CoursInfo

PHP — Développement Web Dynamique

Développez des applications web dynamiques avec PHP : formulaires, sessions et connexion à une base de données MySQL.

Niveau : Intermédiaire • Durée : ~16h • 8 chapitres • Certificat inclus

---

1. Introduction à PHP et à la pile LAMP

PHP (PHP: Hypertext Preprocessor) est un langage de script côté serveur : le code s'exécute sur le serveur web, et seul le résultat (du HTML) est envoyé au navigateur. C'est le langage derrière WordPress, une grande partie de Facebook historiquement, et la majorité des sites dynamiques du web.

La pile LAMP

Lettre	Composant	Rôle
L	Linux	Système d'exploitation du serveur
A	Apache	Serveur web qui reçoit les requêtes HTTP
M	MySQL	Base de données
P	PHP	Langage qui génère le HTML dynamiquement

```
<?php

echo
"Bonjour depuis PHP !"
;

?>
```

■ Extension .php : un fichier doit se terminer en .php pour qu'Apache l'exécute comme du code serveur plutôt que de l'envoyer tel quel.

2. Syntaxe de base et variables

```
<?php


$nom
=
"Fatou"
;
// chaîne de caractères


$age
=
22
;
// entier


$moyenne
=
15.5
;
// nombre décimal


$actif
=
true
;
// booléen


// Concaténation avec le point

echo
"Bonjour "
.
$nom
;


// Interpolation directe dans une chaîne double-guillemets

echo
"Bonjour $nom, tu as $age ans"
;

?>
```

■■ Toutes les variables PHP commencent par $, contrairement à JavaScript. C'est une erreur fréquente pour les débutants habitués à d'autres langages.

3. Conditions et boucles

```
<?php


if
(
$note
>=
10
) {
echo
"Admis"
;
}
elseif
(
$note
>=
8
) {
echo
"Rattrapage"
;
}
else
{
echo
"Ajourné"
;
}


// Boucle for


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


// Boucle foreach — la plus utilisée avec les tableaux


$cours
= [
```

```
"HTML5"
,
"CSS3"
,
"PHP"
];

foreach
(
$cours

as

$c
) {
echo
$c
.
"<br>"
;
}

?>
```

4. Fonctions et tableaux associatifs

```
<?php


function
addition(
$a
,
$b
) {

return

$a
+
$b
;
}
echo addition(
4
,
5
);
// 9


// Tableau indexé


$notes
= [
12
,
15
,
9
];


// Tableau associatif (clé => valeur)


$etudiant
= [

"nom"
=>
"Fatou"
,

"filiere"
=>
"Réseaux"

];
echo
$etudiant
[
```

```
"nom"
];

?>
```

5. Traiter un formulaire avec $_GET et $_POST

Quand un formulaire HTML est soumis, PHP récupère les données envoyées via les tableaux superglobaux $_GET (dans l'URL) ou $_POST (dans le corps de la requête, invisible dans l'URL).

```
<form

action
=
"traitement.php"

method
=
"POST"
>


<input

type
=
"text"

name
=
"nom"
>


<button

type
=
"submit"
>
Envoyer
</button>


</form>

<?php


if
(
$_SERVER
[
'REQUEST_METHOD'
] ===
'POST'
) {

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

```
"Bonjour $nom"
;
}

?>
```

■■ Toujours filtrer les entrées : htmlspecialchars() neutralise le HTML/JS injecté dans un champ, ce qui protège contre les attaques XSS.

6. Sessions et cookies

HTTP est un protocole « sans état » : chaque requête est indépendante. Les sessions et cookies permettent de mémoriser un utilisateur d'une page à l'autre (ex. rester connecté).

```
<?php

session_start();
// à appeler avant tout affichage, sur chaque page


// Stocker une donnée de session (côté serveur)


$_SESSION
[
'utilisateur'
] =
'fatou@example.com'
;


// La relire sur une autre page


if
(isset(
$_SESSION
[
'utilisateur'
])) {
echo
"Connecté en tant que "
.
$_SESSION
[
'utilisateur'
];
}


// Détruire la session (déconnexion)

session_destroy();

?>
```

■■ Session vs cookie : une session stocke les données côté serveur (sécurisé) et n'envoie qu'un identifiant au navigateur ; un cookie classique stocke la donnée directement chez le client.

7. Connexion à MySQL avec mysqli / PDO

PHP devient réellement puissant combiné à une base de données. La bibliothèque mysqli (ou PDO) permet de dialoguer avec MySQL.

```
<?php


$conn
=
new
mysqli(
"localhost"
,
"admin"
,
"motdepasse"
,
"cours_info"
);


if
(
$conn
->connect_error) {
die(
"Erreur de connexion : "
.
$conn
->connect_error);
}


$resultat
=
$conn
->query(
"SELECT * FROM cours"
);

while
(
$ligne
=
$resultat
->fetch_assoc()) {
echo
$ligne
[
'titre'
];
}

$conn
->close();

?>
```

Sécurité : requêtes préparées

Concaténer directement une variable dans une requête SQL expose à l'injection SQL. Les requêtes préparées séparent le code SQL des données saisies par l'utilisateur.

```
$stmt
=
$conn
->prepare(
"SELECT * FROM utilisateurs WHERE email = ?"
);

$stmt
->bind_param(
"s"
,
$email
);

$stmt
->execute();

$resultat
=
$stmt
->get_result();
```

■■ Ne jamais faire : "SELECT * FROM users WHERE email='" . $_POST['email'] . "'" — un attaquant peut injecter du SQL arbitraire dans ce champ.

8. Projet final : formulaire de contact fonctionnel

■ Projet final : Construisez un formulaire de contact complet : Formulaire HTML (nom, email, sujet, message) envoyé en POST Script PHP qui valide les champs (non vides, email valide avec filter_var($email, FILTER_VALIDATE_EMAIL)) Insertion du message dans une table MySQL via une requête préparée Message de confirmation affiché à l'utilisateur

■ Étape suivante : approfondissez la base de données avec le cours MySQL pour bien structurer vos tables (types, clés, relations).

Exercices & Quiz de vérification

Testez vos connaissances avant de passer au cours suivant. Chaque question a une correction détaillée à dérouler.

1. Comment déclare-t-on une variable en PHP ?

- A. let nom = "Fatou";
- B. $nom = "Fatou";
- C. var nom : "Fatou";

Correction : Réponse : B. Toutes les variables PHP commencent obligatoirement par le symbole $.

2. Quelle superglobale contient les données d'un formulaire envoyé en POST ?

- A. $_GET
- B. $_POST
- C. $_FORM

Correction : Réponse : B. $_POST contient les champs envoyés avec method="POST" ; $_GET contient ceux passés dans l'URL.

3. Pourquoi utiliser des requêtes préparées avec MySQL ?

- A. Elles sont plus courtes à écrire
- B. Elles empêchent l'injection SQL
- C. Elles sont obligatoires en PHP

Correction : Réponse : B. Elles séparent le code SQL des données utilisateur, empêchant un attaquant d'injecter du SQL malveillant.

4. Que fait session_start() ?

- A. Elle démarre le serveur Apache
- B. Elle initialise ou reprend la session utilisateur en cours
- C. Elle crée une base de données

Correction : Réponse : B. Elle doit être appelée en tout début de script, sur chaque page utilisant $_SESSION.

Ressources supplémentaires

- Manuel officiel PHP (php.net) — Référence complète en français
- PHP.net — Prévenir les injections SQL
- PHP.net — Extension mysqli
- MDN — Introduction à la programmation côté serveur