "Valeur");

# **PHP**

## **Manipulation des données**

# Formulaires HTML et PHP

***Les formulaires sont essentiels pour interagir avec les utilisateurs. Ils permettent de collecter des données que PHP peut traiter. Cette section explique comment créer des formulaires HTML, récupérer les données en PHP, et sécuriser leur traitement.***

# Formulaires HTML et PHP

## Création d'un Formulaire HTML

Un formulaire HTML contient différents champs (texte, bouton, case à cocher, etc.) pour capturer les données de l'utilisateur. Ces données sont envoyées au serveur à l'aide des méthodes **GET** ou **POST**.

### Exemple de formulaire simple

```
<!DOCTYPE html>
<html>
<head>
    <title>Formulaire Exemple</title>
</head>
<body>
    <form action="traitement.php" method="POST">
        <label for="nom">Nom :</label>
        <input type="text" id="nom" name="nom" required>
        <br><br>
        <label for="email">Email :</label>
        <input type="email" id="email" name="email" required>
        <br><br>
        <button type="submit">Envoyer</button>
    </form>
</body>
</html>
```

**action** : URL ou fichier où les données seront envoyées.

**method** : Méthode HTTP utilisée pour envoyer les données (POST ou GET).

# Formulaires HTML et PHP

## Méthodes GET et POST

### Méthode GET

- Envoie les données dans l'URL.
- Utilisée pour des recherches ou des données non sensibles.
- Limitation de taille (environ 2000 caractères).

Exemple :

Formulaire avec method="GET" :

```
<form action="traitement.php" method="GET">
    <label for="recherche">Recherche :</label>
    <input type="text" id="recherche" name="recherche">
    <button type="submit">Rechercher</button>
</form>
```

Résultat dans l'URL :

<http://localhost/traitement.php?recherche=motclé>

# Formulaires HTML et PHP

## Méthodes GET et POST

### Méthode POST

- Envoie les données dans le corps de la requête HTTP.
- Plus sécurisé que GET (les données ne sont pas visibles dans l'URL).
- Pas de limite de taille pour les données.

Exemple :

Formulaire avec method= »POST» :

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

# Formulaires HTML et PHP

## Récupérer les données en PHP

PHP fournit des superglobales pour accéder aux données des formulaires :

- **`$_GET`** : Données envoyées par la méthode GET.
- **`$_POST`** : Données envoyées par la méthode POST.
- **`$_REQUEST`** : Contient à la fois les données GET et POST.

Exemple :

Fichier **traitement.php** :

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $nom = $_POST['nom'];
    $email = $_POST['email'];

    echo "Nom : " . htmlspecialchars($nom) . "<br>";
    echo "Email : " . htmlspecialchars($email);
}
?>
```

**`$_SERVER['REQUEST_METHOD']`** : Vérifie si le formulaire a été soumis avec POST.

**`htmlspecialchars()`** : Protège contre les injections HTML.

# Formulaires HTML et PHP

## Validation et sécurisation des formulaires

### 1. Validation côté serveur

Assurez-vous que les données reçues sont valides avant de les traiter.

Exemple :

```
<?php  
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    $nom = trim($_POST['nom']);  
    $email = trim($_POST['email']);  
  
    if (empty($nom)) {  
        echo "Le champ Nom est requis.";  
    } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {  
        echo "L'email n'est pas valide.";  
    } else {  
        echo "Données valides. Merci, $nom !";  
    }  
}  
?>
```

# Formulaires HTML et PHP

## Validation et sécurisation des formulaires

### 2. Sécurisation

Nettoyage des données :

```
$nom = htmlspecialchars(strip_tags($nom));
```

Échapper les caractères spéciaux avant d'insérer dans une base de données :

```
$nom = mysqli_real_escape_string($connexion, $nom);
```

# Formulaires HTML et PHP

## Utilisation avancée : envoi et traitement

### Formulaire avec plusieurs champs

```
<form action="traitement.php" method="POST">
    <label for="nom">Nom :</label>
    <input type="text" id="nom" name="nom" required>
    <br><br>
    <label for="email">Email :</label>
    <input type="email" id="email" name="email" required>
    <br><br>
    <label for="message">Message :</label>
    <textarea id="message" name="message"></textarea>
    <br><br>
    <label>
        <input type="checkbox" name="newsLetter" value="oui"> S'inscrire à la news
    </label>
    <br><br>
    <button type="submit">Envoyer</button>
</form>
```

# Formulaires HTML et PHP

## Utilisation avancée : envoi et traitement

### Traitement des données

```
<?php  
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    $nom = htmlspecialchars($_POST['nom']);  
    $email = htmlspecialchars($_POST['email']);  
    $message = htmlspecialchars($_POST['message']);  
    $newsletter = isset($_POST['newsletter']) ? 'Oui' : 'Non';  
  
    echo "Nom : $nom<br>";  
    echo "Email : $email<br>";  
    echo "Message : $message<br>";  
    echo "Inscription à la newsletter : $newsletter";  
}  
?>
```

# Formulaires HTML et PHP

## Téléchargement de fichiers

PHP permet aussi de gérer les fichiers envoyés via un formulaire.

### Formulaire pour envoyer un fichier

```
<form action="upload.php" method="POST" enctype="multipart/form-data">
    <label for="fichier">Choisir un fichier :</label>
    <input type="file" id="fichier" name="fichier">
    <button type="submit">Télécharger</button>
</form>
```

### Traitement du fichier

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_FILES['fichier'])) {
    $nomFichier = $_FILES['fichier']['name'];
    $cheminTemp = $_FILES['fichier']['tmp_name'];
    $dossierDestination = "uploads/";

    if (move_uploaded_file($cheminTemp, $dossierDestination . $nomFichier)) {
        echo "Fichier téléchargé avec succès : $nomFichier";
    } else {
        echo "Échec du téléchargement.";
    }
}
?>
```

# Formulaires HTML et PHP

## Exercice pratique

### Formulaire de Contact

- Créez un formulaire avec les champs suivants :

- Nom
- Email
- Sujet
- Message

- Traitez les données en PHP :

- Validez les champs (tous obligatoires).
- Affichez un message de confirmation.

# Formulaires HTML et PHP

## Quiz

- Quelle méthode d'envoi est utilisée pour inclure les données dans l'URL ?
  - GET
  - POST
  - PUT
- Quelle fonction PHP est utilisée pour protéger contre les injections HTML ?
  - trim()
  - htmlspecialchars()
  - mysqli\_escape\_string()

## Gestion des variables globales

*En PHP, les variables globales sont des variables qui peuvent être accessibles depuis n'importe quel point du script, y compris au sein des fonctions ou des classes. PHP propose plusieurs superglobales pour gérer différentes informations, telles que les données utilisateur, les configurations serveur, ou les sessions.*

# Gestion des variables globales

## Les Superglobales PHP

Les superglobales sont des tableaux prédéfinis accessibles partout dans un script PHP. Elles contiennent des données liées à l'environnement d'exécution, aux requêtes HTTP, et aux sessions.

Liste des principales superglobales :

Superglobale	Description
<code>\$_GET</code>	Données envoyées via la méthode HTTP GET.
<code>\$_POST</code>	Données envoyées via la méthode HTTP POST.
<code>\$_REQUEST</code>	Combine les données <code>\$_GET</code> , <code>\$_POST</code> , et <code>\$_COOKIE</code> .
<code>\$_FILES</code>	Données sur les fichiers uploadés via un formulaire HTML.
<code>\$_SESSION</code>	Variables de session de l'utilisateur actuel.
<code>\$_COOKIE</code>	Données stockées dans les cookies HTTP du client.
<code>\$_SERVER</code>	Informations sur le serveur et l'environnement d'exécution.
<code>\$_ENV</code>	Variables d'environnement du système d'exploitation.
<code>\$_GLOBALS</code>	Toutes les variables globales disponibles dans le contexte courant.

# Gestion des variables globales

## Utilisation des Superglobales

### 1 accéder aux données utilisateur

Exemple avec \$\_GET :

```
<?php  
// URL : http://example.com/page.php?nom=Ali&age=30  
$nom = $_GET['nom'];  
$age = $_GET['age'];  
  
echo "Nom : $nom<br>";  
echo "Âge : $age";  
?>
```

Exemple avec \$\_POST :

```
<?php  
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    $email = $_POST['email'];  
    echo "Email envoyé : $email";  
}  
?>
```

# Gestion des variables globales

## Utilisation des Superglobales

### 1 Gérer les cookies

Définir un cookie :

```
<?php  
setcookie("utilisateur", "Ali", time() + 3600); // Cookie valide pour 1 heure  
?>
```

Lire un cookie :

```
<?php  
if (isset($_COOKIE['utilisateur'])) {  
    echo "Utilisateur : " . $_COOKIE['utilisateur'];  
}  
?>
```

# Gestion des variables globales

## La Superglobale `$_SERVER`

`$_SERVER` fournit des informations sur le serveur, le script en cours d'exécution, et la requête HTTP.

Exemples courants :

Clé	Description
<code>\$_SERVER['PHP_SELF']</code>	Chemin du fichier PHP courant.
<code>\$_SERVER['SERVER_NAME']</code>	Nom de domaine ou adresse IP du serveur.
<code>\$_SERVER['HTTP_USER_AGENT']</code>	Informations sur le navigateur client.
<code>\$_SERVER['REMOTE_ADDR']</code>	Adresse IP du client.

Exemple :

```
<?php
echo "Nom du serveur : " . $_SERVER['SERVER_NAME'] . "<br>";
echo "Fichier en cours : " . $_SERVER['PHP_SELF'] . "<br>";
echo "Adresse IP du client : " . $_SERVER['REMOTE_ADDR'];
?>
```

# Gestion des variables globales

## La Superglobale `$_GLOBALS`

`$_GLOBALS` est un tableau associatif contenant toutes les variables globales définies dans le script.

Exemples :

```
<?php  
$nom = "Mahamane";  
  
function afficherNom() {  
    global $nom; // Accès direct  
    echo "Nom (global) : " . $nom . "<br>";  
  
    // Accès via $_GLOBALS  
    echo "Nom (via \$_GLOBALS) : " . $_GLOBALS['nom'];  
}  
  
afficherNom();  
?>
```

# Gestion des variables globales

## Les Sessions avec \$\_SESSION

### 1 Initialisation des sessions

Pour utiliser les sessions, démarrez-les avec **session\_start()**.

Exemples :

```
<?php
session_start();
$_SESSION['nom'] = "Ali";
$_SESSION['age'] = 30;

echo "Session créée.";
?>
```

# Gestion des variables globales

## Les Sessions avec \$\_SESSION

### 2 Accéder aux variables de session

```
<?php
session_start();
if (isset($_SESSION['nom'])) {
    echo "Nom : " . $_SESSION['nom'] . "<br>";
    echo "Âge : " . $_SESSION['age'];
}
?>
```

### 3 Supprimer une variable de session

```
<?php
unset($_SESSION['nom']);
```

### 4 Détruire une session

```
<?php
session_start();
session_destroy(); // Supprime toutes les données de session
?>
```

# Gestion des variables globales

## Bonnes pratiques avec les variables globales

### 1 Validation des entrées :

Nettoyez et validez toujours les données reçues via les superglobales pour éviter les failles comme les injections SQL ou XSS.

```
$nom = htmlspecialchars($_POST['nom']);
```

### 2 Limitez l'utilisation de \$\_GLOBALS :

Préférez passer des arguments dans les fonctions ou utiliser des objets pour éviter les erreurs liées à des variables globales.

### 3 Utilisez des Sessions sécurisées :

Configurez un identifiant de session sécurisé :

```
ini_set('session.cookie_httponly', 1);
ini_set('session.cookie_secure', 1);
ini_set('session.use_only_cookies', 1);
```

# Gestion des variables globales

## Exercice pratique

### Manipulation de \$\_GET et \$\_POST :

Créez une page avec :

- Un formulaire pour rechercher un mot via GET.
- Un autre formulaire pour soumettre des données via POST.

### Gestion des Sessions :

- Créez une session pour enregistrer un nom d'utilisateur.
- Affichez le nom sur une autre page.

## Gestion des variables globales

### Quiz

Quelle superglobale contient les informations sur les cookies ?

- a) \$\_SESSION
- b) \$\_COOKIE
- c) \$\_REQUEST

Quelle fonction supprime toutes les données de session ?

- a) session\_unset()
- b) session\_destroy()
- c) unset(\$\_SESSION)

## Gestion des fichiers

***La gestion des fichiers est une compétence essentielle en PHP, permettant la lecture, l'écriture, la création, la suppression et la manipulation de fichiers sur le système. Cette section détaille les fonctions PHP pour travailler avec des fichiers de manière efficace et sécurisée.***

# Gestion des fichiers

## Ouverture et fermeture de fichiers

### Fonction fopen()

La fonction fopen() ouvre un fichier et retourne un pointeur à utiliser pour les opérations de lecture/écriture.

Syntaxe :

```
fopen($nom_fichier, $mode);
```

### Modes courants :

Mode	Description
'r'	Lecture seule, commence au début du fichier.
'w'	Écriture seule, écrase le contenu existant ou crée un nouveau fichier.
'a'	Écriture seule, ajoute les données à la fin du fichier ou crée un fichier.
'r+'	Lecture et écriture, commence au début du fichier.
'w+'	Lecture et écriture, écrase le contenu existant ou crée un nouveau fichier.
'a+'	Lecture et écriture, ajoute les données à la fin du fichier.

# Gestion des fichiers

## Ouverture et fermeture de fichiers

### Exemple : Ouverture et Fermeture

```
<?php  
$fichier = fopen("exemple.txt", "r");  
  
if ($fichier) {  
    echo "Fichier ouvert avec succès.";  
    fclose($fichier); // Toujours fermer le fichier après usage  
} else {  
    echo "Impossible d'ouvrir le fichier.";  
}  
?>
```

# Gestion des fichiers

## Lecture de fichiers

Lecture ligne par ligne avec **fgets** :

```
<?php
$fichier = fopen("exemple.txt", "r");
if ($fichier) {
    while (($ligne = fgets($fichier)) !== false) {
        echo $ligne . "<br>";
    }
    fclose($fichier);
}
?>
```

Lecture complète avec **file\_get\_contents** :

```
<?php
$contenu = file_get_contents("exemple.txt");
echo nl2br($contenu); // nl2br pour convertir les sauts de ligne en HTML
?>
```

# Gestion des fichiers

## Écriture dans un fichier

### Écrire ou remplacer avec **fwrite**

```
<?php
$fichier = fopen("exemple.txt", "w");
if ($fichier) {
    fwrite($fichier, "Nouvelle ligne écrite dans le fichier.\n");
    fclose($fichier);
    echo "Écriture réussie.";
}
?>
```

### Ajouter du contenu avec **a**

```
<?php
$fichier = fopen("exemple.txt", "a");
if ($fichier) {
    fwrite($fichier, "Ajout d'une ligne supplémentaire.\n");
    fclose($fichier);
    echo "Contenu ajouté.";
}
?>
```

# Gestion des fichiers

## Manipulation de fichiers

### 1 Vérification de l'existence d'un fichier

```
<?php
if (file_exists("exemple.txt")) {
    echo "Le fichier existe.";
} else {
    echo "Le fichier n'existe pas.";
}
?>
```

### 2 Suppression d'un fichier

```
<?php
if (file_exists("exemple.txt")) {
    unlink("exemple.txt");
    echo "Fichier supprimé.";
} else {
    echo "Fichier introuvable.";
}
?>
```

# Manipulation des données

## Gestion des fichiers

### Manipulation de fichiers

#### 3 Renommer un fichier

```
<?php
if (file_exists("exemple.txt")) {
    rename("exemple.txt", "nouveau_nom.txt");
    echo "Fichier renommé.";
} else {
    echo "Fichier introuvable.";
}
?>
```

#### 4 Copier un fichier

```
<?php
if (file_exists("exemple.txt")) {
    copy("exemple.txt", "copie.txt");
    echo "Fichier copié.";
}
?>
```

# Gestion des fichiers

## Gestion des Téléchargements de fichiers

PHP permet d'uploader des fichiers depuis un formulaire HTML.

### Formulaire HTML

```
<form action="upload.php" method="POST" enctype="multipart/form-data">
    <label for="fichier">Choisir un fichier :</label>
    <input type="file" id="fichier" name="fichier">
    <button type="submit">Télécharger</button>
</form>
```

### Traitement en PHP

```
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['fichier'])) {
    $nomFichier = $_FILES['fichier']['name'];
    $cheminTemp = $_FILES['fichier']['tmp_name'];
    $dossierDestination = "uploads/";

    if (move_uploaded_file($cheminTemp, $dossierDestination . $nomFichier)) {
        echo "Fichier téléchargé avec succès : $nomFichier";
    } else {
        echo "Échec du téléchargement.";
    }
}
```

# Manipulation des données

## Gestion des fichiers

### Manipulation des répertoires

#### 1 Crédation d'un répertoire

```
<?php
if (!is_dir("nouveau_dossier")) {
    mkdir("nouveau_dossier");
    echo "Dossier créé.";
}
?>
```

#### 2 Lecture d'un répertoire

```
<?php
$dossier = opendir(".");
if ($dossier) {
    while (($fichier = readdir($dossier)) !== false) {
        echo "Nom du fichier/dossier : $fichier<br>";
    }
    closedir($dossier);
}
?>
```

#### 3 Suppression d'un répertoire vide

```
<?php
if (is_dir("nouveau_dossier")) {
    rmdir("nouveau_dossier");
    echo "Dossier supprimé.";
}
?>
```

## Gestion des fichiers

### Bonnes pratiques

#### Validation des Entrées :

Vérifiez les noms et types des fichiers uploadés pour éviter les failles de sécurité.

```
$typeFichier = mime_content_type($cheminTemp);
if ($typeFichier !== 'image/jpeg') {
    echo "Format non autorisé.";
}
```

#### Gestion des Erreurs :

Utilisez des blocs try-catch ou des vérifications pour gérer les erreurs.

```
if (!file_exists("inconnu.txt")) {
    echo "Erreur : fichier introuvable.";
}
```

#### Permissions :

Assurez-vous que les permissions des fichiers et répertoires sont configurées correctement.

# Gestion des fichiers

## Exercice : Lecture et Écriture

### Exercices pratiques :

- Créez un fichier, écrivez-y du texte.
- Lisez et affichez son contenu.

### Exercice : Téléchargement

- Créez un formulaire pour télécharger un fichier.
- Affichez son nom, taille et type après l'upload.

## Gestion des fichiers

### Quiz

Quelle fonction est utilisée pour supprimer un fichier ?

- a) delete()
- b) unlink()
- c) remove()

Quel mode permet d'ajouter du contenu à un fichier ?

- a) w
- b) a
- c) r

# **PHP**

**Bases de données avec PHP**

# Introduction à MySQL

***MySQL est un système de gestion de base de données relationnelles (SGBDR) open source très populaire. Il permet de stocker, organiser et manipuler des données de manière efficace. En association avec PHP, MySQL est souvent utilisé pour créer des sites web dynamiques.***

# Introduction à MySQL

## Qu'est-ce qu'une base de données ?

### Définition

Une base de données est un ensemble structuré de données, organisé pour permettre un accès, une gestion et une mise à jour efficaces.

### Types de bases de données

- **Bases de données relationnelles** : Utilisent des tables reliées entre elles (ex : MySQL, PostgreSQL).
- **Bases de données NoSQL** : Utilisent des formats non tabulaires (ex : MongoDB).

### Concepts clés

- **Table** : Une collection de données organisée en lignes et colonnes.
- **Ligne (ou enregistrement)** : Une unité de données dans une table.
- **Colonne (ou champ)** : Une catégorie de données dans une table.
- **Clé primaire** : Un identifiant unique pour chaque enregistrement.
- **Clé étrangère** : Un champ reliant deux tables.

Exemple de table : utilisateurs

id	nom	email
1	Alice	alice@example.com
2	Bob	bob@example.com

# Introduction à MySQL

## Présentation de MySQL

### Pourquoi MySQL ?

- **Open Source** : Gratuit et largement adopté.
- **Rapide et fiable** : Conçu pour les applications web et les systèmes transactionnels.
- **Support SQL** : Langage standard pour interagir avec les bases de données.
- **Intégration avec PHP** : Idéal pour les sites web dynamiques.

### Composants clés

- **Serveur MySQL** : Le logiciel qui gère les bases de données.
- **Client MySQL** : Outils pour interagir avec le serveur (ex : ligne de commande, interfaces graphiques comme phpMyAdmin).

# Introduction à MySQL

## Le Langage SQL (Structured Query Language)

SQL est le langage standard pour manipuler les bases de données relationnelles.

### Catégories de commandes SQL

DML (Data Manipulation Language) : Gérer les données.

- SELECT : Lire des données.
- INSERT : Ajouter des données.
- UPDATE : Modifier des données.
- DELETE : Supprimer des données.

DDL (Data Definition Language) : Définir la structure.

- CREATE : Créer des bases de données ou des tables.
- ALTER : Modifier la structure.
- DROP : Supprimer des bases ou des tables.

DCL (Data Control Language) : Contrôler les droits.

- GRANT : Donner des permissions.
- REVOKE : Retirer des permissions.

TCL (Transaction Control Language) : Gérer les transactions.

- COMMIT : Enregistrer les changements.
- ROLLBACK : Annuler les changements.

# Introduction à MySQL

## Connexion PHP à MySQL

PHP utilise des extensions comme **mysqli** ou **PDO** pour interagir avec MySQL.

### Connexion avec mysqli

```
<?php
$serveur = "localhost";
$utilisateur = "root";
$motDePasse = "";
$baseDeDonnees = "test";

// Créer une connexion
$connexion = new mysqli($serveur, $utilisateur, $motDePasse, $baseDeDonnees);

// Vérifier la connexion
if ($connexion->connect_error) {
    die("Connexion échouée : " . $connexion->connect_error);
}

echo "Connexion réussie.";
?>
```

# Introduction à MySQL

## Connexion PHP à MySQL

### Connexion avec PDO

```
<?php
try {
    $connexion = new PDO("mysql:host=localhost;dbname=test", "root", "");
    $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connexion réussie.";
} catch (PDOException $e) {
    echo "Erreur : " . $e->getMessage();
}
?>
```

# Introduction à MySQL

## Commandes SQL de base

### Créer une base de données

```
CREATE DATABASE exemple_db;
```

### Utiliser une base de données

```
USE exemple_db;
```

### Créer une table

```
CREATE TABLE utilisateurs (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50),
    email VARCHAR(100),
    date_inscription TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

### Insérer des données

```
INSERT INTO utilisateurs (nom, email)
VALUES ('Alice', 'alice@example.com');
```

# Introduction à MySQL

## Commandes SQL de base

### Lire des données

```
SELECT * FROM utilisateurs;
```

### Mettre à jour des données

```
UPDATE utilisateurs  
SET email = 'nouvel_email@example.com'  
WHERE id = 1;
```

### Supprimer des données

```
DELETE FROM utilisateurs WHERE id = 1;
```

# Introduction à MySQL

## Exercices pratiques

### Créer une base de données

- Créez une base de données appelée mon\_site.
- Ajoutez une table articles avec les colonnes id, titre, contenu, et date.

### Manipuler des données

- Insérez 3 enregistrements dans la table articles.
- Affichez tous les articles.
- Modifiez le titre du deuxième article.
- Supprimez le troisième article.

# Introduction à MySQL

## Quizz

Quelle commande est utilisée pour créer une table ?

- a) INSERT
- b) CREATE TABLE
- c) SELECT

Quelle commande est utilisée pour lire des données ?

- a) UPDATE
- b) DELETE
- c) SELECT

# CRUD en PHP

*Le CRUD est l'acronyme de Create, Read, Update, Delete, représentant les quatre opérations de base pour manipuler les données dans une base de données. Dans cette section, nous développerons une application CRUD complète en PHP utilisant MySQL.*

## Préparation de la base de données

### Création de la table

Avant de commencer, créez une table nommée produits dans votre base de données.

```
CREATE DATABASE boutique;

USE boutique;

CREATE TABLE produits (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(100) NOT NULL,
    description TEXT,
    prix DECIMAL(10, 2) NOT NULL,
    date_ajout TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

## Préparation de la base de données

### Création de la table

Avant de commencer, créez une table nommée produits dans votre base de données.

```
CREATE DATABASE boutique;

USE boutique;

CREATE TABLE produits (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(100) NOT NULL,
    description TEXT,
    prix DECIMAL(10, 2) NOT NULL,
    date_ajout TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

# CRUD en PHP

## Configuration de la connexion

### Fichier config.php

Créez un fichier pour gérer la connexion à la base de données.

```
<?php  
$serveur = "localhost";  
$utilisateur = "root";  
$motDePasse = "";  
$baseDeDonnees = "boutique";  
  
try {  
    $connexion = new PDO("mysql:host=$serveur;dbname=$baseDeDonnees", $utilisateur,  
    $connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
} catch (PDOException $e) {  
    die("Erreur de connexion : " . $e->getMessage());  
}  
?>
```

Incluez ce fichier dans tous les scripts nécessitant une connexion.

## Ajout (Create)

### Formulaire d'ajout

Créez un formulaire HTML pour ajouter un produit.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Ajouter un produit</title>
</head>
<body>
    <h1>Ajouter un Produit</h1>
    <form action="ajouter_action.php" method="post">
        <label>Nom :</label>
        <input type="text" name="nom" required><br>
        <label>Description :</label>
        <textarea name="description"></textarea><br>
        <label>Prix :</label>
        <input type="number" name="prix" step=".01" required><br>
        <button type="submit">Ajouter</button>
    </form>
</body>
</html>
```

# CRUD en PHP

## Ajout (Create)

### Traitement en PHP

Ajoutez les données dans la base.

```
<?php
// ajouter_action.php
require 'config.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $nom = $_POST['nom'];
    $description = $_POST['description'];
    $prix = $_POST['prix'];

    $sql = "INSERT INTO produits (nom, description, prix) VALUES (:nom, :description";
    $stmt = $connexion->prepare($sql);
    $stmt->execute(['nom' => $nom, 'description' => $description, 'prix' => $prix]);

    echo "Produit ajouté avec succès ! <a href='lister.php'>Voir les produits</a>";
}

?>
```

## Lecture (Read)

### Affichage des produits

Récupérez et affichez les produits de la base.

```
<?php
// lister.php
require 'config.php';

$sql = "SELECT * FROM produits ORDER BY date_ajout DESC";
$stmt = $connexion->query($sql);

?>
```

```
<?php while ($produit = $stmt->fetch(PDO::FETCH_ASSOC)): ?>
<tr>
    <td><?= $produit['id'] ?></td>
    <td><?= $produit['nom'] ?></td>
    <td><?= $produit['description'] ?></td>
    <td><?= $produit['prix'] ?> €</td>
    <td><?= $produit['date_ajout'] ?></td>
    <td>
        <a href="modifier.php?id=<?= $produit['id'] ?>">Modifier</a>
        <a href="supprimer.php?id=<?= $produit['id'] ?>" onclick="return confirm('
    </td>
</tr>
<?php endwhile; ?>
```

## Mise à jour (Update)

### Formulaire de modification

Chargez les données existantes dans un formulaire.

```
require 'config.php';

$id = $_GET['id'];
$sql = "SELECT * FROM produits WHERE id = :id";
$stmt = $connexion->prepare($sql);
$stmt->execute(['id' => $id]);
$produit = $stmt->fetch(PDO::FETCH_ASSOC);
?>

<form action="modifier_action.php" method="post">
    <input type="hidden" name="id" value="<?= $produit['id'] ?>">
    <label>Nom :</label>
    <input type="text" name="nom" value="<?= $produit['nom'] ?>" required><br>
    <label>Description :</label>
    <textarea name="description"><?= $produit['description'] ?></textarea><br>
    <label>PrixA :</label>
    <input type="number" name="prix" step="0.01" value="<?= $produit['prix'] ?>">
    <button type="submit">Modifier</button>
</form>
```

# CRUD en PHP

## Mise à jour (Update)

### Traitement en PHP

Mettez à jour les données.

```
<?php
// modifier_action.php
require 'config.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $id = $_POST['id'];
    $nom = $_POST['nom'];
    $description = $_POST['description'];
    $prix = $_POST['prix'];

    $sql = "UPDATE produits SET nom = :nom, description = :description, prix = :pr
$stmt = $connexion->prepare($sql);
$stmt->execute(['nom' => $nom, 'description' => $description, 'prix' => $prix,
echo "Produit modifié avec succès ! <a href='lister.php'>Retour à la liste</a>
}
?>
```

# CRUD en PHP

## Suppression (Delete)

Supprimez un produit par son ID..

```
<?php
// supprimer.php
require 'config.php';

$id = $_GET['id'];
$sql = "DELETE FROM produits WHERE id = :id";
$stmt = $connexion->prepare($sql);
$stmt->execute(['id' => $id]);

echo "Produit supprimé avec succès ! <a href='lister.php'>Retour à la liste</a>";
?>
```

# CRUD en PHP

## Exercice pratique

### Étendez le CRUD :

- Ajoutez un champ "catégorie" à la table produits.
- Mettez à jour les formulaires et scripts pour inclure ce champ.

### Ajoutez des fonctionnalités :

- Pagination dans la liste des produits.
- Recherches par nom ou prix.

# **PHP**

**Gestion des sessions et cookies**

## Sessions en PHP

***Les sessions en PHP permettent de conserver des données entre les pages web pour un utilisateur donné. Elles sont très utiles pour gérer des systèmes comme les connexions utilisateur, les paniers d'achat, ou le suivi de préférences.***

# Sessions en PHP

## Qu'est-ce qu'une session ?

- Une session en PHP est une manière de stocker des informations spécifiques à un utilisateur sur le serveur. Une fois qu'une session est démarrée, des données peuvent être stockées dans la variable superglobale **`$_SESSION`** et accessibles sur plusieurs pages.
- **Durée** : La session persiste tant qu'elle n'est pas explicitement détruite ou jusqu'à ce qu'elle expire.
- **Stockage** : Les données de session sont sauvegardées côté serveur, tandis qu'un cookie contenant l'identifiant unique de session (généralement `PHPSESSID`) est envoyé au client.

# Sessions en PHP

## Utilisation de sessions

### Démarrer une session

Avant d'utiliser une session, vous devez la démarrer avec **session\_start()**. Cela doit être fait avant tout autre envoi de contenu dans la page.

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

### Stocker des données dans une session

Vous pouvez ajouter des informations dans la session en utilisant la superglobale **\$\_SESSION**.

```
<?php  
session_start();  
  
// Ajouter des informations dans la session  
$_SESSION['username'] = 'Alice';  
$_SESSION['email'] = 'alice@example.com';  
$_SESSION['role'] = 'admin';  
  
echo "Données de session enregistrées.";  
?>
```

# Sessions en PHP

## Utilisation de sessions

### Supprimer ou réinitialiser une session

#### Supprimer une variable spécifique

Pour supprimer une variable particulière de la session, utilisez `unset()` :

```
<?php  
session_start();  
  
// Supprimer une variable spécifique  
unset($_SESSION['email']);  
?>
```

#### Réinitialiser toutes les variables

Pour supprimer toutes les variables de session sans détruire la session elle-même, utilisez `session_unset()`

#### Détruire complètement une session

Pour détruire une session, utilisez `session_destroy()`. Cela supprime toutes les données stockées sur le serveur pour cette session.

## Sessions en PHP

### Exemple pratique : authentification avec sessions

Créons un système de connexion simple utilisant les sessions.

#### Page de connexion

Un formulaire simple pour collecter les informations de connexion.

```
<body>
    <h1>Connexion</h1>
    <form action="traitement_connexion.php" method="post">
        <label for="email">Email :</label>
        <input type="email" id="email" name="email" required><br>
        <label for="password">Mot de passe :</label>
        <input type="password" id="password" name="password" required><br>
        <button type="submit">Se connecter</button>
    </form>
</body>
```

# Sessions en PHP

## Exemple pratique : authentification avec sessions

### Traitement de la connexion

Vérifiez les informations d'utilisateur et démarrez une session si elles sont correctes.

```
<?php
// traitement_connexion.php
session_start();

// Simuler une base de données utilisateur
$utilisateurs = [
    'alice@example.com' => 'password123',
    'bob@example.com' => 'securepass'
];

// Récupérer les informations du formulaire
$email = $_POST['email'];
$password = $_POST['password'];

// Vérification des informations
if (isset($utilisateurs[$email]) && $utilisateurs[$email] === $password) {
    // Stocker les informations de l'utilisateur dans la session
    $_SESSION['email'] = $email;
    $_SESSION['connecte'] = true;

    header('Location: dashboard.php'); // Rediriger vers une page protégée
    exit();
} else {
    echo "Email ou mot de passe incorrect.";
}
?>
```

# Sessions en PHP

## Exemple pratique : authentification avec sessions

### Page protégée

Affichez des informations si l'utilisateur est connecté. Sinon, redirigez-le vers la page de connexion.

```
<?php  
// dashboard.php  
session_start();  
  
// Vérifier si l'utilisateur est connecté  
if (!isset($_SESSION['connecte']) || !$_SESSION['connecte']) {  
    header('Location: connexion.php');  
    exit();  
}  
  
// Afficher le tableau de bord  
echo "Bienvenue, " . $_SESSION['email'] . "!";  
echo "<br><a href='deconnexion.php'>Se déconnecter</a>";  
?>
```

# Sessions en PHP

## Exemple pratique : authentification avec sessions

### Déconnexion

Déconnectez l'utilisateur en détruisant sa session.

```
<?php  
// deconnection.php  
session_start();  
  
// Supprimer toutes les données de session  
session_unset();  
session_destroy();  
  
// Rediriger vers la page de connexion  
header('Location: connexion.php');  
?>
```

# Sessions en PHP

## Exercices pratiques

### Système de compteur de visites :

Créez une page qui compte le nombre de fois qu'un utilisateur l'a visitée en utilisant une session.

### Système de panier :

Ajoutez des produits à un panier stocké dans une session et affichez le contenu du panier.

## Sessions en PHP

### Quiz

Une session utilise-t-elle un cookie pour lier l'utilisateur ?

- a) Oui
- b) Non

Quelle fonction permet de réinitialiser toutes les variables de session sans la détruire ?

- a) session\_destroy()
- b) session\_unset()

# Cookies

***Les cookies sont de petits fichiers texte stockés sur l'ordinateur de l'utilisateur. Ils permettent de conserver des données entre plusieurs visites ou pages, côté client. PHP facilite la gestion des cookies grâce à une série de fonctions et superglobales.***

# Cookies

## Qu'est-ce qu'un Cookie ?

- **Stockage côté client** : Contrairement aux sessions, les cookies sont stockés dans le navigateur de l'utilisateur.
- **Durée de vie** : Vous pouvez définir une expiration pour un cookie, après quoi il est automatiquement supprimé.
- **Portée** : Les cookies sont accessibles via le domaine et le chemin définis lors de leur création.

# Cookies

## Utilisation des cookies

### Créer un cookie

Pour créer un cookie, utilisez la fonction **setcookie()** avant tout envoi de contenu (comme `session_start()`).

```
<?php  
// Créer un cookie nommé "utilisateur" qui expire dans 1 heure  
setcookie('utilisateur', 'Alice', time() + 3600);  
  
// Ajouter des options (sécurisé et HTTP only)  
setcookie(  
    'preferences',  
    json_encode(['theme' => 'clair', 'langue' => 'fr']),  
    [  
        'expires' => time() + 86400, // 1 jour  
        'secure' => true,           // Accessible uniquement via HTTPS  
        'httponly' => true,          // Inaccessible via JavaScript  
        'samesite' => 'Strict'       // Protection contre les attaques CSRF  
    ]  
);  
?>
```

## Cookies

### Utilisation des cookies

#### Créer un cookie

Paramètres de setcookie() :

**Nom** : Le nom du cookie.

**Valeur** : La valeur du cookie.

**Expire** : Timestamp Unix indiquant la date d'expiration.

**Chemin** : Chemin pour lequel le cookie est valide (par défaut /).

**Domaine** : Sous-domaine pour lequel le cookie est valide (par défaut le domaine courant).

**Secure** : Le cookie est transmis uniquement via HTTPS.

**HttpOnly** : Empêche l'accès JavaScript au cookie.

**SameSite** : Précise si le cookie est limité aux requêtes provenant du même site.

## Cookies

### Utilisation des cookies

#### Lire un cookie

Les cookies envoyés par le client sont accessibles via la superglobale `$_COOKIE`.

```
<?php  
// Vérifier si le cookie existe  
if (isset($_COOKIE['utilisateur'])) {  
    echo "Bonjour, " . $_COOKIE['utilisateur'];  
} else {  
    echo "Aucun cookie utilisateur trouvé.";  
}  
?>
```

## Cookies

### Utilisation des cookies

#### Mettre à jour un cookie

Pour modifier un cookie, utilisez setcookie() avec le même nom, mais une nouvelle valeur ou de nouvelles options.

```
<?php  
// Modifier le cookie "utilisateur"  
setcookie('utilisateur', 'Bob', time() + 3600);  
?>
```

#### Supprimer un cookie

Pour supprimer un cookie, définissez une date d'expiration passée.

```
<?php  
// Supprimer un cookie en définissant une expiration passée  
setcookie('utilisateur', '', time() - 3600);  
?>
```

# Cookies

## Exemple pratique : Cookies de Préférences

Créons un système simple pour enregistrer les préférences de thème d'un utilisateur.

### Formulaire de préférences.

```
<body>
    <h1>Choisissez vos préférences</h1>
    <form action="traitement_preferences.php" method="post">
        <label for="theme">Thème :</label>
        <select name="theme" id="theme">
            <option value="clair">Clair</option>
            <option value="sombre">Sombre</option>
        </select><br>

        <label for="langue">Langue :</label>
        <select name="langue" id="langue">
            <option value="fr">Français</option>
            <option value="en">Anglais</option>
        </select><br>

        <button type="submit">Enregistrer</button>
    </form>
</body>
```

# Cookies

## Exemple pratique : Cookies de Préférences

Traitement et création des cookies.

```
<?php  
// traitement_preferences.php  
  
// Récupérer les préférences de l'utilisateur  
$theme = $_POST['theme'];  
$langue = $_POST['langue'];  
  
// Stocker les préférences dans des cookies  
setcookie('theme', $theme, time() + (86400 * 30)); // Expire dans 30 jours  
setcookie('langue', $langue, time() + (86400 * 30));  
  
// Rediriger l'utilisateur  
header('Location: accueil.php');  
exit();  
?>
```

# Cookies

## Exemple pratique : Cookies de Préférences

### Utilisation des cookies.

```
<?php  
// accueil.php  
  
// Vérifier les préférences  
$theme = isset($_COOKIE['theme']) ? $_COOKIE['theme'] : 'clair';  
$langue = isset($_COOKIE['langue']) ? $_COOKIE['langue'] : 'fr';  
  
// Appliquer les préférences  
if ($theme === 'sombre') {  
    echo "<body style='background-color: black; color: white;'>";  
} else {  
    echo "<body style='background-color: white; color: black;'>";  
}  
  
echo "<h1>Bienvenue !</h1>";  
echo "Langue : " . ($langue === 'fr' ? 'Français' : 'Anglais');  
?>
```

# Cookies

## Exemple pratique : Cookies de Préférences

### Cookies vs Sessions.

Aspect	Cookies	Sessions
Stockage	Côté client	Côté serveur
Sécurité	Moins sécurisé (accessible par le client)	Plus sécurisé
Durée	Personnalisable (par expiration)	Jusqu'à la fermeture du navigateur (par défaut)
Taille maximale	4 Ko	Limité par la mémoire du serveur

## Cookies

### Bonnes pratiques

#### Sécuriser les cookies :

- Utilisez l'option secure pour HTTPS.
- Activez HttpOnly pour empêcher les scripts d'accéder aux cookies sensibles.

#### Limiter l'utilisation :

- Ne stockez pas de données sensibles dans les cookies (ex. mots de passe).
- Limitez la taille des cookies pour éviter les problèmes de performance.

#### Combiner avec sessions :

- Utilisez les cookies pour des données non sensibles (ex. préférences).
- Combinez-les avec des sessions pour une gestion sécurisée des données sensibles.

## Cookies

### Exercices pratiques

#### Créer un système de langue préférée :

- Enregistrez la langue préférée de l'utilisateur dans un cookie et affichez une page dans la langue correspondante.

#### Compteur de Visites :

- Utilisez un cookie pour compter combien de fois un utilisateur a visité une page.

## Cookies

### Quizz

Un cookie est-il stocké côté client ou serveur ?

- a) Client
- b) Serveur

Quelle option empêche JavaScript d'accéder à un cookie ?

- a) HttpOnly
- b) Secure

# **PHP**

**PHP avancé**

# Programmation Orientée Objet (POO)

***La Programmation Orientée Objet (POO) est une méthode de programmation qui structure le code en entités appelées objets, définis à partir de classes. Cette approche permet de rendre le code plus modulaire, réutilisable et facile à maintenir.***

# Programmation Orientée Objet (POO)

## Principes fondamentaux de la POO

**Classe** : Une classe est un modèle ou une structure qui définit les propriétés (attributs) et les comportements (méthodes) d'un objet.

**Objet** : Une instance concrète d'une classe.

**Attributs** : Les variables définies dans une classe pour décrire les propriétés de l'objet.

**Méthodes** : Les fonctions définies dans une classe qui décrivent les comportements d'un objet.

**Encapsulation** : Restreindre l'accès direct aux données d'un objet et fournir des méthodes pour y accéder ou les modifier.

**Héritage** : Une classe peut hériter des propriétés et des méthodes d'une autre classe.

**Polymorphisme** : La capacité d'utiliser une méthode dans plusieurs contextes grâce à la surcharge ou à l'héritage.

# Programmation Orientée Objet (POO)

## Créer une Classe et un Objet

```
<?php  
// Définition de la classe  
class Voiture {  
    // Attributs  
    public $marque;  
    public $couleur;  
  
    // Constructeur  
    public function __construct($marque, $couleur) {  
        $this->marque = $marque;  
        $this->couleur = $couleur;  
    }  
  
    // Méthode  
    public function demarrer() {  
        echo "La voiture $this->marque de couleur $this->couleur démarre.<br>";  
    }  
}  
  
// Instanciation de la classe  
$maVoiture = new Voiture("Toyota", "Rouge");  
$maVoiture->demarrer();  
?>
```

# Programmation Orientée Objet (POO)

## Encapsulation

L'encapsulation protège les données d'une classe et fournit des méthodes pour y accéder.

```
<?php  
class CompteBancaire {  
    private $solde = 0;  
  
    // Getter  
    public function getSolde() {  
        return $this->solde;  
    }  
  
    // Setter  
    public function deposer($montant) {  
        if ($montant > 0) {  
            $this->solde += $montant;  
        }  
    }  
}  
  
$compte = new CompteBancaire();  
$compte->deposer(1000);  
echo "Solde : " . $compte->getSolde(); // Solde : 1000  
?>
```



# Programmation Orientée Objet (POO)

## Héritage

Une classe peut hériter des attributs et méthodes d'une autre classe.

```
<?php  
class Animal {  
    public $nom;  
  
    public function __construct($nom) {  
        $this->nom = $nom;  
    }  
  
    public function parler() {  
        echo "$this->nom fait un bruit.<br>";  
    }  
}  
  
// Classe enfant  
class Chien extends Animal {  
    public function parler() {  
        echo "$this->nom aboie.<br>";  
    }  
}  
  
$animal = new Animal("Animal");  
$animal->parler();  
  
$chien = new Chien("Max");  
$chien->parler();  
?>
```

# Programmation Orientée Objet (POO)

## Polymorphisme

Le polymorphisme permet d'utiliser une méthode ou une classe de manière flexible.

### Surcharge des Méthodes

Une classe enfant peut redéfinir une méthode de la classe parent.

```
<?php  
class Forme {  
    public function aire() {  
        return 0;  
    }  
  
    class Carré extends Forme {  
        private $cote;  
  
        public function __construct($cote) {  
            $this->cote = $cote;  
        }  
  
        public function aire() {  
            return $this->cote * $this->cote;  
        }  
    }  
  
    $carre = new Carré(4);  
    echo "Aire du carré : " . $carre->aire(); // Aire du carré : 16  
?>
```

# Programmation Orientée Objet (POO)

## Interfaces et classes abstraites

### Classe abstraite

Une classe abstraite sert de modèle pour d'autres classes. Elle ne peut pas être instanciée directement.

```
<?php  
abstract class Vehicule {  
    abstract public function demarrer();  
}  
  
class Moto extends Vehicule {  
    public function demarrer() {  
        echo "La moto démarre.<br>";  
    }  
}  
  
$moto = new Moto();  
$moto->demarrer();  
?>
```

# Programmation Orientée Objet (POO)

## Interfaces et classes abstraites

### Interface

Une interface définit un contrat que les classes doivent respecter.

```
<?php  
interface Transport {  
    public function deplacer();  
}  
  
class Avion implements Transport {  
    public function deplacer() {  
        echo "L'avion vole.<br>";  
    }  
}  
  
$avion = new Avion();  
$avion->deplacer();  
?>
```

# Programmation Orientée Objet (POO)

## Exemple pratique : Gestion d'une bibliothèque

### Classe Livre

```
<?php  
class Livre {  
    private $titre;  
    private $auteur;  
  
    public function __construct($titre, $auteur) {  
        $this->titre = $titre;  
        $this->auteur = $auteur;  
    }  
  
    public function afficher() {  
        echo "Titre : $this->titre, Auteur : $this->auteur<br>";  
    }  
}  
?>
```

# Programmation Orientée Objet (POO)

## Exemple pratique : Gestion d'une bibliothèque

### Classe Bibliothèque

```
<?php  
class Bibliotheque {  
    private $livres = [];  
  
    public function ajouterLivre(Livre $livre) {  
        $this->livres[] = $livre;  
    }  
  
    public function afficherLivres() {  
        foreach ($this->livres as $livre) {  
            $livre->afficher();  
        }  
    }  
?>
```

# Programmation Orientée Objet (POO)

## Exemple pratique : Gestion d'une bibliothèque

### Utilisation

```
<?php  
$livre1 = new Livre("1984", "George Orwell");  
$livre2 = new Livre("Le Petit Prince", "Antoine de Saint-Exupéry");  
  
$bibliotheque = new Bibliotheque();  
$bibliotheque->ajouterLivre($livre1);  
$bibliotheque->ajouterLivre($livre2);  
  
$bibliotheque->afficherLivres();  
?>
```

# Programmation Orientée Objet (POO)

## Exercices

**Créer un système de gestion des étudiants :**

- Classe Etudiant avec des attributs comme nom, âge, et classe.
- Classe Ecole pour gérer plusieurs étudiants.

# Programmation Orientée Objet (POO)

## Quizz

**Une classe abstraite peut-elle être instanciée ?**

- a) Oui
- b) Non

**Quel mot-clé protège un attribut dans une classe ?**

- a) private
- b) protected
- c) public

# Traitement des API

*Une API (Application Programming Interface) permet à des applications différentes de communiquer entre elles. Les APIs sont souvent basées sur le protocole HTTP et retournent des données dans des formats structurés comme JSON ou XML. PHP propose des outils puissants pour consommer et interagir avec des APIs externes ou pour créer vos propres APIs.*

# Traitement des API

## Comprendre les APIs

Types d'APIs :

- APIs REST (Representational State Transfer) : Les plus répandues. Utilisent des méthodes HTTP (GET, POST, PUT, DELETE) et renvoient souvent des données au format JSON.
- APIs SOAP (Simple Object Access Protocol) : Plus anciennes, elles utilisent XML et une structure plus rigide.

Principales Méthodes HTTP :

- GET : Récupérer des données.
- POST : Envoyer des données.
- PUT : Mettre à jour des données.
- DELETE : Supprimer des données.

Structure d'une API REST typique : Exemple : <https://api.exemple.com/users/123>

- <https://api.exemple.com> : Domaine de l'API.
- /users : Ressource.
- /123 : Identifiant de la ressource.

# Traitement des API

## Consommer une API avec PHP

Utilisation de `file_get_contents` (Simple)

```
<?php  
$url = "https://jsonplaceholder.typicode.com/posts/1";  
$response = file_get_contents($url);  
  
// Décoder le JSON en tableau PHP  
$data = json_decode($response, true);  
  
echo "Titre : " . $data['title'] . "<br>";  
?>
```

# Traitement des API

## Création d'une API en PHP

### API simple avec PHP (sans framework)

#### Étape 1 : Préparer les données

```
<?php  
// Exemple de données simulées  
$articles = [  
    ['id' => 1, 'titre' => 'Introduction à PHP', 'contenu' => 'Lorem ipsum...'],  
    ['id' => 2, 'titre' => 'API REST avec PHP', 'contenu' => 'Dolor sit amet...']  
];  
?>
```

# Traitement des API

## Création d'une API en PHP

### API simple avec PHP (sans framework)

#### Étape 2 : Traitement de la requête

```
<?php
// Définir l'en-tête pour indiquer le type de contenu
header('Content-Type: application/json');

// Vérifier la méthode HTTP
$method = $_SERVER['REQUEST_METHOD'];

if ($method === 'GET') {
    // Retourner tous les articles
    echo json_encode($articles);
} elseif ($method === 'POST') {
    // Récupérer les données POST
    $input = json_decode(file_get_contents('php://input'), true);

    // Ajouter un nouvel article
    $nouvelArticle = [
        'id' => count($articles) + 1,
        'titre' => $input['titre'],
        'contenu' => $input['contenu']
    ];
    $articles[] = $nouvelArticle;

    // Retourner la réponse
    echo json_encode(['message' => 'Article ajouté', 'article' => $nouvelArticle])
} else {
    // Méthode non supportée
    http_response_code(405);
    echo json_encode(['message' => 'Méthode non autorisée']);
}
?>
```

# Traitement des API

## Création d'une API en PHP

### API avec un framework (Laravel, Symfony)

Les frameworks comme Laravel ou Symfony offrent des outils puissants pour créer des APIs avec routage, middleware, et validation intégrés. Par exemple, avec Laravel :

```
Route::get('/articles', [ArticleController::class, 'index']);  
Route::post('/articles', [ArticleController::class, 'store']);
```

# Traitement des API

## Exemple pratique : API de gestion des Utilisateurs

### Serveur (API en PHP)

```
<?php

// Simuler une base de données
$utilisateurs = [
    ['id' => 1, 'nom' => 'Alice', 'email' => 'alice@example.com'],
    ['id' => 2, 'nom' => 'Bob', 'email' => 'bob@example.com']
];

// Définir le type de contenu
header('Content-Type: application/json');

// Récupérer la méthode
$method = $_SERVER['REQUEST_METHOD'];

// Router la requête
if ($method === 'GET') {
    echo json_encode($utilisateurs);
} elseif ($method === 'POST') {
    $input = json_decode(file_get_contents('php://input'), true);
    $nouvelUtilisateur = [
        'id' => count($utilisateurs) + 1,
        'nom' => $input['nom'],
        'email' => $input['email']
    ];
    $utilisateurs[] = $nouvelUtilisateur;
    echo json_encode(['message' => 'Utilisateur ajouté', 'utilisateur' => $nouvelU
} else {
    http_response_code(405);
    echo json_encode(['message' => 'Méthode non autorisée']);
}
?>
```

# Traitement des API

## Client (PHP Consommateur de l'API)

```
<?php  
$url = "http://localhost/api/utilisateurs";  
  
// Requête GET  
$response = file_get_contents($url);  
$data = json_decode($response, true);  
print_r($data);  
  
// Requête POST  
$postData = ['nom' => 'Charlie', 'email' => 'charlie@example.com'];  
$options = [  
    'http' => [  
        'header' => "Content-Type: application/json",  
        'method' => 'POST',  
        'content' => json_encode($postData)  
    ]  
];  
$context = stream_context_create($options);  
$response = file_get_contents($url, false, $context);  
echo $response;  
?>
```



# Traitement des API

## Bonnes pratiques

### Validation des données :

Toujours valider et nettoyer les données reçues de l'utilisateur.

### Gestion des erreurs :

Retourner des codes d'erreur HTTP appropriés (ex. 400 pour une mauvaise requête, 401 pour une autorisation manquante).

### Authentification :

Protéger l'API avec des clés API, tokens, ou OAuth.

### Documentation :

Documenter votre API avec des outils comme Swagger ou Postman.

### Limiter les appels :

Implémenter des quotas pour éviter les abus (rate-limiting).

# Traitement des API

## Exercices

**Créer une API qui gère des produits :**

Implémentez les méthodes GET, POST, PUT et DELETE pour une ressource produit.

**QCM :**

Quelle méthode HTTP est utilisée pour mettre à jour une ressource ?

- a) GET
- b) POST
- c) PUT
- d) DELETE

Quel format est le plus utilisé dans les APIs modernes ?

- a) XML
- b) JSON
- c) CSV

*La sécurité est un aspect critique dans le développement d'applications web. Une application vulnérable peut être compromise par des attaques comme le vol de données, la modification de contenu ou la prise de contrôle totale. En PHP, plusieurs pratiques et outils permettent de sécuriser votre code et de protéger vos applications.*

# Sécurité en PHP

## Les menaces communes

### Injection SQL

Se produit lorsque des entrées utilisateur sont directement intégrées dans les requêtes SQL.

Exemple de vulnérabilité :

```
<?php  
$nom = $_GET['nom'];  
$query = "SELECT * FROM utilisateurs WHERE nom = '$nom'";  
mysqli_query($connexion, $query);
```

### Cross-Site Scripting (XSS)

Permet d'injecter du contenu malveillant (souvent du JavaScript) dans une page web.

Exemple :

Un champ de commentaire non filtré qui permet à un utilisateur de soumettre

```
<script>alert('Hacked!')</script> .
```

### Cross-Site Request Forgery (CSRF)

Exploite une session utilisateur active pour effectuer des actions non autorisées sur un site.

### Inclusion de fichiers

Inclure dynamiquement des fichiers non sécurisés peut permettre à un attaquant d'exécuter du code arbitraire.

### Brute force et vol de Sessions

Essayer de deviner les mots de passe ou voler les cookies de session.

# Sécurité en PHP

## Bonnes pratiques pour sécuriser le code

### Validation et nettoyage des données

Toujours valider les entrées utilisateur et les nettoyer.

Exemple :

```
<?php  
$nom = filter_input(INPUT_GET, 'nom', FILTER_SANITIZE_STRING);  
?>
```

### Préparer les requêtes SQL (PDO ou MySQLi Préparé)

Utiliser des requêtes préparées pour éviter les injections SQL.

Exemple avec PDO :

```
<?php  
$connexion = new PDO('mysql:host=localhost;dbname=ma_base', 'utilisateur', 'motdepasse');  
  
$stmt = $connexion->prepare("SELECT * FROM utilisateurs WHERE nom = :nom");  
$stmt->bindParam(':nom', $nom);  
$stmt->execute();  
$resultats = $stmt->fetchAll();  
?>
```

# Sécurité en PHP

## Bonnes pratiques pour sécuriser le code

### Générer des tokens pour CSRF

Associez un jeton unique à chaque formulaire pour empêcher les requêtes non autorisées.

Exemple :

```
<?php
// Génération du token
session_start();
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
?>
<form method="POST">
    <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']
</form>
```

Validation du token côté serveur :

```
<?php
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die('Requête invalide !');
}
?>
```

# **PHP**

**Travail avec les frameworks PHP**

# Pourquoi utiliser un Framework ?

- ....

# Introduction à Laravel

• ....

## Autres frameworks

- ....

# **PHP**

**Projet final**

## Créer un projet complet, par exemple :

- Système de gestion des utilisateurs : Inscription, connexion, modification de profil.
- Blog : Ajout, édition, et suppression d'articles.
- Boutique en ligne : Gestion des produits, panier, commandes.