Module PHP

Fouziya Kouddane

SOMMAIRE

INTRODUCTION

C'est Rasmus Leford qui a créé le langage PHP en 1994 suite à un travail personnel qui désignait à l'époque Personnal Home Page. Puis en 1997 Zeev Suraski et Andi Gutmans ont entamé des travaux d'amélioration du langage PHP. La première version officielle est alors nommée PHP3 dont l'acronyme récursif désigne désormais PHP Hypertext Preprocessor.

PHP est inspiré principalement de trois langages de programmation, à savoir le langage C, Perl et Java (dont le model objet a servi de base pour développer PHP5). D'autres langages comme Shell et C++ ont aussi eu de l'influence sur certains aspects du PHP.

Nous sommes actuellement à la version 8 sortie en novembre 2020.

QU'EST CE QUE PHP

PHP est un langage de programmation coté serveur. Contrairement à Javascript qui s'exécute côté client, PHP s'exécute entièrement sur le serveur qui héberge le site Web. Son exécution sera moins rapide que celle du Javascript à cause du temps que prennent les requêtes pour parvenir au serveur et la réponse renvoyée au navigateur.

Quelques avantages du php:

- Code source confidentiel: Le code source n'est jamais visible par le client, ce qui permet de manipuler des données confidentielles.

Open source: PHP est un langage de programmation libre de droit.

Multi-plateform: PHP s'exécute sur des serveurs d'applications que l'on peut installer sur de nombreux systèmes d'exploitation (Unix/Linux, Windows, Mac OS, BSD...)

Sa syntaxe simple et intuitive

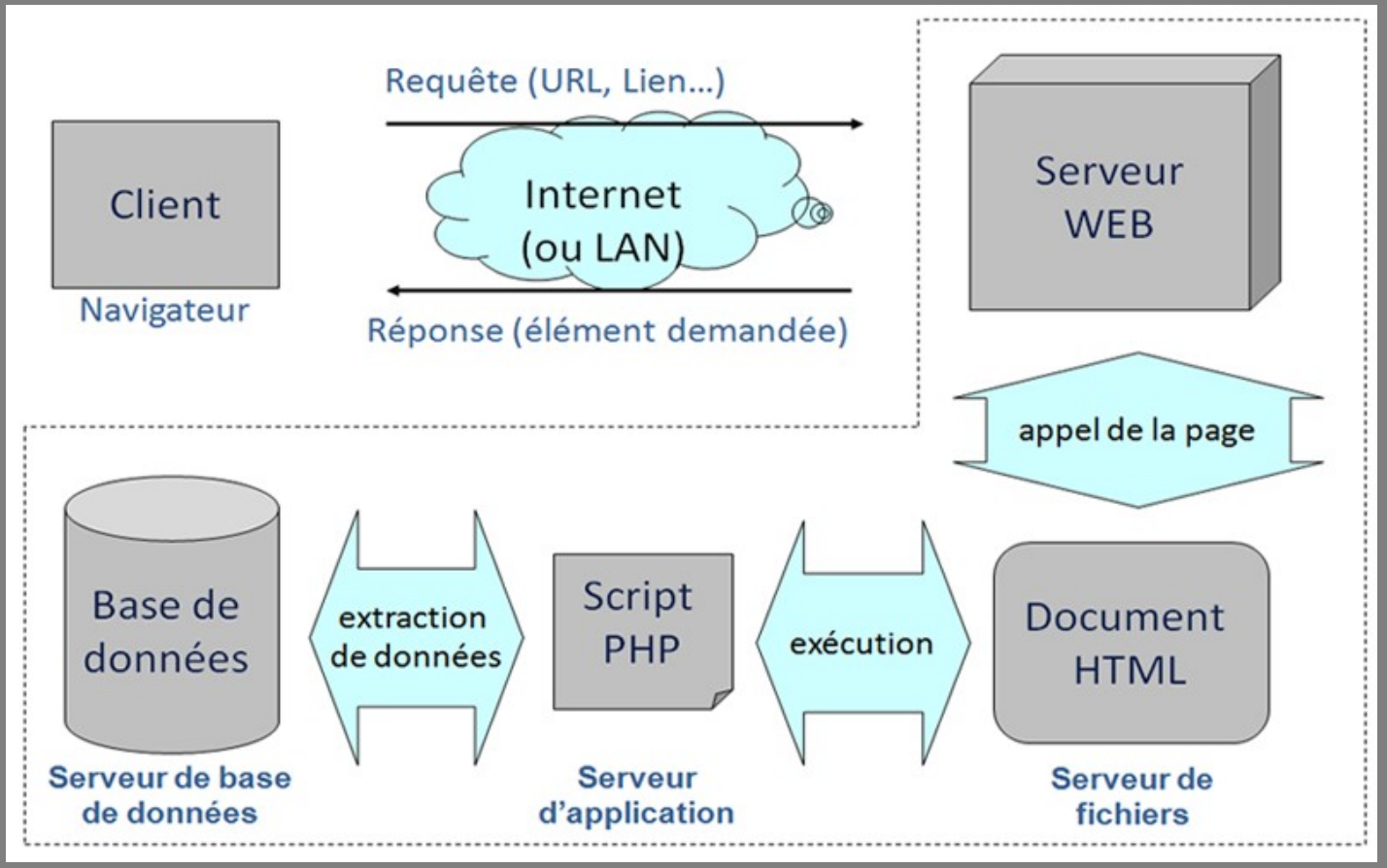
Les bases de données: il peut cependant interagir avec presque tous les SGDB connus.

Richesse fonctionnelle: PHP prend en charge de nombreuses bibliothèques

La documentation : www.php.net.

Frameworks diversifiées: Zend Framework, Symfony, CodeIgniter, Laravel...

L'ENVIRONNEMENT DE PHP



C'est pour cela que nous utilisons WampServer.

L'ENVIRONNEMENT DE PHP: Le fichier php.ini

C'est un fichier texte qui permet de configurer le moteur PHP sur le serveur Web. Certaines fonctionnalités de PHP sont figées et ne peuvent pas être modifiées, en revanche, d'autres sont personnalisables.

Il contient des directives que le moteur PHP prend en considération lors de la compilation du code. Il renferme beaucoup de directives qui permettent de manipuler certaines options.
L'emplacement du fichier **php.ini** peut changer d'une plateforme de développement à une autre. Pour WAMP, il suffit de cliquer sur la petite icone du logiciel sur la barre des tâches puis sélectionner l'option **Fichiers de configuration** ou **Configuration PHP** (l'option à sélectionner change légèrement d'une version à autre, mais je suis sûr que vous vous en sortirez), puis vous sélectionnez **php.ini**.

Attention:
Une mauvaise manipulation du fichier **php.ini** peut paralyser le serveur Web. Il vaut mieux faire une copie de ce fichier avant de l'éditer. Si un problème survient il suffit de rétablir la version sauvegardée.

COMMENT INTÉGRER LE CODE

Le code php s'ouvre par **<?php** et se referme par **?>**

Exemple:

```
<?php
        echo « Hello world! »;
?>
```

Le fichier php doit être enregistré sous le suffixe « .php »

Exercice:

Ecrivez le code ci-dessus dans votre éditeur et affichez Hello world sur le navigateur.

NOTIONS DE BASE: Les commentaires

En PHP on peut intégrer des commentaires comme vous avez pu le voir pour le html, le css et le Javascript. Les commentaires en PHP s'écrivent de la même manière.

Commentaire sur une ligne avec //

Commentaire sur plusieurs lignes: avec /* et */

Exemple:

```
<?php
  // Commentaire de fin de ligne
  /*
    Bloc entier
    vu comme un
    commentaire
  */
?>
```

NOTIONS DE BASE:
Les variables

En PHP, comme pour les autres langages, les variables servent à stocker des valeurs qui peuvent changer au cours du programme.

En PHP, les noms de variables sont préfixées par le symbole $ et commencent par une lettre minuscule, majuscule ou le underscore _. Cependant, ils peuvent contenir des chiffres au milieu ou à la fin jamais au début. Les espaces ne sont pas autorisés .

Exemple:

```
<?php
        $a=10;   // Juste

        $_a1=true;   // Juste

        $2a=« Hello: »;   // Faux (le chiffre ne doit pas figurer au début)

        $a b=5;      // Faux (le nom de la variable ne doit pas contenir d'espaces)
?>
```

NOTIONS DE BASE:
Les variables

En PHP, il existe des variables scalaires et des variables tableau (que nous verrons plus tard).

Le variables scalaires peuvent être:

- Des nombres entiers (nombres sans virgule positifs ou négatifs).
- Des nombres décimaux (de type double).
- Des chaines de caractères (des suites de caractères quelconques).
- Des booléens (qui peuvent avoir les valeurs true ou false).

Si une variable appelée dans le script existe déjà et a déjà reçu une valeur, celle ci sera utilisée.
Sinon
alors le moteur PHP lui affecte la valeur 0 par défaut.

Exemple:

```
<?php
        $a=10;
        $b=$a+1;   // $b vaut ?
        $d=$c+1;   // $d vaut ?
?>
```

NOTIONS DE BASE:
Les variables

Assignation par références

Elle permet à deux variables de pointer sur le même contenu. Cette assignation est assurée par une simple

affectation avec le paramètre &.

Exemple: $a=&$b;

Dans ce cas, les variables $a et $b pointent sur le même contenu, si, après cette assignation, on change la valeur de

$a alors $b changera aussi et aura la même valeur que $a et inversement.

NOTIONS DE BASE:
Les constantes

Comme les variables, les constantes servent aussi à stocker des valeurs dans un programme, mais à l'inverse des variables, leurs valeurs ne changent pas.

Pour définir un constante on utilise la fonction **define(name, value)**.
Name étant l'identifiant de la constante à définir et **value** sa valeur.

La fonction echo permet d'afficher une chaine de caractère dans le navigateur.

Exemple:

```
<?php
        define(« constante », « Hello !»);
        echo constante;  // Affiche  ??
?>
```

Question:

Que se passe-t-il si nous écrivons ceci à la suite du code ci-dessus?

```
define(« constante », « Bonjour !»);
echo constante;
```

NOTIONS DE BASE:
Les opérateurs

Les opérateurs sont des symboles qui permettent de faire des opérations sur les variables. Les opérateurs sont souvent les mêmes dans la plupart des langages de programmation et ils sont représentés par des symboles similaires dans la plupart des cas.

En PHP on distingue 5 familles d'opérateurs

- Opérateurs arithmétiques.
- Opérateurs d'incrémentation.
- Opérateurs assignement.
- Opérateurs de comparaison.
- Opérateurs logiques.

NOTIONS DE BASE: Les opérateurs

Opérateurs arithmétiques: ils permettent d'effectuer des calculs mathématiques classiques à savoir: l'addition (+), la soustraction (-), la multiplication (*), la division (/) et le modulo (qui signifie le reste de la division)(%).

Exemple:

```
$a=10;
$b=20;
$c=$a+$b; // $c vaut 30
$d=$a-$b; // $d vaut -10
```

Opérateurs d'incrémentation

Ce sont des opérateurs qui permettent de modifier la valeurs d'une variable en l'augmentant ou la diminuant de 1. Deux opérateurs sont Utilisés: incrémentation (++) et décrémentation (--).

Exemple:

```
$a=10;
$a++; // $a vaut donc 11
$a--; // $a vaut à nouveau 10
```

Parfois on préfère regrouper les opérations d'incrémentation dans la troisième famille qui constitue les opérateurs d'assignement.

NOTIONS DE BASE: Les opérateurs

Opérateurs d'assignement: on peut affecter une valeur à une variable directement ou à travers une opération (comme l'incrémentation
ou la décrémentation). Les opérateurs disponibles pour cette famille sont: l'affectation directe (=), l'affectation avec addition (+=),
l'affectation avec soustraction (-=), l'affectation avec multiplication (*=), l'affectation avec division (/=) et l'affectation avec modulo (%=).

Exemple:

```
$a=10;
$a+=5; // $a vaut 15
```

On peut écrire l'instruction $a++ comme ceci: $a+=1 ou encore: $a=$a+1.

Opérateurs de comparaison: ils permettent de tester si une condition est vraie ou fausse. Pour cela on utilise: égal (==), strictement inférieur (<), inférieur ou égal (<=), strictement supérieur (>), supérieur ou égal (>=), différent (!=), identique (===), non identique (!==).

Exemple:

```
$a=10;
if($a<0)
  $nb="Nombre négatif";
```

Opérateurs logiques

Ce sont des opérateurs qui regroupent logiquement plusieurs conditions. Les trois opérateurs utilisés sont: ET logique (&&), OU logique (||) et NON logique (!).

LES STRUCTURES DE CONTRÔLE: if/else

- La structure if else permet de vérifier si une condition est vraie pour exécuter un traitement.
  Exemple:

```
<?php
    if(condition){
       instructions;
    }
    else{
        instructions;
       }
 ?>
```

Le mot clé elseif remplace la séquence else suivie de if. L'utilisation des accolades n'est donc pas nécessaire et le code devient plus court.

Exercice:

Ecrire un code php permettant de vérifier la valeur du variable si $a=1 alors on affichera « un » et cela jusqu'à 5.

LES STRUCTURES DE CONTRÔLE: Opérateurs ternaires

L'opérateur ternaire permet de raccourcir la syntaxe de la structure de contrôle **if else** comme ceci:

(condition) ? (instructionVraie) : (instructionFaux)

Exemple:

```
<?php
   $a=1;
   echo ($a<0)?('$a est négatif'):('$a est positif');
?>
```

LES STRUCTURES DE CONTRÔLE: Switch case

Si la structure **if else** permet, de vérifier deux cas pour une condition, la structure **swith case**, permet de vérifier autant de cas que l'on souhaite.

```
<?php
   $jours=lundi;
   switch($jour){
      case lundi : echo « monday"; break;
      case mardi : echo « tuesday »; break;
      default: echo "Autre";
   }
?>
```

Les mot clé:

- **Break** met fin à la structure une fois la condition satisfaite.

- **Default** traitement par défaut.

LES STRUCTURES DE CONTRÔLE: Boucle « for »

La boucle **for** permet de faire des instructions de manière répétitif tant que la condition est vraie.

```
for(initialisation; sortie; incrémentation){
    instructions;
}
```

Exemple:

```
<?php
  for($i=0; $i<4; $i++){
     echo « Hello »;
  }
?>
```

Vérifions le contenu de la page via l'inspecteur du navigateur.

LES STRUCTURES DE CONTRÔLE: Boucle « while »

La boucle **while** permet de faire des instructions répétitive comme la boucle for.

Le code précédent s'écrirait comme suit:

```
<?php
  $i=0;
  while($i<4){
    echo « Hello »;
    $i++;
  }
?>
```

LES STRUCTURES DE CONTRÔLE: Boucle « do while »

La boucle **do while** ressemble beaucoup à la boucle **while** sauf que la vérification de la condition est placé après le traitement et non pas avant. Ce qui signifie que la première itération est toujours exécutée.

Avec la structure **do While** le code précédent devient:

```
<?php
    $i=0;
    do{
      echo « Hello »;
      $i++;
    }
    while($i<4);
  ?>
```

LES STRUCTURES DE CONTRÔLE: Break

Le mot clé **break** permet de mettre fin à une boucle prématurément.

Exemple:

```
<?php
   for($i=0;$i<4;$i++){
      if($i==3)
         break;
         echo « Hello »;
   }
?>
```

LES STRUCTURES DE CONTRÔLE: Continue

Le mot-clé **continue** permet d'ignorer l'itération courante est passer immédiatement à l'itération suivante.

Exemple:

```
<?php
   for($i=0;$i<4;$i++){
      if($i==3)
         continue;
         echo "$i <br/>";
   }
?>
```

LES CHAINES DE CARACTÈRES: Concaténation

- Pour concaténer les chaînes de caractères on utilise un point. Cet opérateur permet d'unir deux chaînes de caractères en une seule.

  Exemple:

```
<?php
   $a= « Hello »;
   $b= « World »;
   $c=$a . «  » . $b;
   echo $c;
?>
```

LES CHAINES DE CARACTÈRES: Les fonctions

- strlen($str): retourne un entier qui représente le nombre de caractères que contient la chaîne $str.
- strtoupper($str): convertir la chaîne $str en majuscule.
- strtolower($str): convertir la chaîne $str en minuscule.
- ucfirst($str): convertit le premier caractère de la chaîne $str en majuscules.
- trim($str): supprime les espaces de début de de fin de la chaîne $str.
- rtrim($str) ou chop($str): supprime les espaces de fin de la chaîne $str.
- substr($str,$deb,$nbr): extrait une partie de la chaîne de caractères en commençant de la position $deb et en retournant $nbr caractères (Notez que la position du premier caractère de la chaîne est 0).
- ord($car): retourne le code ASCII du caractère $car.
- chr($int): retourne le caractère correspondant au code ASCII $int.
- addslashes($str): ajoute des antislashs avant les caractères spéciaux comme l'antislash, simple cote ou double cote qui se trouvent dans la chaîne de caractères $str.
- stripslashes($str): supprime les antislashs qui se trouvent dans la chaîne de caractères $str.
- strip_tags($str): supprime les balises HTML qui se trouvent dans la chaîne de caractères $str.
- htmlentities($str): convertit certains caractères de $str en mot clés HTML.
- htmlspecialchars() : est pratique pour éviter que des données fournies par les utilisateurs contiennent des balises HTML, comme pour un forum ou un chat.
- str_replace($occ,$rem,$str): retourne une chaîne de caractères où toutes les occurrences définies par $occ sont remplacées une à une par l'entrée respectives définie en $remp au sein de la chaîne de caractères $str.

LES CHAINES DE CARACTÈRES: L'affichage

Comme nous l'avons vu plus haut, echo permet d'afficher une chaîne de caractères.
Il existe aussi la structure print qui permet d'afficher un message de la même manière que la structure echo .

Syntaxe avec echo:

```
<?php
  echo "Bonjour";
?>
```

Syntaxe avec print:

```
<?php
  print "Bonjour"; // Affiche: Bonjour
?>
```

Autre façon: <?="Bonjour"?>

LES FORMULAIRES

Prenons un simple formulaire html:

```
<form method="POST" action="">
        <input type="text" name="prenom" /><br />
        <input type="submit" name="valider" value="Vér
    ifier" />
</form>
```

LES FORMULAIRES: Récupération du contenu

La variables $_POST est en réalité un tableau associatif, c'est à dire un tableau qui utilise des clés au lieux d'indexes.

La variables $_POST contient la valeur du champ de formulaire dont le nom est passé en tant que clé.

Par exemple, pour récupérer la valeur que le client a saisi dans la zone de texte nommée prenom, on fait appel à la variable $_POST["prenom"].

La variable $_POST est appelée si le formulaire en question utilise la méthode POST.

Si le formulaire utilise la méthode GET alors on appelle la variable $_GET. Tout comme le tableau $_POST, le tableau $_GET utilise comme clé, les noms des champs de formulaires.

$_POST et $_GET sont des variables superglobales, c'est à dire qu'elles sont reconnues dans n'importe quel contexte (à l'intérieur des fonctions comme à l'extérieur, voir même à l'intérieur des méthodes d'une classe).

LES FORMULAIRES: Fonctions spécifiques

empty(): permet de vérifier si la variable passée en paramètre est vide, retourne un booléen (true, false).

isset(): permet de vérifier si la variable passée en paramètre existe, retourne un booléen (true, false).

unset(): permet de supprimer la variable passée en paramètre.

gettype(): permet de retourner le type de la variable passée en paramtère.

is_numeric(): vérifie si la variable passée an paramètre ne contient qu'une suite de caractères numériques. retourne un booléen (true, false).

is_int(): vérifie si la variable passée an paramètre est de type entier et retourne un booléen (true, false).

LES FORMULAIRES: Exercice pratique

Créer un formulaire comprenant les champs suivants:

« Nom », « Prénom », « Mail », « Tél ».

Ecrire le code php permettant de vérifier si les champs sont bien renseignés, sinon envoyer un message d'erreur.

LES FORMULAIRES:
Masquer les messages d'erreur

Il existe deux solutions pour maquer les messages d'erreurs:

- Interdire l'affichage des erreurs en manipulant le fameux fichier **php.ini** (chose fortement déconseillée lors de la phase développement)
- Masquer les messages d'erreur en utilisant le symbole arobas (@) avant l'instruction qui la cause. Utilisons cette méthode.

LES FONCTIONS PRÉDÉFINIES

Voici un récapitulatif des fonctions prédéfinies en php:

http://www.conseil-webmaster.com/formation/php/recapitulatif-sur-fonctions-php.php

LE TP

Travail à faire en groupe de 3:

Créer un formulaire d'inscription avec les champs suivants:

civilité, nom, prénom, date de naissance, adresse, cp, mail .

Puis appliquer les vérifications qui s'imposent.

Le but ici n'est pas de stocker les données en base de données (bdd)