PHP
FONDAMENTAUX

Sommaire

Introduction

INTRODUCTION

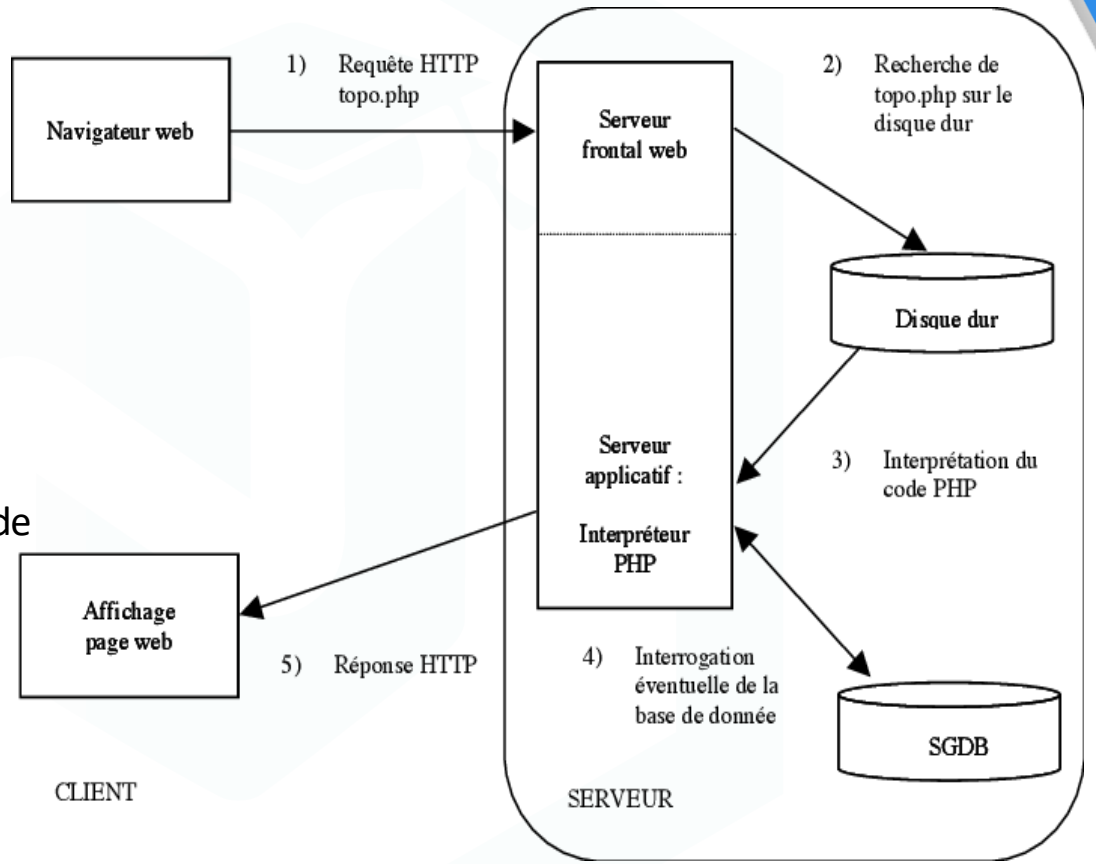
C’est quoi le PHP ?

- PHP : acronyme récursif pour "Hypertext Preprocessor"
- PHP : naissance au milieu de l'année 1990
- PHP est un langage de script
- PHP est interprété du côté du serveur.
- PHP supporte de nombreux SGBD (MySQL, Oracle, PostgreSQL, etc.)
- PHP est langage de programmation impératif, qui permet de réaliser une séquence d’instruction
- PHP est un langage libre
- PHP est adapté à la création de pages web dynamique
- La version actuelle est PHP 8.3

INTRODUCTION

Comment ça marche ?

- Le client web demande une page PHP
- Le serveur web identifie ce fichier dans son système de gestion de fichiers
- Le fichier est transmis au module d'interprétation PHP du serveur
- Le code HTML est généré par l'interpréteur à partir du code PHP
- Le serveur web répond au client



Installation

INSTALLATION

XAMPP

Lorsqu'on développe des applications web en **PHP**, il est crucial de comprendre que **PHP** est un langage **côté serveur**. Cela signifie que pour exécuter du code **PHP** et afficher les résultats dans un navigateur, il est indispensable d'avoir un serveur web capable **d'interpréter ce code**. C'est là qu'intervient **Apache**, qui agit comme un intermédiaire entre l'utilisateur et le serveur, traitant les **requêtes HTTP et exécutant les scripts PHP**.

XAMPP est une distribution facile à installer qui regroupe **Apache**, **MySQL**, **PHP**. Il simplifie grandement le processus d'installation et de configuration.

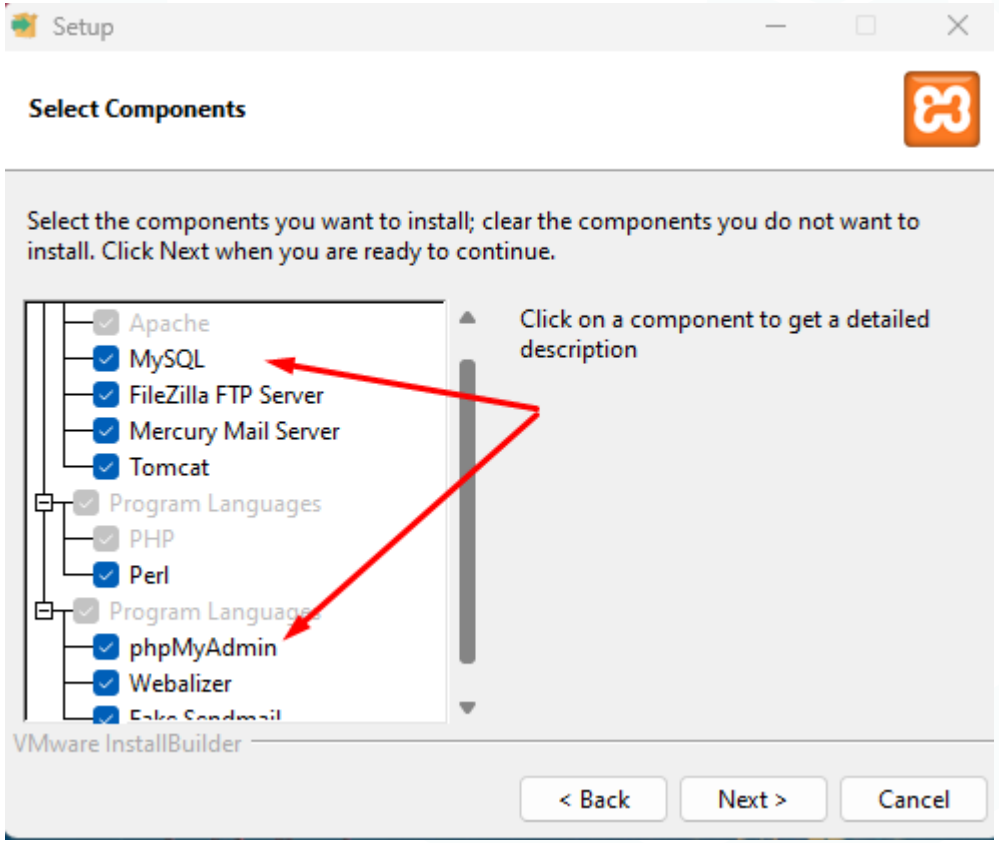
https://www.apachefriends.org/fr/download.html

INSTALLATION

XAMPP

Quelques étapes d'installation :

Vérifiez bien que les éléments suivants soient cochés.

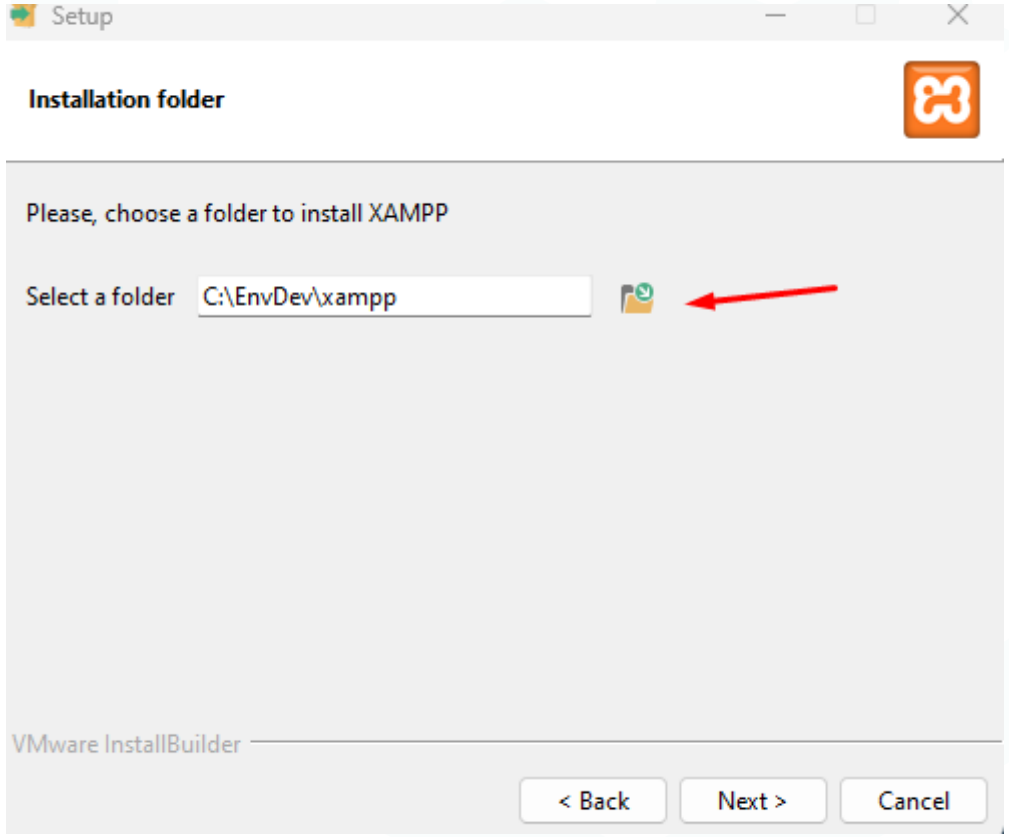


INSTALLATION

XAMPP

Quelques étapes d'installation :

Spécifiez le chemin d'installation.

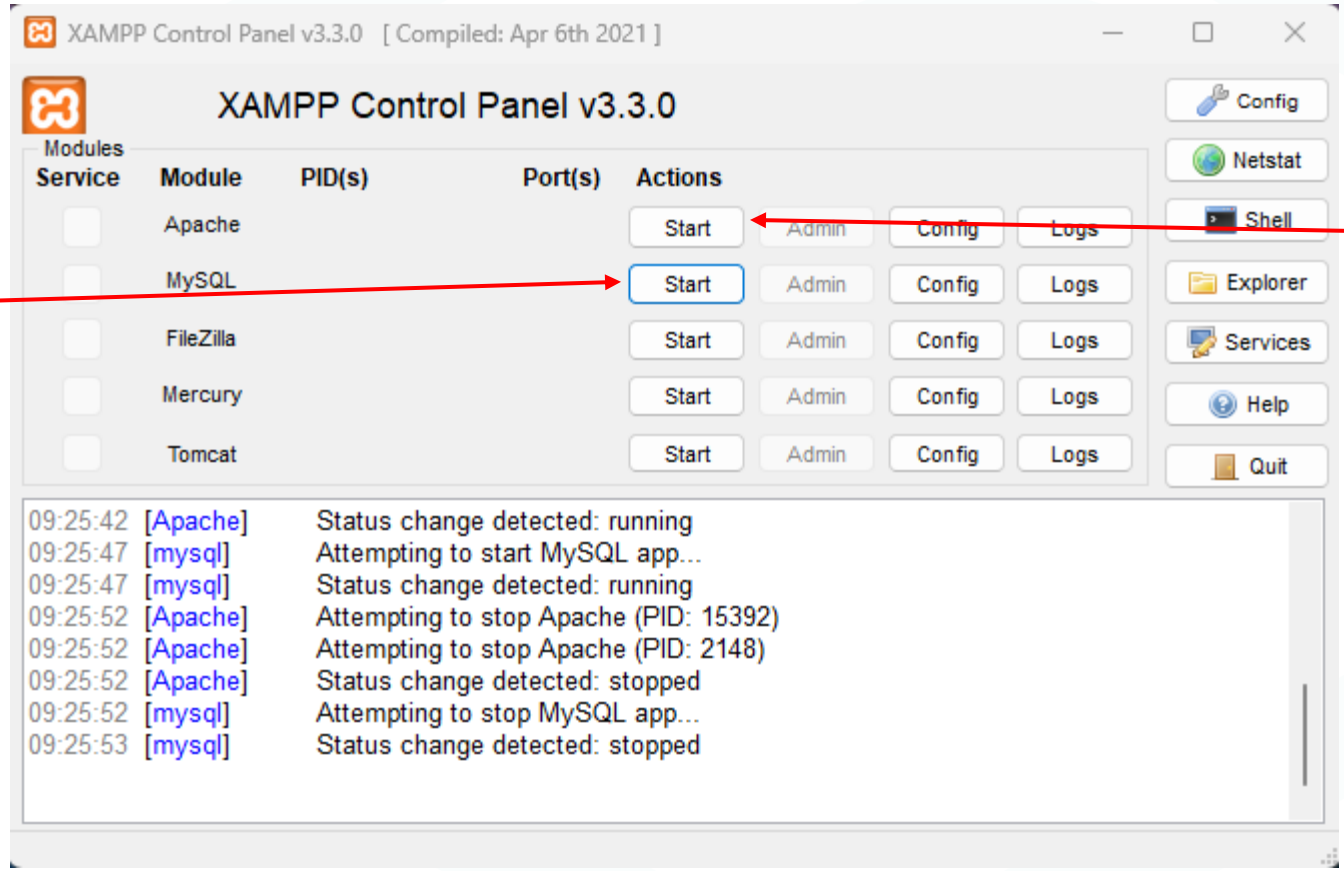


INSTALLATION

XAMPP

Une fois l'installation terminée vous allez arriver sur l'interface du control panel de xampp

Permet de démarrer ou arrêter le service de base de données Mysql



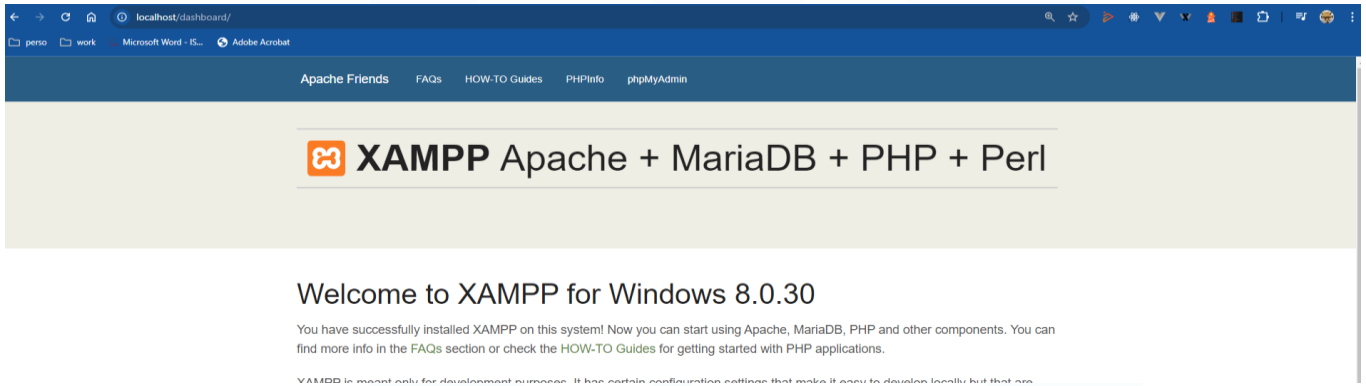
Permet de démarrer ou arrêter le serveur apache

INSTALLATION

XAMPP

Une fois le serveur local apache démarré, cliquez sur le bouton admin.

Cela va ouvrir le Dashboard de votre serveur local apache dans votre navigateur.



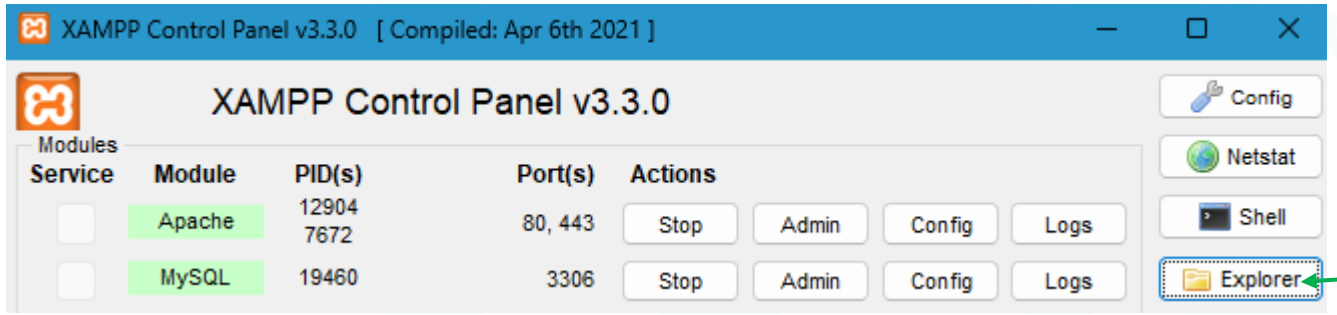
http://localhost/dashboard/

localhost dans l'url, précise que l'on est sur un **serveur local**, qui n'existe que sur notre machine. On peut remplacer **localhost** avec l'IP suivante 127.0.0.1 qui correspond à l'IP du localhost.

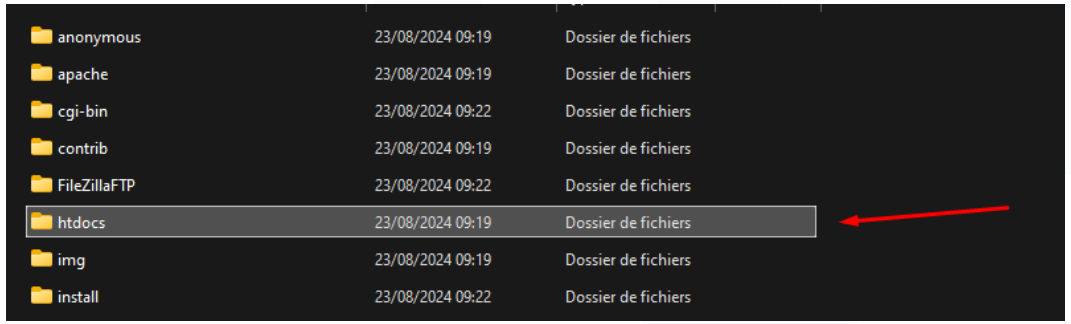
INSTALLATION

XAMPP

Pour créer un fichier **PHP** et stocker vos différents projets qui nécessiteront **apache**, cliquez sur le Bouton explorer du control panel.



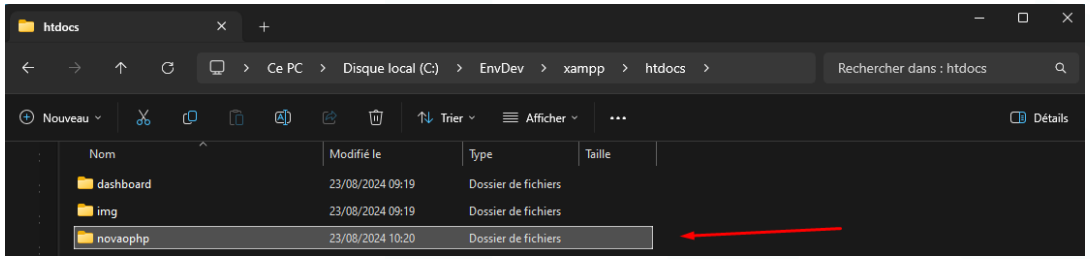
Cela va ouvrir votre explorateur de fichiers, allez ensuite dans le dossier « **htdocs** », c'est dans ce dossier que vous allez stocker vos différents fichiers ou projets.



INSTALLATION

XAMPP

Une fois le dossier **htdocs** ouvert, il est recommandé de créer un nouveau dossier, dans lequel nous allons stocker nos projets et fichiers PHP.



En allant à l'adresse suivante : http://localhost/novaophp/
Nous arrivons sur la page suivante :

Le **serveur** vous affiche un **explorateur** des différents fichiers du dossier **novaophp**.

Tous les fichiers et projets que vous allez rajouter dans ce dossier seront accessibles depuis cette adresse.

Fondamentaux

FONDAMENTAUX

Comment créer un script php ?

Créer un fichier avec l'extension .php ;

Y insérer du code HTML et/ou PHP.

Le code PHP doit être délimité par les balises

- ***<?php CODE PHP ?>***

```
<!DOCTYPE html>
<html lang="fr">
   <head>
          <meta charset="utf-8" />
          <meta name="author" content="Adrien" />
          <title>Mon premier exemple</title>
  </head>
  <body>
          <?php phpinfo(); ?>
 </body>
</html>
```

```
<!DOCTYPE html>
<html lang="fr">
    <head>
          <meta charset="utf-8" />
          <title>Mon second exemple</title>
    </head>
<body>
          <?php echo "Hello World"; //Afficher Hello
World  ?>
</body>
</html>
```

FONDAMENTAUX

Inclusion de script php

En PHP, la méthode **include** est utilisée pour inclure le contenu d'un fichier PHP dans un autre fichier PHP. Cela permet de réutiliser du code et de structurer un projet.

- index.php :

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>include</title>
</head>
<body>
    <?php include("monScript.php");?>
</body>
</html>
```

- monScript.php :

```php
<?php
echo "Hello from monScript.php";
```

- Rendu du fichier index.php :

Hello from monScript.php

Le chemin peut être absolu (complet) ou relatif (par rapport à l'emplacement du fichier actuel).

FONDAMENTAUX

Les Variables

- Pas de déclaration obligatoire (de type) de variables
- Les types sont **boolean**, **integer**, **float**, **double**, **string**, **array** …
- Les noms des variables sont en **alphanumérique**, sensible à la casse et précédés par **$**

```php
<?php

$a = -123; // a negative number
$b  =  1.234;
$c  =  1.2e3;
$d  =  FALSE;
$e = TRUE;
$f = "l'autruche";
$g = "toto et $f";

echo $g;
```

FONDAMENTAUX

Conversion de type

Le **PHP** effectue automatiquement la conversion d'une donnée d'un type à un autre sans nécessiter une instruction de conversion.

```php
<?php
$foo = "0";
echo gettype($foo) . "<br>";// $foo is string (ASCII 48)

$foo += 2;
echo gettype($foo) . "<br>";// $foo is now an integer (2)

$foo = $foo + 1.3;
echo gettype($foo) . "<br>";// $foo is now a double (3.3)

$foo = 5 . "10 Small Pigs";
echo gettype($foo). "<br>"; // $foo is string ("510 Small
Pigs")
?>
```

- En PHP, la concaténation de chaînes de caractères peut se faire à l'aide de l'opérateur de concaténation « . »

FONDAMENTAUX

Les constantes

Les constantes sont des identifiants (noms) qui représentent des valeurs qui ne peuvent pas être modifiées pendant l'exécution du script.

Elles sont définies avec la fonction **define()** et sont utilisées sans le symbole **$**.

```php
<?php

define("PI", 3.14);
echo PI; // Affiche la valeur de la constante PI (3.14)

?>
```

FONDAMENTAUX

Les constantes et variables « magiques »

En PHP, les constantes et variables magiques sont des éléments prédéfinis par le langage, qui ont des noms spéciaux.

Exemples de constantes magiques :

- **__FILE__** : Le chemin complet du fichier PHP en cours.
- **__LINE__** : Le numéro de la ligne courante dans le script.
- **__DIR__** : Le répertoire du fichier PHP en cours.
- **__CLASS__** : Le nom de la classe en cours.

Exemples de variables magiques :

- **$_SERVER** : Une superglobale contenant des informations sur le serveur et l'exécution du script.
- **$_GET** : Une superglobale contenant les données envoyées par la méthode GET.
- **$_POST** : Une superglobale contenant les données envoyées par la méthode POST.
- **$_SESSION** : Une superglobale contenant les variables de session

FONDAMENTAUX

Les opérateurs

- Opérateurs arithmétiques :
  - + ou - ou * ou / ou % ou ++ ou --
- Opérateurs d'affectation
  - = ou += ou -= ou .= ou /= ou =
- Opérateur de concaténation de chaînes :
  - « . »
- Opérateurs de concaténation de tableaux :
  - +
- Opérateurs de comparaison :
  - == ou != ou > ou < ou >= ou <=
- Opérateurs logiques :
  - && ou || ou !

FONDAMENTAUX

Les Tableaux

Les tableaux en PHP sont des structures de données qui peuvent contenir une collection d'éléments. Les tableaux peuvent être indexés numériquement ou associativement.

```php
// Déclaration d'un tableau indexé numériquement
$tableau = array("Pomme", "Banane", "Orange");
// ou en utilisant la syntaxe courte (à partir de PHP 5.4)
$tableau = ["Pomme", "Banane", "Orange"];
```

Dans un tableau indexé numériquement, chaque élément est associé à un index numérique commençant par zéro (0, 1, 2, etc.).

```php
// Déclaration d'un tableau associatif
$personne = array("nom" => "John", "âge" => 30, "ville" => "Paris");
// ou avec la syntaxe courte
$personne = ["nom" => "John", "âge" => 30, "ville" => "Paris"];
```

Dans un tableau associatif, chaque élément est associé à une clé (ou un nom)

FONDAMENTAUX

Les Tableaux ajout de valeur

- Accès aux éléments d'un tableau par index numérique :

```
echo $tableau[0]; // Affiche "Pomme"
```

- Accès aux éléments d'un tableau par clé associative :

```
echo $personne["âge"]; // Affiche 30
```

- Ajouter un élément dans un tableau par index numérique :

```
$tableau[] = "Fraise"; // Ajoute "Fraise" à la fin du tableau
```

- Ajouter un élément dans un tableau par clé associative :

```
$personne["profession"] = "Développeur"; // Ajoute une nouvelle clé
```

- Suppression d'un élément d'un tableau par index :

```
unset($tableau[1]); // Supprime l'élément ayant l'index 1 ("Banane")
```

- Suppression d'un élément d'un tableau clé :

```
unset($personne["âge"]); // Supprime la clé "âge" et sa valeur associée
```

FONDAMENTAUX

Les Tableaux multidimensionnel

Un tableau multidimensionnel est un tableau qui contient d'autres tableaux en tant qu'éléments.
En PHP, cela signifie que vous avez un tableau dont chaque élément peut lui-même être un tableau.

```php
$got = [
    "Lannister" => [
        "Pere" => "Tywin Lannister",
        "Soeur" => "Cersei Lannister",
        "Frere" => "Jaime Lannister"
    ],
    "Stark" => [
        "Pere" => "Robb Stark",
        "Mere" => "Catelyn Stark",
        "Fils" => "Jon Stark"
    ]
];

echo $got["Lannister"]["Pere"]; // Affiche "Tywin Lannister"
```

FONDAMENTAUX

Les structures conditionnelles

- Structure **if** :

```php
$note = 15;
if ($note >= 10) {
    echo "Bravo ! Vous avez réussi l'examen.";
} else {
    echo "Désolé, vous devez reprendre l'examen.";
}
```

- Structure **if elseif** :

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

- Structure **ternaire** :

```php
$age = 20;
$isMmajeur = ($age >= 18) ? "Majeur" : "Mineur";
echo "Vous êtes $est_majeur.";
```

L'opérateur ternaire
condition ? valeur_si_vrai : valeur_si_faux
permet une forme concise de la structure **if...else**

FONDAMENTAUX

Les structures conditionnelles

```php
$jour = "Mercredi";

switch ($jour) {
    case "Lundi":
    case "Mardi":
    case "Mercredi":
    case "Jeudi":
    case "Vendredi":
        echo "C'est un jour de semaine.";
        break;
    case "Samedi":
    case "Dimanche":
        echo "C'est le week-end.";
        break;
    default:
        echo "Ce n'est pas un jour valide.";
}
```

La structure **switch** :

- La valeur de l'expression est comparée à chaque **case**.
- Lorsqu'une correspondance est trouvée les instructions sont exécutées.
- Le mot-clé **break** est utilisé pour sortir de la structure avoir exécuté les instructions correspondantes à une **case**.
- Si aucun **des case** ne correspond à la valeur de l'expression, les instructions du **default** sont exécutées (**s'il est présent**).

FONDAMENTAUX

Les structures itératives

- Boucle while :

```
$compteur = 1;

while ($compteur <= 5) {
    echo "Tour $compteur
<br>";
    $compteur++;
}
```

La boucle **while** exécute un bloc de code tant qu'une condition spécifiée est vraie.

- Boucle for :

```
for ($i = 1; $i <= 5; $i++) {

    echo "Iteration $i <br>";

}
```

La boucle **for** est utilisée pour exécuter un bloc de code un nombre spécifié de fois.

FONDAMENTAUX

Les structures itératives

- Boucle do while :

```php
$nombre = 10;

do {
    echo "$nombre <br>";
    $nombre--;
} while ($nombre > 0);
```

La boucle **do...while** est similaire à while, mais **elle garantit l'exécution** du bloc de code au moins une fois

- Boucle foreach :

```php
$fruits = ["Pomme",
"Banane"];

foreach ($fruits as $fruit)
{
    echo "$fruit <br>";
}
```

La boucle **foreach** est spécialement conçue pour **parcourir les éléments** d'un tableau

FONDAMENTAUX

Les fonctions

Les fonctions en **PHP** permettent d'organiser et **de réutiliser du code** en le regroupant dans des blocs isolés et nommés. Elles sont déclarées avec le **mot-clé function** suivi du nom de la fonction et peuvent accepter des paramètres en entrée pour effectuer des opérations spécifiques.

```php
function nomDeLaFonction($parametre1, $parametre2, ...) {
    // Bloc de code à exécuter
    // Utilisation des paramètres pour effectuer des opérations
    return $resultat; // Optionnel : renvoie une valeur en sortie
}
```

- Exemple :

```php
function multiplier($a, $b) {
    $resultat = $a * $b;
    return $resultat;
}

// Appel de la fonction et récupération de la valeur de retour
$produit = multiplier(5, 3);
echo "Le produit est : $produit";
```

FONDAMENTAUX

Les fonctions sur les chaînes de caractères

- int **strlen**(string $ch) longueur de $ch
- int **strcmp**(string $ch1, string $ch2) compare deux chaînes
- string **trim**(string $ch) supprime les espaces en début/fin
- string **ltrim**(string $ch) supprime les espaces au début
- string **rtrim**(string $ch) supprime les espaces à la fin
- string **ucfirst**($ch) la première lettre en majuscule
- string **ucwords**($ch) la première lettre de chaque mot
- string **strtolower**(string $ch) tout en minuscules
- string **strtoupper**(string $ch) tout en majuscules
- string **nl2br**(string $string) remplace \n par

FONDAMENTAUX

Les fonctions sur les chaînes de caractères

- int **strpos**(string $ch1, mixed $ch2) :
  - vérifie si $ch2 est sous-chaîne de $ch1 et
  - retourne la position de la 1ere occurrence de $ch1 dans $ch2
- int **strrpos**(string $ch1, mixed $ch2) :
  - vérifie si $ch2 est sous-chaîne de $ch1 et
  - retourne la position de la dernière occurrence de $ch1 dans $ch2
- mixed **str_replace**(mixed $ch1, mixed $ch2, mixed $ch3) :
  - toutes les occurrences de $ch2 dans $ch1 ont été remplacé par $ch3

FONDAMENTAUX

Les fonctions sur les tableaux

- int **count**(mixed $tab) :
  - compte le nombre d'éléments
- array **explode**(string $d, string $ch):
  - retourne le tableau des sous-chaînes de $ch divisé pas $.
- int **array_push**(array &$tab, mixed $v1[, mixed $v2...]) :
  - Empile $v1, $v2 dans &$tab
- mixed **array_pop**(array $tab) :
  - dépile le tableau
- void **unset**(mixed $var):
  - supprime un élément du tableau
- array **array_unique**(array $tab) :
  - supprime les doublons
- array **array_merge**(array $t1[,array $t2,array $t3...]) :
  - fusionne les tableaux
- array **array_intersect**(array $t1[,array $t2,...]):
  - retourne l'intersection des arguments
- array **array_diff**(array $t1, array $t2[, array $t3...]) :
  - retourne la différence ensembliste entre $t1 et $t2...

FONDAMENTAUX

Les fonctions sur les tableaux

Les fonctions pour trier :

- bool **asort**(array $tab) :
  - les valeurs du tableau par ordre croissant
- bool **arsort**(array $tab) :
  - les valeurs du tableau par ordre décroissant
- bool **ksort**(array $tab) :
  - les clefs du tableau par ordre croissant
- bool **krsort**(array $tab):
  - les clefs du tableau par ordre décroissant

FONDAMENTAUX

Les sessions

Les **sessions** en PHP permettent de stocker des informations utilisateur spécifiques sur le serveur, accessibles et utilisables tout au long de la visite de l'utilisateur sur le site. Elles offrent un moyen de maintenir des données d'une page à l'autre sans avoir besoin de les passer via des formulaires ou des URL.

Pour utiliser les sessions en PHP, la première étape est de démarrer une session à l'aide de la fonction **session_start()**

Une fois la session démarrer nous pouvons accéder à la variable magique **$_SESSION** qui est un tableau.

```php
session_start();
$_SESSION['utilisateur'] = 'John';
$_SESSION['role'] = 'admin';

echo $_SESSION['utilisateur']; // Affiche 'John'
echo $_SESSION['role']; // Affiche 'admin'
```

Fermer une session :

Pour détruire complètement la session, utilisez **session_destroy()**.

Cela effacera toutes les données de la session.

FORMULAIRES

GESTION DES FORMULAIRES

Principe

Lorsque l'utilisateur clique sur le bouton d'envoi (**submit**), une requête **HTTP** est envoyée au serveur à destination du script désigné par l'attribut **action** de la balise **<form>**

La requête contient les associations :

- nom du champ ↔ valeur

Les associations se trouvent :

- soit dans l'enveloppe **HTTP** si la méthode **POST** est utilisée
- soit dans **l'URL** s'il s'agit de la méthode **GET**

GESTION DES FORMULAIRES

Principe

- Les valeurs du formulaire sont stockées sur le serveur dans les tableaux associatifs appelés **$_POST** ou **$_GET** (et dans **$_REQUEST**)
- Les clés de ces tableaux sont les noms associés aux champs par l'attribut **name**
- Les valeurs associées aux clés sont les informations saisies par l'utilisateur
- Lire les valeurs

GESTION DES FORMULAIRES

Variable magique $_GET

$_GET est une superglobale en PHP qui récupère des données envoyées à un script PHP via la méthode **HTTP GET**. Cette méthode est couramment utilisée pour passer des paramètres dans **l'URL**.

- Formulaire html :

```
<form action="traitement.php" method="get">
    <label for="nom">Nom :</label>
    <input type="text" id="nom" name="nom"><br>
    <label for="email">Email :</label>
    <input type="email" id="email" name="email"><br>
    <input type="submit" value="Envoyer">
</form>
```

- Traitement.php

```
$nom = $_GET["nom"];
$email =( ($_GET["email"];
echo  $nom ." ". $email ;
```

GESTION DES FORMULAIRES

Variable magique $_GET

$_POST est une superglobale en PHP qui récupère des données envoyées à un script PHP via la méthode **HTTP POST.** Cette méthode est couramment utilisée pour soumettre des données depuis un formulaire HTML.

- Formulaire html :

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

- Traitement.php

```
$nom = $_POST["nom"];
$email = $_POST["email"];
echo  $nom . " " . $email;
```

GESTION DES FORMULAIRES

Vérification des champs

Il est important de valider les données du formulaire pour s'assurer qu'elles correspondent aux attentes.

isset() et **empty()** sont deux fonctions en **PHP** utilisées pour vérifier et manipuler les variables :

- **isset(**$var**)** :
  - Vérifie si une variable est définie et si elle n'est pas **NULL**. Renvoie **true** si la variable existe et n'est pas NULL, sinon renvoie **false**.

```php
if (isset($_POST["name"])) {
    $name = $_POST["name"];
    echo "La variable \$nom est définie.";
}
```

- **empty(**$var**)**:
  - Vérifie si une variable est vide.
    Renvoie **true** si la variable est vide, sinon renvoie **false**.

```php
if (empty($_POST["name"])) {
    echo "La variable \$_POST['$nom'] est vide.";
}
```

GESTION DES FORMULAIRES

Sanitization (Nettoyage des données)

Sécuriser les données provenant des utilisateurs utiliser des techniques de **sanitization** (nettoyage des données)

- Échappement : Utilisez des fonctions comme **htmlspecialchars()** pour échapper les données avant de les afficher dans du HTML afin d'éviter les **attaques XSS**.
- Validation : Validez les données pour vous assurer qu'elles correspondent au format attendu.

```
$entree_utilisateur = "<script>alert('Attaque XSS');</script>";
// Échapper les données pour affichage sécurisé dans du HTML
$entree_securisee = htmlspecialchars($entree_utilisateur);
echo $entree_securisee; // Affichera le texte, pas l'alerte XSS
```

En combinant **isset()**, **empty()**, et les techniques de **sanitization**, vous pouvez vérifier l'existence des variables, leur contenu et nettoyer les données provenant des utilisateurs pour éviter les vulnérabilités de sécurité dans vos applications PHP