Université Paris Nanterre

Université Paris Nanterre

Programmation web et Introduction à PHP

L2 MIASHS

M. NAFI

m.nafi@parisnanterre.fr

Année: 2022/2023

Objectifs du cours

- Création de **sites web statiques, interactifs** et **dynamiques** à l'aide de:
  - Langage **html**
  - Langage **CSS**
  - Langage **JavaScript**
  - Langage **PHP**

Plan du cours

1. Introduction
2. Langage Html
3. Langage CSS
4. Langage Javascript
5. Langage PHP

Plan du cours

1. Introduction
2. Langage HTML
3. Langage CSS
4. Langage Javascript
5. **Langage PHP**

Introduction

- PHP (PHP Hypertext Preprocessor)
- Il a été créé en 1994 par Rasmus Lerdorf
- Un langage de script utilisé pour le développement de pages web **dynamiques**
- Les **scripts** ou codes PHP sont exécutés sur le serveur. Le résultat est retourné au client (navigateur) sous forme de document ou code **HTML**

- Syntaxe du PHP est **proche** de celle du langage C, Perl ou Java

- PHP est **multiplateformes** (Windows, Linux, Mac OS, etc)

- **Compatible** avec la plupart des serveurs web (Apache, IIS, etc)

- Supporte un large éventail de **bases de données** (MySQL, MariaDB, PostgreSQL, etc)
- **Open source** et gratuit
- Un fichier PHP porte l'extension **.php** et contient du texte, html, css, javascript et le code php.

Exemple

```
<!DOCTYPE HTML>
<HTML>
    <head>
        <title> titre </title>
    </head>
    <body>
        <h1> <?php echo "Salut";?>  </h1>
    </body>
</HTML>
```

Serveur


```
<!DOCTYPE HTML>
<HTML>
    <head>
        <title> titre
        </title>
    </head>
    <body>
     <h1> Salut </h1>
    </body
</HTML>
```

Client

Installation

Besoins

- Installer un serveur web (Apache)
- Installer l'interpréteur PHP
- Installer un SGBD (MySQL)

Ou installer **XAMPP**(Apache MySQL PHP Perl)

- **W**AMP: Windows
- **L**AMP: Linux
- **M**AMP: Mac OS

Syntaxe

- Un fichier PHP porte l'extension **.php** et contient du texte, html, css, javascript et le code ou script php.
- Un script PHP est délimité par les deux balises: <script language='php'> et </script> ou **<?php** et **?>**
- Une instruction se termine par **';'** (séparateur)

Ex.

```
<?php
    echo  " Hello world  ";
?>
```

Variables

- Les variables sont des **conteneurs** pour le stockage des informations

- En PHP, une variable commence par le signe $, suivi du nom de cette variable

- Le nom de la variable doit commencer par une **lettre** ou le caractère de **soulignement** '_'.

- Le nom des variables est **sensible** à la casse ($x ≠ $X)
- Le nom des variables ne doit pas commencer par un nombre
- Le nom d'une variable ne peut contenir que des caractères alphanumériques et des caractères de soulignement (A-z, 0-9 et _)

- Nb: Donner des noms **significatifs** aux variables (**$age** au lieu de **$a)**

- Ex.<?php
  $x = “PHP!";
  $y = 1;
  ?>

- Après exécution de ce script PHP, la variable:
- **$x** contiendra la chaine de caractère “PHP!“
- **$y** la valeur entière “1”

Portée d'une variable

- **Locale**: déclarée et accessible seulement **à l'intérieur** d'une fonction
- **Globale**: déclarée et accessible **en dehors** d'une fonction

- Le mot-clé « global » est utilisé pour accéder à une variable globale à l'intérieur d'une fonction

- **Statique**: on utilise le mot clé « static » pour **garder** la valeur d'une variable locale (ie elle ne sera pas supprimée à la fin de l'exécution de la fonction)

Variables superglobales

- Toujours accessibles de n'importe où et quel que soit leurs portées
  - $_POST
  - $_GET
  - $_REQUEST
  - $GLOBALS
  - $_SERVER
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION

- $_POST: utilisée pour collecter les données de formulaire soumises avec la méthode ‘Post’
- $_GET: utilisée pour collecter les données de formulaire soumises avec la méthode ‘Get’
- $_REQUEST: utilisée pour collecter les données après la soumission d'un formulaire HTML
- $GLOBALS: utilisé pour accéder aux variables globales depuis n'importe quel endroit du script PHP
- $_SERVER: contient des informations sur les en-têtes, les chemins et les emplacements des scripts

Types de données

- PHP attribue **automatiquement** un type de données à la variable, en fonction de sa valeur

- Les variables peuvent stocker des données de **différents** types tels que: Integer, float, string, array, boolean, object, Null, resource

Integer

- Nombre entier **négatif** ou **positif**

  Ex. $x = 20;

- Quelques fonctions sur les entiers
  - is_int()
  - is_integer()
  - is_long()

Float

- Nombre décimal ou à virgule flottante
- Ex. $x = 20.5;
- Quelques **fonctions** sur les nombres réels
  - is_float()
  - is_double()

String

- Une **séquence** ou une **chaine** de caractères
- Texte entre **guillemets**

  Ex. $x= "Apprendre PHP"
- Voici quelques fonctions couramment utilisées pour manipuler les chaînes de caractères:
  - **strlen()** retourne la **taille** d'une chaine de caractères
  - **strrev()** retourne **l'inverse** d'une chaine de caractères
  - **strpos()** retourne **la position** d'une sous chaine
  - **str_replace()** **remplace** certains caractères par d'autres

Boolean

- **Deux** valeurs ou états: vrai ou faux

Ex. $x=true;

$y=false;

Array

- Variable contenant plusieurs valeurs

Ex.

$couleur = **array** **(“rouge",“orange",“vert");**

Object

- Concept de la programmation **orientée objet**
- Une **instance** d'une classe
- Un objet hérite toutes les **propriétés** et tous les **comportements** de la classe

Null

- Aucune valeur ou une seule valeur:null

Fonction vardump()

- Fonction var_dump() est utilisée pour connaitre le type et la valeur d'une variable.

- Ex.

```
<?php
   $x=10;
   echo var_dump($x);
?>
```

Résultat: int(10)

Opérateurs

- Affectation (=, +=, -=,*=,/=,%=)
- Arithmétiques (+,-,*,/,%)
- Concaténation (.)
- Logiques (&& (et), || (ou), ! (non))
- Comparaison (==,!=,<,<=, >,>=)

Structures de contrôle (1/2)

- Instructions conditionnelles
  - If () else
  - If () elseif () else
  - Switch

Structures de contrôle (2/2)

Boucles

- **While**: Tant que la condition est satisfaite, un bloc de code est exécuté.
- **Do …while**: Répéter un bloc de code tant que la condition est vérifiée (le bloc est exécuté au **moins** une **seule** fois)
- **For**: exécuter un bloc de code un nombre de fois **spécifique**
- **Foreach**: utilisé pour les array

While

- Syntaxe

```
while (condition est vraie) {
        code à executer;
}
```

Ex.

```
<?php
    $x = 1;
    while($x <= 10) {
    echo "$x <br>";
    $x++;
    }
?>
```

Do ... while

- Syntaxe

```
Do{
        code à executer;
} while (condition est vraie)
```

- Ex.

```
<?php
    $x = 1;
    Do{
        echo "$x <br>";
        $x++;
    } while($x <= 10)
?>
```

For

- Syntaxe

```
for (init;  test;  increment)
  {
       Code à executer;
   }
```

- Ex.

```
<?php
    $x = 1;
    for ($x=1;$x <= 10;$x++) {
    echo "$x <br>“;
   }
?>
```

Foreach

- Syntaxe

```
foreach ($array as $valeur){
        code à executer;
}
```

- Ex.

```
<?php
    $x = array("1","2","3");
    foreach ($x as $valeur) {
    echo "$valeur <br>;
   }
?>
```

Fonctions

- PHP possède plus de 1000 fonctions prédéfinies
- Une fonction est un bloc d'instructions
- Elle ne s'exécute pas automatiquement
- On peut créer nos propres fonctions

```
function NomFonction (parameteres) {
        // code à executer;
}
```

- *NomFonction* doit commencer par une **lettre** ou « _ »
- Une fonction doit être définie avant d'être appelée

Exemple

- Ecrire une fonction qui détermine le maximum entre deux nombres
- On peut la nommer Max

```
<?php
    Function Max (int $x, int $y) {
        If ($x>$y){
            echo "le maximum est: ".$x;
        else
            echo "le maximum est: ".$y;
                }
        }
    Max(12, 3); // appel de la fonction Elle affiche le maximum est:12
    ?>
```

```php
<?php
function maxi(int $x, int $y=3) {
  if ($x>$y){
  echo "le maximum est:".$x;
  }
   else {
   echo "le maximum est: ".$y;
  }
}
maxi(4,5); // affiche le maximum est:5
maxi(2); // affiche le maximum est:3 utilise la valeur par défaut
?>
```

- La fonction qui retourne la somme de deux entiers

```
<?php
  function somme(int $a, int $b) {
  return $a + $b;
 }
 echo somme(2, “3 ");
 ?>
```

- Sortie 5 // comme strict n’est pas activé, “3" est transformé en int(3), et la fonction retourne 5

- Langage PHP est faiblement typé

```
<?php
    declare(strict_types=1); // types doivent être respectés
    function somme(int $a, int $b) {
    return $a + $b;
    }
    echo somme(2,“3 ");
?>
```

Sortie: PHP Fatal error

Quelques fonctions mathématique

- **pi():** renvoie la valeur de **pi**
- **min():** renvoie la valeur **minimale** d'un tableau
- **max():** renvoie la valeur **maximale** d'un tableau
- **abs():** calcule la valeur **absolue** d'un nombre
- **sqrt():** calcule la **racine carrée** d'un nombre
- **round():** arrondit un nombre à la valeur **la plus proche**
- **Rand():** renvoie un nombre **aléatoire**
- **etc**

Tableaux (array)

- Un tableau stocke **plusieurs** valeurs dans une seule variable
- On peut accéder aux valeurs en se référant aux numéros d'indice ou clés
- En PHP, la notation **[ ]** ou la fonction **array()** sont utilisées pour créer un tableau
- Ex. $tab=[ ] // tableau vide

  $tab=array() // tableau vide

- Il existe **trois** types de tableaux:
  - **Indexés**: utilisent des indices *numériques*
  - **Associatifs**: utilisent des *clés* nommées (clé=>valeur)
  - **Multidimensionnels**: contenant un ou *plusieurs* tableaux

Tableau indexé

- Il existe deux façons de créer un tableau indexé:
  - L'index est attribué *automatiquement*

```
$couleurs=array('rouge', 'vert', 'bleu');
```

  - L'index est affecté *manuellement*

```
$couleurs[0]='rouge';
$couleurs[1]='vert';
$couleurs[2]='bleu';
```

  - On peut utiliser la boucle **for** pour parcourir tous les éléments d'un tableau indexé.
  - Ex.

```
For($x=0; $x<count($couleurs);$x++)
    echo $couleurs[$x];
```

Tableau associatif

- Utilise des *clés nommées*
- Il existe deux façons de créer ce type de tableau :
  - $age = array("Peter"=>“25", "Ben"=>"30", "Jean"=>"40");
  - $age['Peter'] = “25";
    $age['Ben'] = "30";
    $age['Jean'] = "40";
- On peut utiliser la boucle foreach pour parcourir les éléments d’un tableau associatif.

- Ex. <?php

```
$age = array("Peter"=>“25", "Ben"=>"30“);

foreach($age as $x => $value) {
        echo “Clé=" . $x . ",Valeur=" . $value;
        echo "<br>";
}
?>
```

Tableau multidimensionnel

- Dans un tableau à **deux** dimensions, on a besoin de **deux** indices pour sélectionner un élément.
- Ex. $personne=array(

  array('Peter',' 25'),

  array('Ben',' 30'));

Nom	Age
Peter	25
Ben	30

- Pour accéder à l'age de 'Peter', on utilise deux indices $personne[0][1]='25' ou deux boucles **for**

Quelques fonctions

- Count(): compte le nombre d'éléments d'un tableau (taille)
- array_reverse(): inverse les éléments d'un tableau
- array_unique(): supprime les doublons
- sort(): trie un tableau indexé dans l'ordre croissant des ses valeurs
- etc

- Ex. <?php
  $couleur = array("rouge", "bleu", "vert");
  echo count($couleur);
  ?>

- Résultat: 3

Tri des éléments d'un tableau

- Les éléments d'un tableau peuvent être triés par ordre alphabétique ou numérique, croissant ou décroissant.
- Les fonctions de tri sont:
  - **sort():** tableau indexé, ordre croissant de ses valeurs
  - **rsort()**: tableau indexé, ordre décroissant de ses valeurs
  - **asort()**: tableau associatif, ordre croissant de ses valeurs
  - **ksort()**:tableau associatif, ordre croissant de ses clés
  - **arsort()**: tableau associatif, ordre décroissant de ses valeurs
  - **krsort()**: tableau associatif, ordre décroissant de ses clés

Exemple 1

```
<?php
    $colors = array("red", "blue", "green");
    rsort($colors); // ordre alphabétique décroissant
    $taille = count($colors);
    for($x = 0; $x < $taille; $x++) {
    echo $colors[$x]; // affichage des éléments du tableau
    echo "<br>";
  }
?>
```

Résultat: red
green
blue

Exemple 2

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
  asort($age);
  ?>

- Résultat:

  Key=Peter, Value= 35
  Key=Ben, Value= 37
  Key=Joe, Value= 43

Exemple 3

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
  ksort($age);
  ?>
- Résultat ?

  Key=Ben, Value=37
  Key=Joe, Value=43
  Key=Peter, Value=35

Exemple 4

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
  arsort($age);
  ?>
- Résultat ?

  Key=Joe, Value= 43
  Key=Ben, Value= 37
  Key=Peter, Value= 35

Exemple 5

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
  krsort($age);
  ?>
- Résultat ?

  Key=Peter, Value=35
  Key=Joe, Value=43
  Key=Ben, Value=37

Formulaires

- GET et POST sont deux méthodes utilisées pour ***collecter*** les données de formulaire.
- **Ex**

```
<form method='POST' action='file.php'>
        Nom:<input type='text' name='name'>
        <input type='submit'>
</form>
```

- File.php

```
<?php
        echo 'le nom est: $_POST[name]';
?>
```

- $_POST[name] crée un tableau associatif contenant des **clés** et des **valeurs** (clé1=>valeur1, clé2=>valeur2 …)

  - Clé: nom d'un champ de formulaire
  - Valeur: valeur saisie par l'utilisateur

Get vs Post

- **GET**
  - Données **visibles** (noms et valeurs sont affichées dans l'URL)
  - Quantité de données à envoyer est **limitée** (2000 caractères)
  - Utilisée pour l'envoi des données **non sensibles**
- **POST**
  - Données sont **invisibles** pour les autres (noms et valeurs sont encapsulés dans le **corps** de la requête HTTP)
  - **Aucune** limite sur la taille des données à envoyer
  - Utilisée pour l'envoi des données **sensibles** comme les *mots de passes*.

NB: Il est donc préférable d'utiliser POST pour la soumission des données.