

# Chapitre 1 (p2): Concepts de base du langage PHP

---

ING 3

S. NOUFEL

# Objectifs

---

1. S'initier à la programmation web en utilisant PHP
2. Maîtriser les concepts de bases (déclaration, structures conditionnelles, structures itératives)

# Plan

---

1. Introduction
2. Objectifs
3. Environnement de travail  
(Propositions)
4. Concepts de base

# Introduction

---

## Histoire:

Langage PHP est créé en 1994 par Ramus Lerdorf

En 1997, il est réécrit par Zeev Suraski & Andi Gutmans

En 2000 PHP4

Actuellement PHP8

....

Nov 2025 le passage à la version PHP 8.5

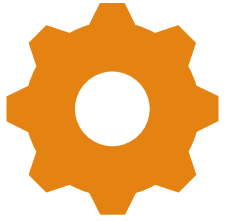


PHP is about as exciting as your toothbrush. You use it every day, it does the job, it is a simple tool, so what? Who would want to read about toothbrushes?

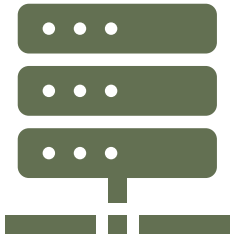
— Rasmus Lerdorf —

AZ QUOTES

# Pourquoi PHP?



Le passage des pages  
statiques vers les  
pages dynamiques



PHP s'exécute du côté  
serveur

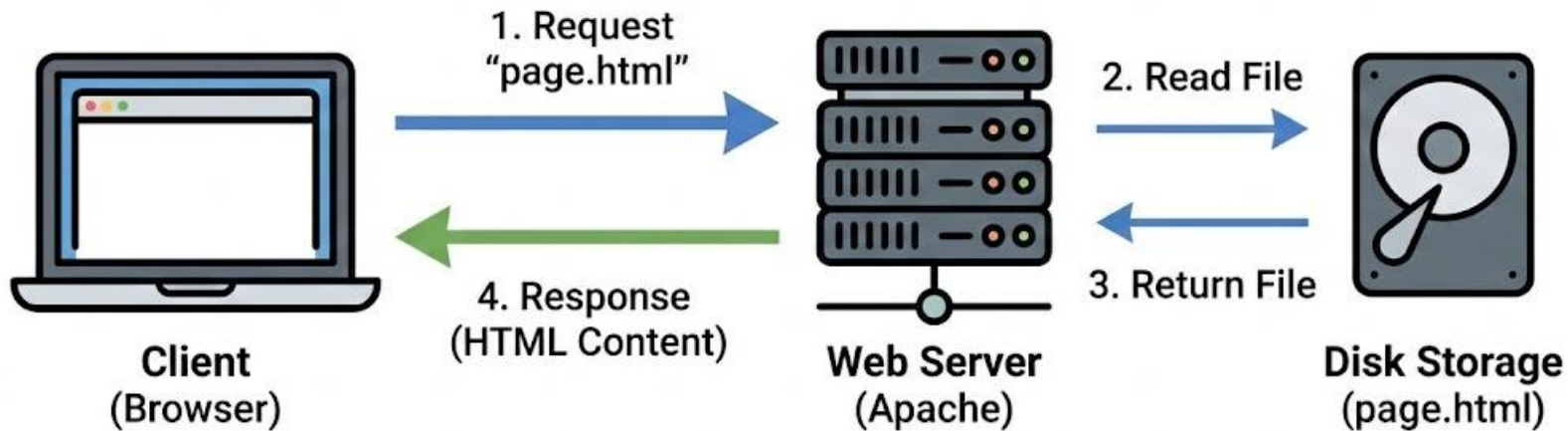


Open source



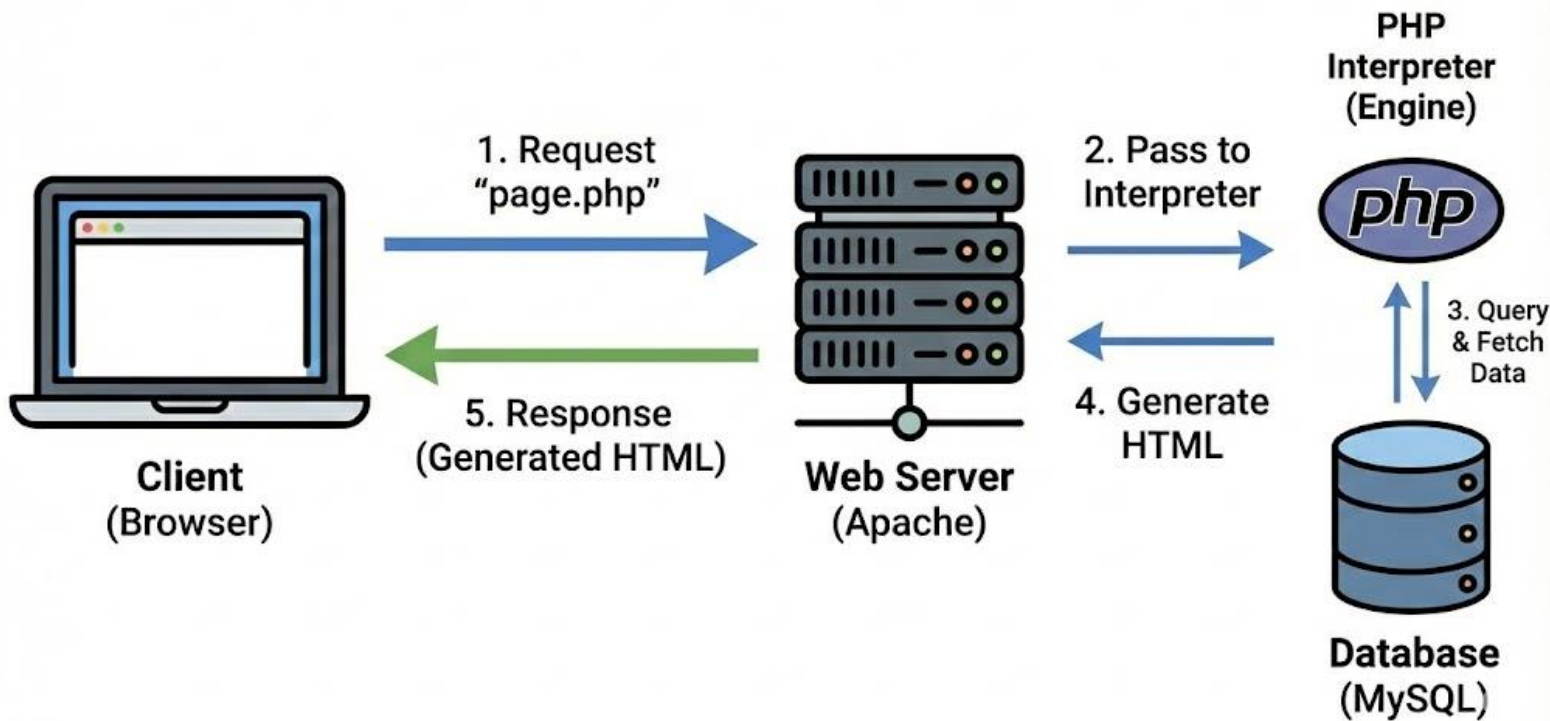
Il offre toutes les fonctionnalités  
nécessaires pour faire des  
applications web complètes (BD,  
session, sécurité, crypto...)

# Pages statiques (html)



1. Envoyer une requête à un serveur web via le navigateur
2. Localiser le serveur et la page
3. Envoyer le code html de la page
4. Interprétation du code par le navigateur

# Page dynamique (html + PHP)



1. Envoyer la requête
2. Localiser le serveur et le fichier demandé
3. Exécuter le fichier demandé par le serveur web (Apache)
4. Générer le code HTML
5. Envoyer le résultat HTML au client web
6. Interprétation du résultat par le serveur web

# Environnement de travail

Propositions:

- Serveur web Apache
- Serveur de base de données
- Éditeur de code PHP
- GUI pour l'administration de la BD
- Navigateur

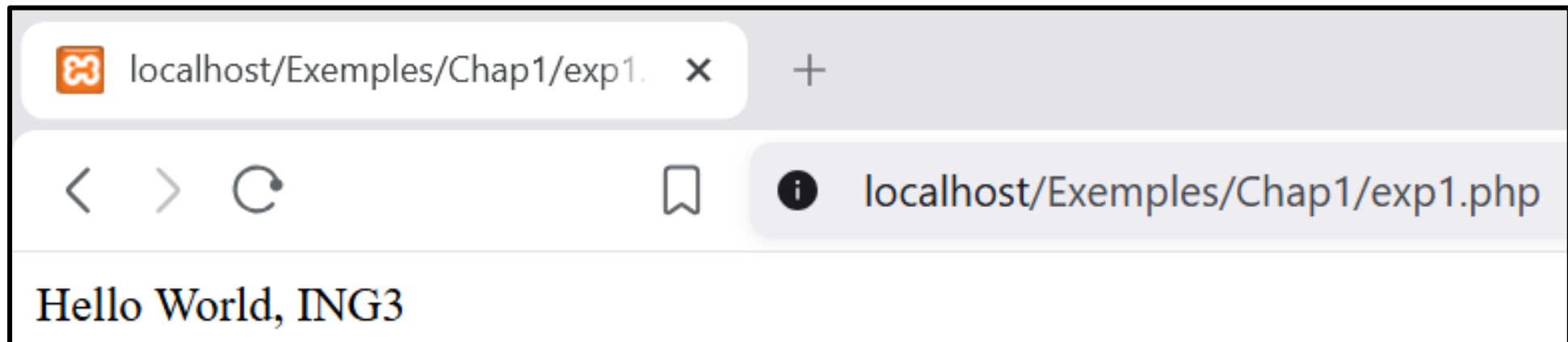


# First web-App: Le fameux « hello world »

1st Script: Hello World Ing3

```
<?php
    // affichage d'un simple message de bienvenue
    echo "Hello world, ING3";

?>
```



## II. Éléments de programmation

---

# Éléments de programmation

---

## Les variables et constantes

- Déclaration et initialisation
- Variables et types de données
- Constantes (utilisateurs et prédéfinies)
- Tableaux

# Éléments de programmation

---

## Les instructions

- Commentaires
- Opérateurs
- Tests : if, switch
- Boucles : for, while, do ...while
- Instructions break et continue

# Déclaration et initialisation des variables

---

- Zone ayant un nom pouvant stocker des valeurs qui peuvent changer
- Précédée par le symbole '\$'
- Pas de déclaration explicite
- Initialisée à l'aide du symbole '='  
(\$variable = expression)

# Variables et types de données

PHP faiblement typé (variable peut varier au fur et à mesure des affectations successives)

## Exemples

- Variable entière
  - \$N=5; (Décimal)
  - \$z=0123 (Octal)
  - \$t=0xFF34 (Hexadécimal)
- Variable réelle
  - \$x=7.25;
  - \$y=1.4e10;
- Variable texte
  - \$var='jour';
  - \$var="bon";

L'affectation consiste à affecter la valeur d'une expression à une variable.

**\$variable = expression**

L'expression peut être :

- Constante
- Variable
- Combinaison de constantes et/ou variables

```
<?php
$nom = "Saad" ;
$filiere = " ING 3 ";

echo "bonjour $nom ";
echo "<br/>";
echo " bonjour $nom " . "je suis un etudiant en $filiere";
?>
```

# Variables et types de données

---

## ➤ Type Booléen

```
<?php
$resultat = true;
echo "la valeur de la var resultat = ".$resultat;

$resultat = false;
echo "la valeur de la var resultat = ". $resultat;
?>
```

- Pas de type char. Un caractère est une chaîne de longueur 1
- Possibilité de convertir une variable en un type primitif à l'aide du cast :
  - \$var="123"
  - \$nbre=(int)\$var

# Variables et types de données

---

- Interprétation des variables en PHP les « double quotes » et « single quotes »

```
$message = " ing 3";  
$message2 = 'ing 3';  
  
echo " hello $message <br/>";  
echo 'hello $message <br/>';  
  
echo " hello $message2 <br/>";  
echo 'hello $message2 <br/>';
```

- On peut utiliser \n, \', \", \\, ...

```
echo 'hello $message2 \n  \\ bienvenu à l\'université internationale de Rabat <br/>';  
echo "hello $message2 \n bienvenu à l\'université internationale de Rabat <br/>";
```

# Variables et types de données

---

Quelques fonctions utiles pour les variables

- `empty($var)` renvoie vrai si la variable est vide
- `isset($var)` renvoie vrai si la variable existe
- `unset($var)` détruit une variable
- `gettype($var)` retourne le type d'une variable
- `is_long()`, `is_double()`, `is_string()`, `is_array()`, `is_bool()`, `is_object()`, `is_float()`, `is_numeric()`, `is_integer()`, `is_int()`, ...

```
$nom = "PHP";  
  
echo "la variable nom est vide ?" . empty($nom) . "<br/>";  
echo "la variable nom est initialisée ?" . isset($nom);  
echo "la variable nom est initialisée ?" . gettype($nom);
```

# Constantes

---

- Zones dont le contenu reste fixe
- Ne sont pas précédées par "\$"

Définies par :  
`define("constante",valeur)`

```
define("PI", 3.14);  
echo "PI = " . PI;
```

Définies par :  
`const (PHP 5.3+)`

```
const PI = 3.14;  
echo "PI = " . PI;
```

# Affichage des données

## echo()

- Affiche une ou plusieurs chaînes de caractères
- Aucune valeur de retour
- Peut prendre plusieurs paramètres (séparés par des virgules)

```
// echo()  
echo "Hello", " ", "World";  
echo "Line 1<br>Line 2";
```

## print()

- Affiche une seule chaîne
- Renvoie 1 (peut être utilisé dans des expressions)
- N'accepte qu'un seul argument

```
// print()  
print "Hello World";  
$result = print "Hello";
```

## print\_r()

- Affiche des informations lisibles par l'utilisateur sur une variable
- Principalement utilisé pour déboguer des tableaux et des objets
- Renvoie une chaîne si le deuxième paramètre est vrai

```
// print_r()  
$array = ["name" =>  
"Saad", "age" => 25];  
print_r($array);
```

# Opérateurs

---

## Arithmétique

`+`, `-`, `*`, `/`, `%`, `++`, `--`, `+=`, `-=`, `*=`, `/=`,  
`%=`

## Logique

`&&` (and), `||` (or), `!`, `xor`

## Comparaison

`==`, `<=`, `>=`, `!=`

## Conditionnel

`(condion)?(exp1):(exp2)`

## Opérateur de concaténation

`.` (point)

# Commentaires

---

Partie ignorée par le compilateur.

Permettent de commenter des programmes

Trois types :

- `//` : commentaire en ligne (C++)
- `#` : commentaire en ligne (Perl)
- `/* ... */` : commentaire sur plusieurs lignes (C)

# Tests : if, switch

```
if ($choix_menu == 1) {  
    //instruction1  
}  
elseif ($choix_menu == 2){  
    //instruction2  
}  
elseif ($choix_menu == 3){  
    //instruction3  
}  
else {  
    //return  
}
```

```
$mois = 2;  
if($mois==1){  
    echo "le mois de janvier  
";  
}  
elseif ($mois==2){  
    echo "le mois de février  
";  
}  
else {  
    echo "le mois demandé  
n'existe pas";  
}
```

# Tests : if, switch

```
switch ($choix_menu) {  
case 1 :  
    // traitement 1  
    break; // fin du case 1  
case 2 :  
    // traitement 2  
    break; // fin du case 2  
case 3 :  
    // traitement 3  
    break; // fin du case 3  
default :  
    // cas qui s'exécute par défaut  
}
```

```
$mois = 2;  
switch($mois){  
    case 1:  
        echo "le mois de janvier ";  
        break;  
    case 2:  
        echo "le mois de février ";  
        break;  
    default:  
        echo "le mois demandé n'existe pas";  
        break;  
}
```

# Boucles : for, while, do ...while

---

La syntaxe de la boucle for:

```
for (init itérateur; condition d'arrêt; incréméntation) {  
    // traitement  
}
```

```
for($i=0; $i < 10; $i++){  
    echo "la valeur de l'itérateur = " . $i . "<br>";  
}
```

la valeur de l'itérateur = 0  
la valeur de l'itérateur = 1  
la valeur de l'itérateur = 2  
la valeur de l'itérateur = 3  
la valeur de l'itérateur = 4  
la valeur de l'itérateur = 5  
la valeur de l'itérateur = 6  
la valeur de l'itérateur = 7  
la valeur de l'itérateur = 8  
la valeur de l'itérateur = 9

# Boucles : for, while, do ...while

## Syntaxe de la boucle while

```
while (condition){  
    // traitement  
}
```

```
$note = 1;  
while($note <10){  
    echo "il faut doubler d'effort ";  
    $note +=1;  
}  
  
echo "vous avez une note >= 10";
```

il faut doubler d'effort  
il faut doubler d'effort  
il faut doubler d'effort  
il faut doubler d'effort  
il faut doubler d'effort  
il faut doubler d'effort  
il faut doubler d'effort  
il faut doubler d'effort  
il faut doubler d'effort  
il faut doubler d'effort  
vous avez une note >= 10

# Boucles : for, while, do ...while

## Syntaxe de la boucle do while

```
do {  
    // traitement  
} while(condition)
```

```
$note = 10;  
  
do {  
    echo "il faut doubler d'effort ";  
    $note++;  
} while($note <10);  
echo "vous avez une note >= 10";
```

il faut doubler d'effort  
vous avez une note  $\geq 10$

# Instructions break et continue

---

L'instruction continue permet de sauter une occurrence de la boucle while et for

## Syntaxe de la boucle while

```
while (condition){  
    continue;  
    // traitement  
}
```

```
$note = 1;  
while($note <10){  
    if($note % 2){  
        continue;  
    }  
    $note +=1;  
}  
  
echo "vous avez une note >= 10";
```

# Instructions break et continue

---

L'instruction break permet de quitter une structure itérative et continuer l'exécution du code

## Syntaxe de la boucle while

```
while (condition){  
    break;  
    // traitement  
}
```

```
$note = 1;  
while($note <10){  
    if($note >= 8){  
        break;  
    }  
    $note +=1;  
}  
  
echo "votre note est = $note";
```

# Tableaux

---

- Variable de type array
- Accepte des éléments de tout type
- Les éléments peuvent être de types différents et sont séparés par des virgules

# Tableaux

---

## ➤ Tableau initialisé en utilisant la syntaxe array

```
// déclaration des tableaux array  
$tbl1 = array("rouge", "vert", "bleu");
```

## ➤ Tableau initialisé au fur et à mesure

```
// déclaration des tableaux avec initialisation  
$tbl2[] = "saad";  
$tbl2[] = "ing3";  
$tbl2[] = 2025;  
// déclaration des tableaux avec initialisation et index  
$tbl3[0] = "saad";  
$tbl3[1] = "ing3";  
$tbl3[2] = 2025;
```

# Tableaux

---

Quelques fonctions utiles pour les tableaux:

- `count($tab)` ou `sizeof($tab)` : retourne le nombre d'éléments
- `in_array($var,$tab)` : TRUE si \$var existe dans \$tab
- `range($i,$j)` : retourne un tableau contenant un intervalle de valeurs
- `shuffle($tab)` : mélange les éléments d'un tableau
- `sort($tab)` : trie un tableau par ordre croissant
- `rsort($tab)` : trie un tableau par ordre décroissant

```
echo in_array("saad", $tbl3);  
rsort($tbl1);
```

# Tableaux

---

## ➤ Accès aux éléments du tableau

```
echo $tbl[1];
```

## ➤ Parcours d'un tableau

```
// boucle for
for($i = 0; $i<count($tbl1); $i++){
    echo $tbl1[$i] . "<br/>";
}
// boucle foreach
foreach ($tbl1 as $element){
    echo $element . "<br/>";
}
```

# Tableaux

---

## Quelques fonctions

- `implode($str,$tab)` ou `join($str,$tab)` : retournent une chaîne contenant les éléments du tableau `$tab` joints par la chaîne `$str`
- `explode($delim,$str)` : retourne un tableau dont les éléments résultent du hashage de la chaîne `$str` par le délimiteur `$delim`
- `array_merge($tab1,$tab2,...)` : concatène les tableaux passés en arguments
- `array_rand($tab)` : retourne l'indice d'un élément du tableau au hasard

```
echo join($tbl1, "-");  
echo $tbl1[array_rand($tbl1)];
```

# Tableau associatif

---

Chaque élément du tableau est de type : Clé=>valeur

L'initialisation peut se faire de deux méthodes :

```
// tbl associatif
$tbl = array("noufel"=> 16, "saad" => 17);
// tbl associatif avec initialisation des valeurs
$tbl2['filiere'] = "ing3";
$tbl2['year'] = "2025/2026";
```

# Tableau associatif

---

Parcours d'un tableau associatif (boucle foreach)

```
$tbl2['filiere'] = "ing3";  
$tbl2['year'] = "2025/2026";  
  
foreach($tbl2 as $key=>$elem){  
    echo "$key : $elem <br>";  
}  
  
foreach($tbl2 as $elem){  
    echo $elem." <br>";  
}
```

# Tableau associatif

---

Quelques fonctions utiles pour les tableaux associatifs

- `array_count_values($tbl)` : retourne un tableau contenant les valeurs du tableau `$tbl` comme clés et leur fréquence comme valeur (utile pour évaluer la redondance)
- `array_keys($tbl)` : retourne un tableau contenant les clés du tableau associatif `$tbl`
- `array_values($tbl)` : retourne un tableau contenant les valeurs du tableau associatif
- `array_search($val,$tbl)` : retourne la clé associée à la valeur `$val`

# Tableau associatif

---

Quelques fonctions pour tableaux associatifs ou normaux :

- `reset($tab)` : place le pointeur sur le 1er élément
- `current($tab)` : retourne la valeur de l'élément courant
- `key($tab)` : retourne la clé de l'élément en cours
- `next($tab)` : place le pointeur sur l'élément suivant
- `prev($tab)` : place le pointeur sur l'élément précédent
- `each($tab)` : retourne la paire clé/valeur courante et avance le pointeur
- `count($tab)` : retourne le nombre éléments du tableau

# Exercices d'application

---

## Exercice 1

1. Donner un programme qui calcule la somme de tous les nombres existants dans un tableau ?
2. Donner un programme qui affiche le max, le minimum et la moyenne des valeurs existantes dans un tableau d'entier ?

# Exercices d'application

## Exercice 2

Vous gérez un site e-commerce. Le panier du client est stocké dans un tableau associatif où la **Clé** est le nom du produit et la **Valeur** est un tableau contenant [Prix Unitaire, Quantité]

Exemple:

```
$panier = [  
    "Clavier" => [400, 1],  
    "Souris sans fil" => [150, 3],  
    "Clé USB 64Go" => [100, 2]  
];
```

1. **Calculer le Sous-Total:** Parcourez le panier pour calculer le montant total (Prix x Quantité pour chaque article)
2. **Appliquer une Remise:** Si le sous-total dépasse **500 DH**, appliquez une remise de **10%**
3. **Ajouter la TVA:** Ajoutez **20%** de TVA au montant final (après remise)
4. **Affichage:** Affichez la liste des produits achetés et le **Montant Final à Payer**