WEB DYNAMIQUE

Introduction au langage PHP

SAGIM 2024/2025

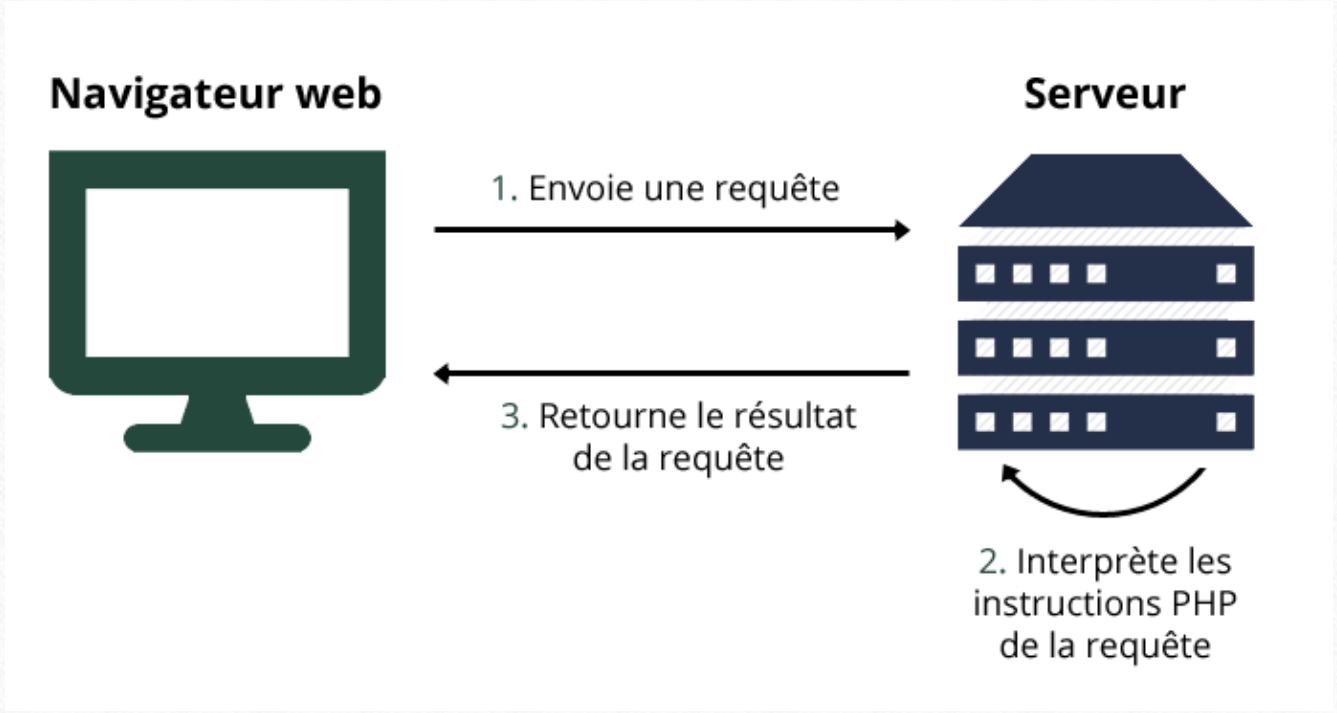
SAGIM

Introduction

- Le PHP est un langage de programmation qui fonctionne **côté serveur**.
- À la différence des langages HTML ou CSS qui s'exécutent côté client, c'est-à-dire dans le navigateur web, le PHP va s'éxécuter sur un serveur.

SAGIM

Introduction



SAGIM

INTRODUCTION

- Lorsqu'un utilisateur demande l'affichage d'une page d'un site internet en PHP au travers de son navigateur web, il envoie ce qu'on appelle une requête à un serveur.
- Quand le serveur reçoit la requête, suivant la catégorie de cette dernière, il la traite de manière différente.

Syntaxe de base

- L'extension de fichier pour les fichiers PHP est .php. Un extrait de code PHP peut être placé n'importe où dans un fichier avec une extension .php.
- Un extrait de code PHP commence toujours par **<?**php ou **<?** et se termine par **?>**.

```
<?php
  Ici le code en PHP
?>
```

SAGIM

Créer sa première page PHP

- Dans votre éditeur de texte favori, ouvrez un nouveau document et commencez par l'enregistrer sous le nom que vous souhaitez, mais avec l'extension .php et inserer ce code

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Ma première page PHP</title>
        </head>
        <body>
                <?php    echo 'Bonjour';  ?>
  </body>
</html>
```

SAGIM

Commentaires

- les commentaires en PHP ne sont pas interprétés. Leur seul but consiste à apporter une aide textuelle à la personne qui lira,

```
<?php
        // Un commentaire sur une ligne
        # Un commentaire sur une ligne
        /*
            Un commentaire qui
              occupe plusieurs lignes.
        */
?>
```

SAGIM

Les variables - PHP

- En PHP, une variable est un conteneur qui va nous permettre de stocker des informations de différents types (texte, entier, booléen, etc.). Elles ne servent qu'à stocker des informations temporairement.

SAGIM

Les variables - PHP

- on peut identifier nos variables avec n'importe quel nom. Cependant, il y a tout de même quelques règles à respecter. Ainsi, pour déclarer une variable en PHP, il faut préciser le signe $ (dollar) avant chaque nom de variable.

```
<?php
        $test = 1;
        $_test = 2;
    $test_1 = 3;
?>
```

SAGIM

Les variables - PHP

- les noms de variables sont sensibles à la casse. Donc les variables $test, $TEST et $Test sont bien trois variables distinctes.

```
<?php

        $test = 1;    $TEST = 2;    $Test = 3;

?>
```

SAGIM

Les variables - PHP

<?php $ville = 'Fontainebleau'; $habitants = 14974; ?>

- Dans la première variable, $ville, nous avons stocké la chaîne de caractères, le texte, Fontainebleau.Tandis que dans la seconde variable, $ville, nous avons stocké la valeur numérique, 14974.

Afficher le contenu d'une variable PHP

- Les variables sont très importantes en PHP. On peut les manipuler pour de nombreux besoins mais le plus basique consiste à afficher la valeur d'une variable. Pour ce faire, il faut utiliser l'instruction **echo**. Elle est très simple à utiliser.

```
<?php
  $ville = 'Fontainebleau';
  echo $ville;
?>
```

SAGIM

Modifier le contenu d'une variable PHP

- la valeur d'une variable peut évoluer au fil de l'exécution du script PHP. Il est tout à fait possible d'affecter une nouvelle valeur à une variable déjà créée.

```
<?php
   $habitants = 14974;    echo $habitants;
   $habitants = 149;echo $habitants;
?>
```

SAGIM

La concaténation en PHP

- Le terme concaténation en PHP désigne l'action de mettre bout à bout au moins deux chaînes de caractères.

```
<?php
    $intro = 'Hello '; $prenom = 'Bob';
    $titre = $intro.$prenom; echo $titre;
?>
```

La concaténation en PHP

- On peut ajouter un espace avant et après le . (point), il n'y a pas d'impact. Il n'y a pas de limitation dans la concaténation des chaînes de caractères. On peut imaginer quelque chose d'un peu plus complexe.

```
<?php
  $intro = 'Hello ‘; $prenom = 'Bob’; $fin = 'copain';
  $titre = $intro.$prenom.', tu veux être mon '.$fin.' ?';
  echo $titre;
?>
```

Les opérateurs arithmétiques en PHP

- Avec les variables, il est possible de réaliser des opérations arithmétiques. On peut donc notamment additionner des variables. Le tableau ci-après regroupe les principales opérations ainsi que les opérateurs que nous utiliserons.

```
<?php
  $a = 1;   $b = 3; $a = $a + 2;     $a = $a * $b;
  $a = $a - 1;      $a = $a / 4;   echo $a;
?>
```

SAGIM

Les opérateurs d'incrémentation et de décrémentation en PHP

- Incrémenter une valeur signifie additionner 1 à cette dernière, tandis que décrémenter consiste à l'inverse, à savoir soustraire de 1.

```
<?php
  $a = 9;      $b = 2;
  $a++;        $b--;
  echo 'La variable $a = '.$a.' et $b = '.$b.'.';
?>
```

Les types de variables

- Le typage des variables est implicite et dynamique en PHP.
- Il n’est donc pas nécessaire de déclarer leur type au préalable ni même de les initialiser avant leur utilisation, comme dans d'autres langages.

SAGIM

Les types de variables

Type de variable	Description	Exemple
String	Chaîne de caractères	`$a = 'Bonjour';` ou `$a = "Bonjour";`
Integer ou int	Nombre entier	`$a = 19;`
Float	Nombre décimal	`$a = 15.5;`
Boolean ou bool	Vrai/Faux	`$a = true;` ou `$a = false;`
Array	Tableau	`$a = array(8, 16, 32);`

Les types de variables

- Nous utiliserons la fonction var_dump(). Tout de suite un exemple pour comprendre comment l'utiliser et voir ce qu'elle retourne.

```
<?php
   $a = 15;       $b = 19.5;     $c = 'Bonjour';     $d = false;
   var_dump($a);          var_dump($b);
   var_dump($c);          var_dump($d);
?>
```

Sensibilité à la casse

- **les variables sont sensibles à la casse**. Une différence dans les noms de variables s'avère être une erreur courante en développement.

```
<?php
  $color = 'red';
  echo 'Ma voiture est '.$color;
  echo 'Ma voiture est '.$COLOR;
  echo 'Ma voiture est '.$coLOR;
?>
```

SAGIM

Les conditions - PHP

- Les **structures conditionnelles** ou tout simplement les **conditions** permettent d'exécuter une partie de code selon si une condition est vérifiée ou non. Par exemple, sur un site e-commerce, on peut finaliser une commande si le panier contient au moins un article. Les conditions s'utilisent principalement avec des variables.

SAGIM

L'instruction if

- En PHP, la condition la plus minimaliste possible requiert l'instruction if. En français, on peut tout simplement traduire cette instruction par si.
- Ci-après la syntaxe d'une condition if.

```
<?php
  $a = 2;
  if($a > 1)
    echo 'La variable $a est plus grand que 1.';
?>
```

SAGIM

L'instruction if

- Il est possible d'imbriquer les conditions. La syntaxe des instructions if n'est pas impactée. Ci-dessous un exemple.

```
<?php    $a = 2;
  if($a > 1) { echo 'La variable $a est plus grand que 1.';
    $b = 0;
    if($b < 2) {  echo '<br />La variable $b est plus petite que 2.'; }
  }  ?>
```

SAGIM

L'instruction else

- Elle est toujours associée à un if. Donc les instructions contenues dans le else sont exécutées uniquement si le test de condition du if n'est pas vérifié. On peut traduire en français cette instruction par sinon.

```
<?php  $a = 2;
  if($a > 1) {
     echo 'La variable $a est plus grand que 1.';
  } else {
     echo 'La variable $a est plus petit ou égale à 1.';
  } ?>
```

SAGIM

L'instruction elseif

- l'instruction elseif est une combinaison de l'instruction if et else. Le bloc d'instruction du elseif est exécuté uniquement si le test de condition du premier if n'est pas vérifié et vaut donc false.

```
<?php   $a = 5;
  if($a < 1) {  echo 'La variable $a est plus petit que 1.';
  } elseif($a > 4) {echo 'La variable $a est plus grand que 4.';
  } else { echo 'La variable $a est entre 1 et 4.'; }
?>
```

SAGIM

Les opérateurs de comparaison

Opérateur	Description	Exemple
==	`true` si la valeur est égale à	`$a == $b`
!=	`true` si la valeur est différente de	`$a != $b`
>	`true` si la valeur est strictement supérieure à	`$a > $b`
<	`true` si la valeur est strictement inférieure à	`$a < $b`
>=	`true` si la valeur est supérieure ou égale à	`$a >= $b`
<=	`true` si la valeur est inférieure ou égale à	`$a <= $b`

Les opérateurs de comparaison

- Nous utiliserons dans l'exemple ci-dessous la fonction var_dump() pour voir comment se comportent les tests de conditions.

```
<?php
  $a = 5;   $b = 10;
  var_dump($a == $b);     var_dump($a != $b);
  var_dump($a > $b);      var_dump($a < $b);
  var_dump($a >= $b);     var_dump($a <= $b);
?>
```

SAGIM

Les opérateurs logiques

- En PHP, les opérateurs logiques sont utilisés pour combiner des tests de conditions. Grâce à ces opérateurs, il est possible d'ajouter plus d'un test de condition dans une seule instruction if. Ci-après un exemple pour les utiliser.

Opérateur	Description	Exemple
&&	`true` seulement si tous les tests de condition sont évalués à true	`$a && $b`
||	`true` seulement si l'un des tests de condition est évalué à true	`$a || $b`
!	Inverse la logique d'un test de condition	`!$a`

SAGIM

Les opérateurs logiques

```
<?php
  $a = 10;      $b = 5;    $c = 0;
  if($a >= 10 || $b >= 10 || $c >= 10)
    echo 'Une des variables $a, $b ou $c a une valeur supérieure ou égale à 10.<br />';
  else
    echo 'Aucune des variables $a, $b ou $c ne contient une valeur supérieure ou égale à 10.<br />';

 ?>
```

EXERCICES:

- **Parmi les variables suivantes, lesquelles ont un nom valide : $a, $_a, $a_a, $AAA, $a!, $1a et $a1 ?**

Solution:

Seules les variables $a, $_a, $a_a, $AAA et $a1 ont un nom valide en PHP. $1a n'est pas un nom de variable valide car il commence par un chiffre, tandis que $a! est également incorrect car il contient un caractère interdit.

SAGIM

EXERCICES:

- Modifier le code ci-dessous pour calculer la moyenne des notes.

```
<?php
  $note_maths = 15;
  $note_francais = 12;
  $note_histoire_geo = 9;
  $moyenne = 0;
  echo 'La moyenne est de '.$moyenne.' / 20.';
?>
```

SAGIM

SOLUTION

```
<?php
   $note_maths = 15;
   $note_francais = 12;
   $note_histoire_geo = 9;
   $moyenne = ($note_maths + $note_francais + $note_histoire_geo) / 3;
   echo 'La moyenne est de '.$moyenne.' / 20.';
?>
```

EXERCICES:

- Calculer le prix TTC du produit.

```
<?php
  $prix_ht = 50;
  $tva = 20;
  $prix_ttc = 0;
  echo 'Le prix TTC du produit est de '.$prix_ttc.' DH.';
?>
```

SOLUTION:

```
<?php
   $prix_ht = 50;
   $tva = 20;
   $prix_ttc = $prix_ht * (1 + ($tva / 100));
   echo 'Le prix TTC du produit est de '.$prix_ttc.' DH.';
?>
```

SAGIM

Les boucles - PHP

- En PHP, il existe 4 boucles différentes. Elles remplissent la même finalité mais comportent des différences.
  - **while**
  - **do...while**
  - **for**
  - **foreach**

Boucle while

L'instruction **while** pourrait être traduite par « tant que ».

on peut dire que cette dernière va exécuter la même portion de code tant que la condition est vérifiée.

```
<?php
  $z = 0;
  while($z < 10) {
    echo 'La variable $z a '.$z.' comme valeur.<br />';   $z++;   }
?>
```

Boucle do…while

- L'instruction do...while pourrait être traduite en français par « faire... tant que ». La boucle do...while est très proche de la boucle while.
- La différence est que la boucle do...while commence par exécuter sa portion de code avant de vérifier sa condition alors que la boucle while fait l'inverse.

SAGIM

Boucle do…while

```
<?php
   $z = 2;
   do {
      echo 'La variable $z a '.$z.' comme valeur.';
   } while($z < 1);
?>
```

SAGIM

Boucle for

l'instruction for pourrait être traduite en français par « pour ».

Les 3 éléments de la boucle for sont les suivants :

Une déclaration de variable ;

Un test de condition ;

Une incrémentaiton de variable.

SAGIM

Boucle for

```
<?php
   for  (  $z = 0  ;  $z < 10  ;  $z++  ) {
      echo 'La variable $z a '.$z.' comme valeur.<br />';
   }
?>
```

SAGIM

Les tableaux

Un tableau est un type de variable spécial. En PHP, comme dans d'autres langages, les tableaux sont appelés des array.

Le type de variable pour un tableau est donc array.

SAGIM

Les tableaux

Dans un tableau, chaque valeur est reliée par défaut à une clé ou à une key en anglais. En somme, un tableau est une suite d'associations de clés et de valeurs (key, value en anglais).

Par défaut, les valeurs sont reliées à une clé numérique. On parle alors de tableau numéroté ou indexé, et en anglais de indexed array.

SAGIM

Les tableaux

```
<?php
   $villes = array(‘casa', ‘tanger', ‘marrakech');
   echo $villes[0].' - '.$villes[1].' - '.$villes[2];
?>
```

Parcourir les tableaux : foreach

La boucle foreach permet de parcourir simplement les tableaux. Elle ne fonctionne d'ailleurs qu'avec ce type de variable.

```
<?php
  $villes = array(‘casa’, ‘tanger', ‘rabat');
  foreach($villes as $key => $ville) {
    echo $ville.' a la clé '.$key.'<br />';
  }
?>
```

SAGIM

EXERCICES

Déclarer une variable $age qui contient la valeur de type integer de votre choix.

Réaliser une condition pour afficher si la personne est mineure ou majeure.

SAGIM

SOLUTION :

```
<?php
   $age = 19;
   if($age >= 18)
      echo 'Vous êtes majeur.';
   else
      echo 'Vous êtes mineur.';
?>
```

EXERCICES

Déclarer une variable $heure qui contient la valeur de type integer de votre choix comprise entre 0 et 24.

Créer une condition qui affiche un message si l'heure est le matin, l'après-midi ou la nuit.

SAGIM

SOLUTION :

```
<?php  $heure = 14;
        if($heure < 0 || $heure > 23) : echo 'Houla, cette heure est incorrecte.';
        elseif($heure >= 7 && $heure < 12) :  echo 'Bonne matinée.';
        elseif($heure >= 12 && $heure < 22) :  echo 'Bonne après-midi.';
        else : echo 'Bonne nuit.';
        endif;
?>
```

SAGIM

EXERCICES

En utilisant la boucle for, afficher la table de multiplication du chiffre 5.

SAGIM

SOLUTION :

```
<?php
   $n = 5;
   for($i = 1;$i <= 10;$i++) {
      echo $n.' x '.$i.' = '.($n * $i).' <br />';
   }
?>
```

SAGIM

EXERCICES:

- Déclarer une variable avec le nom de votre choix et avec la valeur **0**. Tant que cette variable n'atteint pas **20**, il faut :
- l'afficher ;
- incrémenter sa valeur de **2** ;
- Si la valeur de la variable est égale à **10**, la mettre en valeur avec la balise HTML appropriée.

SAGIM

SOLUTION :

```
<?php
  $k = 0;
  while($k <= 20) {
    if($k == 10) {
      echo '<strong>'.$k.'</strong>';
    } else {
      echo $k;
    }
    echo '<br />';
    $k = $k + 2;
  }
?>
```

SAGIM

LES FONCTIONS

SAGIM

Les Fonctions En PHP :

En PHP, une fonction est un bloc de code qui peut être exécuté à plusieurs endroits dans un script.

Les fonctions permettent de structurer le code, de le réutiliser et de le maintenir plus facilement.

Elles peuvent prendre des paramètres en entrée et retourner une valeur en sortie.

SAGIM

Déclaration d'une Fonction

Pour déclarer une fonction en PHP, on utilise le mot-clé **function** suivi du nom de la fonction et d'une paire de parenthèses.

Le corps de la fonction est ensuite défini entre accolades {}.

SAGIM

Déclaration d'une Fonction

```
function direBonjour() {
                    echo "Bonjour  SAGIM !";
                    }
// pour appeler une fonction :
direBonjour();
```

SAGIM

Paramètres de Fonction

Les fonctions peuvent accepter des paramètres (ou arguments) qui sont des valeurs passées à la fonction pour qu'elle les utilise.

SAGIM

Paramètres de Fonction

```
function direBonjourA($nom) {
    echo "Bonjour ".$nom. " ! " ;
}
```

Valeur de Retour

Une fonction peut retourner une valeur en utilisant le mot-clé **return**.
Une fois que **return** est exécuté, la fonction termine son exécution.

SAGIM

Valeur de Retour

```
function addition($a, $b) {
    return $a + $b;
}
$resultat = addition(3, 5);
Echo $resultat;
```

SAGIM

EXERCICES:

Créez une fonction qui prend un âge en paramètre et retourne "Majeur" si l'âge est supérieur ou égal à 18, sinon retourne "Mineur".

SAGIM

Solution:

```php
<?php
function estMajeur($age) {
    return ($age >= 18) ? "Majeur" : "Mineur";
}
echo estMajeur(20); // Affiche "Majeur"
?>
```

SAGIM

EXERCICES:

Créez une fonction estPair($nombre) qui retourne true si le nombre passé en paramètre est pair, sinon false.Testez la fonction avec plusieurs valeurs et affichez un message indiquant si chaque nombre est pair ou impair.

SAGIM

Solution:

```
function estPair($nombre) {
return $nombre % 2 == 0;
}
$nombre = 10;
echo $nombre . (estPair($nombre) ? " est pair" : " est impair") . "\n";
```

SAGIM

EXERCICES:

Développez une fonction genererTableau($n) qui crée un tableau contenant n nombres aléatoires entre 1 et 100.La fonction doit retourner ce tableau.Affichez ensuite les valeurs du tableau généré.

SAGIM

Solution:

```
function genererTableau($n) {
        $tableau = [ ];
            for ($i = 0; $i < $n; $i++) {
                $tableau[ ] = rand(1, 100);
                                }
    return $tableau;
                            }
print_r(genererTableau(5));
```