Introduction PHP

Version	V0
Type	Technique
Date de création	@14 février 2024 09:06
Dernière modification	@16 février 2024 09:33

🔍 Qu'est-ce que PHP ?

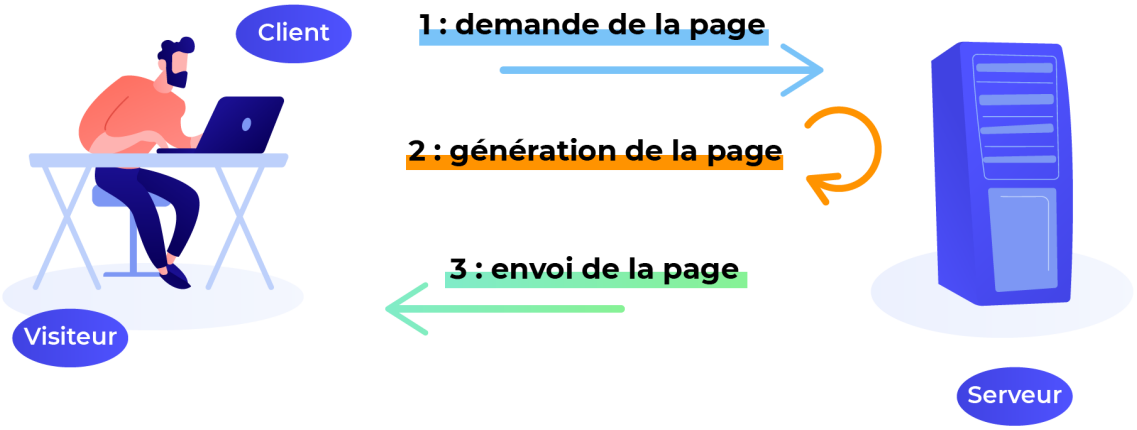
PHP est un langage de script exécuté sur le serveur, ce qui signifie que le code PHP est traité sur le serveur web avant que le résultat ne soit envoyé au navigateur du client sous forme de HTML. Cela permet de créer des pages web dynamiques qui peuvent changer ou s'adapter en fonction des données fournies par l'utilisateur ou stockées sur le serveur.

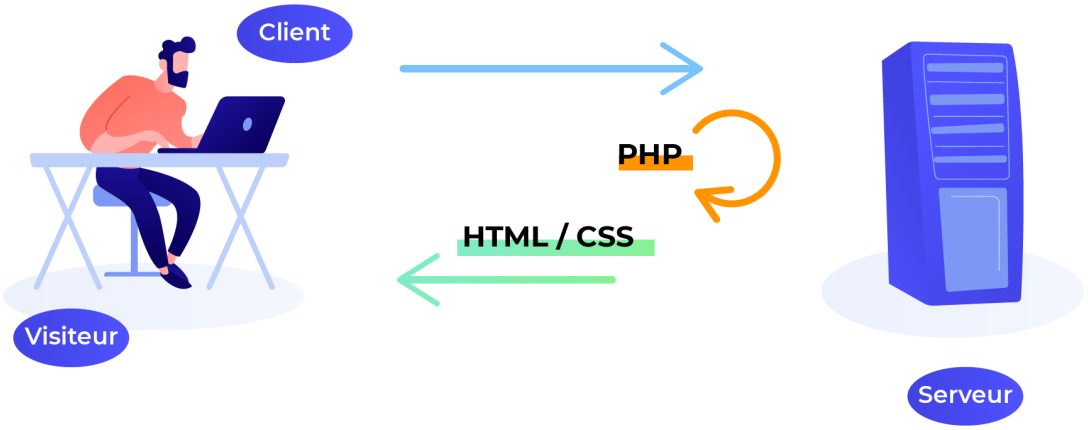
🦏 Qu’est ce que WAMP ?

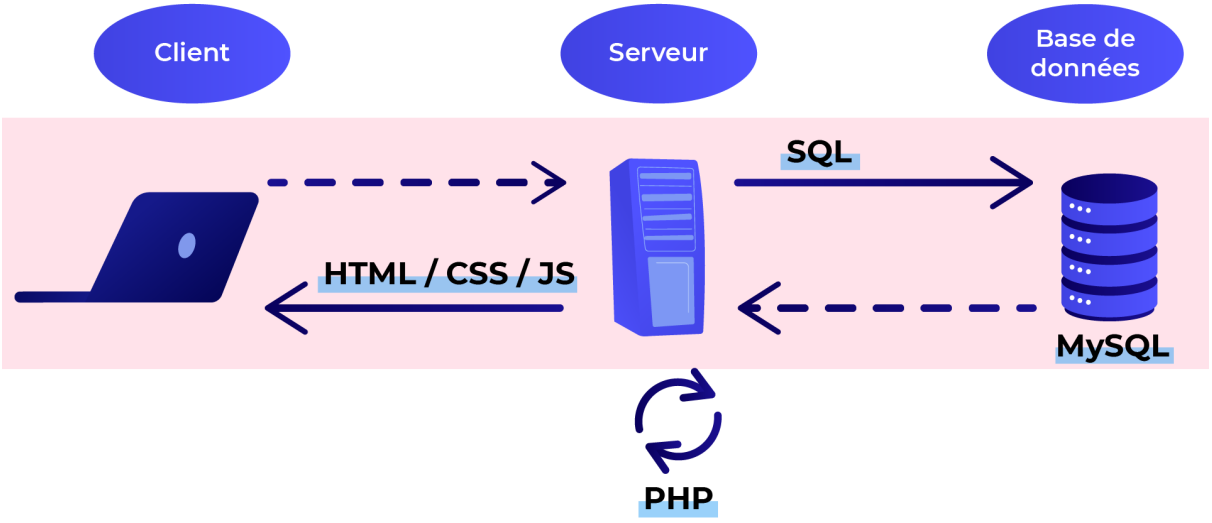
WAMP est un acronyme pour Windows - Apache - MySQL - PHP, désignant un logiciel permettant de mettre en place un serveur local sur un système Windows. Ce paquet regroupe quatre éléments clés : le système d'exploitation Windows, le serveur web Apache, le système de gestion de bases de données MySQL, et les langages de programmation PHP.

WAMP est largement utilisé car il permet aux développeurs de créer un environnement de serveur web sur leur propre ordinateur, sans nécessiter une connexion internet ou un hébergement web externe.

🔁 Le traitement de PHP







✏️ Syntaxe PHP

Déclarer une variable :
En PHP, une variable est déclarée en utilisant le signe

`$` suivi du nom de la variable. Les noms de variables sont sensibles à la casse.

```
$variableSuper = "valeur"; // JS : let variable = "valeur"
```

Déclarer une constante :
Les constantes sont similaires aux variables, mais une fois définies, leur valeur ne peut pas être modifiée. Les constantes sont définies en utilisant la fonction `define()`.

```
define("CONSTANTE", "valeurConstante");
```

Déclarer une fonction :
Une fonction en PHP est un bloc de code qui peut être réutilisé en l'appelant par son nom. Les fonctions sont déclarées en utilisant le mot-clé `function`.

```
function nomFonction() {
    // code à exécuter
}
```

Faire une condition :
Les conditions en PHP permettent de réaliser différentes actions en fonction de conditions spécifiques, utilisant les instructions `if`, `else`, et `elseif`.

```
if (condition) {
    // code si la condition est vraie
} elseif (autreCondition) {
    // code si l'autre condition est vraie
} else {
    // code si aucune des conditions précédentes n'est vraie
}
```

Les différentes boucles :

- **Boucle `for` :** Répète le bloc de code un nombre défini de fois.

  ```
  for ($i = 0; $i < 10; $i++) {
      // code à exécuter pour chaque itération
  }
  ```

- **Boucle `while` :** Exécute le bloc de code tant que la condition spécifiée est vraie.

  ```
  while (condition) {
      // code à exécuter
  }
  ```

- **Boucle `foreach` :** Spécialement conçue pour parcourir les éléments d'un tableau.

```
// Pour les tableaux classiques
foreach ($tableau as $valeur) {
    // code à exécuter pour chaque élément
}

foreach ($tableau as $valeur) {
    echo($valeur['key']);
}

// Pour les tableaux associatifs
foreach ($tableau as $key => $valeur) {
    // code à exécuter pour chaque élément
}
```

Les tableaux :
Les tableaux en PHP sont des variables qui peuvent stocker plusieurs valeurs. Il y a les tableaux indexés numériquement et les tableaux associatifs où chaque valeur est associée à une clé unique.

- **Tableau indexé :**

```
$tableau = array("élément1", "élément2", "élément3");

$tableau = ["élément1", "élément2", "élément3"];
```

- **Tableau associatif :**

```
$tableauAssociatif = array("cle1" => "valeur1", "cle2" => "valeur2");

$tableauAssociatif = [
    "cle1" => "valeur1",
    "cle2" => "valeur2"
];

tableauAssociatif.key

$tableauAssociatif['key']
```

Les superglobales :
PHP dispose de plusieurs superglobales, qui sont des variables globales intégrées disponibles dans tous les contextes du script. Les plus courantes sont
`$_GET`, `$_POST`, `$_SESSION`, `$_COOKIE`, `$_FILES`, `$_ENV`, `$_REQUEST`, et `$_SERVER`.

📝 Formulaires

Méthode GET

Transmission des données : Les données du formulaire sont ajoutées à l'URL. Elles sont attachées à l'URL avec un `?`, suivi par la paire clé/valeur. Par exemple, si votre formulaire contient un champ nommé "nom" et que vous entrez "Jean", l'URL pourrait ressembler à `http://www.exemple.com/formulaire.php?nom=Jean`.

Limitation de la longueur : Étant donné que les données sont ajoutées à l'URL, il y a une limite à la quantité de données que vous pouvez envoyer, qui dépend du navigateur et du serveur web, mais en général, elle est assez petite.

Sécurité : Les données envoyées par GET sont visibles dans l'URL, ce qui n'est pas sécurisé, surtout pour des informations sensibles comme les mots de passe ou les données personnelles.

Méthode POST

Transmission des données : Les données du formulaire sont incluses dans le corps de la requête HTTP, pas dans l'URL. Cela signifie qu'elles ne sont pas visibles dans la ligne d'adresse du navigateur.

Limitation de la longueur : POST n'a pas de limitation de taille pour la quantité de données que vous pouvez envoyer, ce qui le rend plus adapté pour le transfert de grandes quantités de données.

Sécurité : Comme les données ne sont pas exposées dans l'URL, POST est plus sécurisé que GET et est préféré pour l'envoi de données sensibles.

Ce qui se passe côté serveur

Traitement des données : Que vous utilisiez GET ou POST, le serveur web recevra les données et les passera à votre script PHP. En PHP, vous pouvez accéder aux données envoyées par GET via la superglobale `$_GET`, et celles envoyées par POST via `$_POST`.

Réponse du serveur : Après le traitement des données, votre script PHP peut générer une réponse, qui peut être une page HTML, une redirection, un message d'erreur, ou tout autre type de réponse que le serveur web est capable de retourner au client.

ℹ️ La principale différence entre GET et POST réside dans la façon dont les données sont envoyées et traitées. GET est plus adapté pour les requêtes de données non sensibles, tandis que POST est utilisé pour envoyer des données de manière plus sécurisée et sans limitation de taille.

POO en PHP

ℹ️ Quelques points diffèrent de la POO en JS, revenons dessus :

En POO avec PHP, les mots-clés comme `static`, `public`, `private`, etc, sont utilisés pour définir la visibilité des propriétés et des méthodes d'une classe, ainsi que leur comportement en relation avec la classe elle-même et les instances de celle-ci.

`public`

- **Visibilité :** Les propriétés ou méthodes déclarées comme `public` sont accessibles de partout, ce qui signifie que n'importe quel code extérieur à la classe peut y accéder.

- **Exemple d'utilisation :** Vous utilisez `public` pour des méthodes que vous voulez exposer comme les méthodes “getter” et “setter”.

`private`

- **Visibilité :** Les propriétés ou méthodes déclarées comme `private` sont accessibles uniquement à l'intérieur de la classe où elles sont déclarées.
- **Exemple d'utilisation :** `private` est utilisé pour cacher les détails d'implémentation internes de la classe, pour des fonctions auxiliaires ou des variables qui ne doivent pas être accessibles par le code externe.

`protected`

- **Visibilité :** Les propriétés ou méthodes `protected` sont accessibles à l'intérieur de la classe où elles sont déclarées, ainsi que par les classes qui en héritent.
- **Exemple d'utilisation :** `protected` est souvent utilisé lorsque vous créez une base de classe qui est destinée à être étendue, et vous voulez que seules les sous-classes puissent accéder à ces propriétés ou méthodes.

`static`

- **Comportement :** Une propriété ou méthode déclarée comme `static` appartient à la classe elle-même plutôt qu'à une instance spécifique de la classe.
- **Accès :** Vous pouvez accéder à une propriété ou méthode statique directement en utilisant le nom de la classe (par exemple, `ClassName::staticMethod()` ou `ClassName::$staticProperty`) sans avoir besoin de créer une instance de la classe.
- **Exemple d'utilisation :** `static` est utile pour des méthodes utilitaires ou des variables qui doivent être partagées entre toutes les instances d'une classe, comme un compteur d'instances ou une méthode de fabrique.

```php
<?php

class Vehicule
{
    private $color;
    public $fuel;
    protected $km;

    public function __construct($color, $fuel, $km)
    {
        $this->color = $color;
        $this->fuel = $fuel;
        $this->km = $km;
    }

    public function getColor()
    {
        return $this->color;
    }

    public function setColor($color)
    {
        $this->color = $color;
    }

    protected function helloWorld()
    {
```

```
        return 'Hello World';
    }
}
```