

Programmation Web en PHP

PHP et les classes/les objets

- PHP est un langage **non objet** avec une couche objet au dessus : on programme avec des classes, des objets, des méthodes, de la composition, de l'héritage, mais
- Un programme php :
 - exécutable en ligne de commande : `php monprog.php`
 - Accessible via une url : <http://mon.si.te/monprog.php>
- Est forcément un programme **non objet**

un programme typique

- crée des objets et appelle des méthodes mais ...
- n'est pas lui même défini comme une méthode d'un objet ou d'une classe

```
<?php

include_once
'vendor/autoload.php' ;

session_start();

$app = new Dispatcher() ;

$app->run() ;
```

Définition des classes

déclaration des attributs

```
class Personne {  
    protected $nom, $prenom, $age;  
    public function __construct ($n,$p,$a) {  
        $this->nom=$n;  
        $this->prenom=$a;  
        $this->age=$s;  
    }  
    //méthode qui affiche les données d'un objet  
    public function affiche () {  
        echo "nom: $this->nom<br>";  
        echo "prenom: $this->prenom<br>";  
        echo "age: $this->age<br>";  
        $esperance = 82 - $this->age ;  
        print "il vous reste : $esperance années de vie" ;  
    }  
}
```

constructeur

méthode

accès dans l'objet courant

Variable locale

remarques

- pas de types pour les attributs
- pas de types de retour pour les méthodes (sauf PHP7)
- pour les paramètres :
 - pas de type s'il s'agit d'un type simple (sauf PHP7)
 - typage possible (non obligatoire, mais recommandé) s'il s'agit d'objets ou de tableaux
- \$this : obligatoire
 - pourquoi ? des idées ?

utiliser des objets, appeler des méthodes

```
<?php
```

```
...
```

```
$p = new Personne('népllin', 'jean', 42) ;
```

```
$p->affiche() ;
```

```
$age = $p->age ;
```

Définir des méthodes/fonctions

```
function func_name ([$arg1,$arg2, ...]) {  
    <statements>  
    [return ...;]  
}
```

```
<?php  
class Operation {  
    function add($a, $b) {  
        $s = $a+$b;  
        return $s;  
    }  
}
```

paramètres

resultat

appel de la méthode

```
$v1 = 10 ; $v2= 10;  
$o = new Operation() ;  
echo "$v1+$v2= " . $o->add($v1, $v2) ;
```

Portée des variables

- Paramètres et variables définies dans une fonction/méthodes sont des objets LOCAUX
- les variables définies en dehors d'une fonction ou d'une classe sont des objets GLOBAUX

```
<?php
$a = 1; $b = 2; $c = 3;
function sum() {
    global $a, $b;
    $b = $a + $b; $d = $b ;
    echo $c ;
    $c = 5;
}

sum() ;
echo "$b $c $d";
```

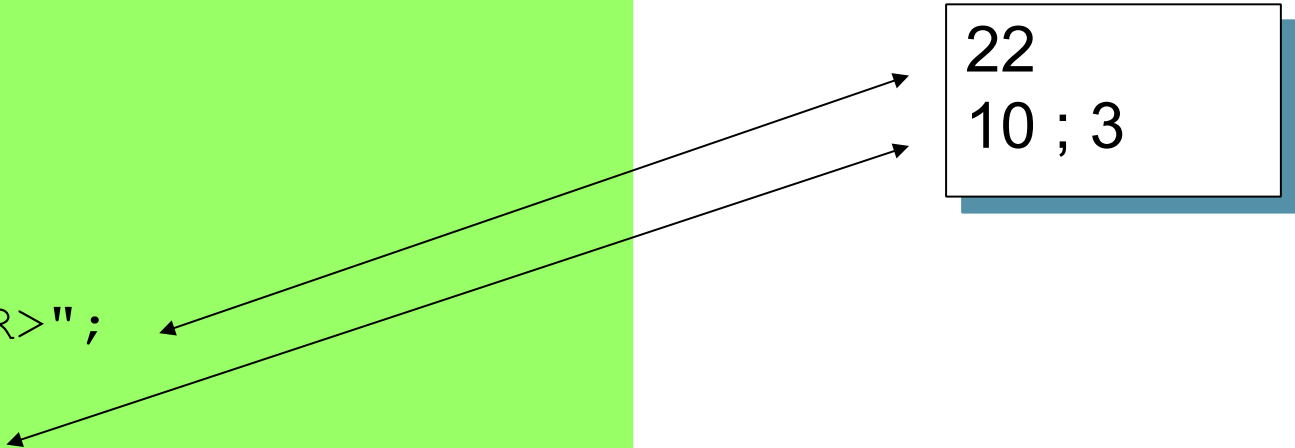
global permet d'accéder à des variables globales à l'intérieur d'une fonction

Warning: Undefined variable: c
Warning: Undefined variable: d
3 3

■ Passage des paramètres

- Les paramètres de types objet sont passés par référence
- Les paramètres de types simples et array sont passés par valeur : la fonction reçoit une copie de la valeur
- On peut passer une référence en faisant précéder le paramètre de &

```
<?php
function Add( $a, & $b) {
    $s = $a+$b;
    $a = 2; $b = 3;
    return $s;
}
$va=10; $vb=12;
echo add($va,$vb) . "<BR>";
echo "$va ; $vb";
```



22
10 ; 3

■ Paramètres optionnels, valeurs par défaut

- Il est possible de définir une **valeur par défaut** pour certains paramètres
- ils deviennent alors **optionnels** : en cas d'absence à l'appel, le paramètre reçoit la valeur par défaut

```
<?php
function foo($x, $y = 3, $z = null) {
    echo $x."<BR>";
    echo $y."<BR>";
    if (is_null($z)) { echo "null" ; }
    else echo "not null";
}

foo(1) ;
foo(4,5) ;
foo(4,5,6) ;
```

1
3
null

4
5
null

4
5
6

■ Déclarations de type des arguments (optionnel)

- Si la valeur donnée est d'un type incorrect alors une erreur est générée : en **PHP 5**, cela sera une **erreur fatale** tandis que **PHP 7** lèvera une **exception TypeError**.
- Sans le mode strict, PHP tente de transtyper dans le type attendu
- Types possibles :
 - en PHP 5 : nom de classe/interface, array, callable
 - En PHP 7 : bool, float, int, string

```
<?php  
declare(strict_types=1);  
function test(bool $param) {}  
test(true);  
test("true");
```

Et non
boolean !

OK

Fatal error: Uncaught TypeError: Argument 1
passed to test() must be an instance of boolean,
string given, [...]

■ Déclaration des types de retour (en PHP 7, optionnel)

- Mêmes types que pour les arguments
- 2 modes disponibles :
 - **Faible (par défaut)** : la valeur de retour est **transtypée**

```
<?php
function sum($a, $b): float {
    return $a + $b;
}
var_dump(sum(1, 2));
```

Retourne
un float

- **Strict** : exception **TypeError** si mauvais type

```
<?php
declare(strict_types=1);
function sum($a, $b): int {
    return $a + $b;
}
var_dump(sum(1, 2.5));
```

Fatal error: Uncaught
TypeError: Return value of
sum() must be of the type
integer, float returned

■ Opérateurs Arithmétiques

Example	Name	Result
$\$a + \b	Addition	Sum of \$a and \$b.
$\$a - \b	Subtraction	Difference of \$a and \$b.
$\$a * \b	Mult.	Product of \$a and \$b.
$\$a / \b	Division	Quotient of \$a and \$b.
$\$a \% \b	Modulus	Remainder of \$a divided by \$b.

Table 10-5. Opérateurs Incrément/Décrément

Example	Name	Effect
$++\$a$	Pre-increment	Increments \$a by one, then returns \$a.
$\$a++$	Post-increment	Returns \$a, then increments \$a by one.
$--\$a$	Pre-decrement	Decrements \$a by one, then returns \$a.
$\$a--$	Post-decrement	Returns \$a, then decrements \$a by one.

■ Opérateurs de comparaison

Example	Name	Result
\$a == \$b	Equal	TRUE if \$a is equal to \$b.
\$a === \$b	Identical	TRUE if \$a is equal to \$b, and they are of the same type.
\$a != \$b	Not equal	TRUE if \$a is not equal to \$b.
\$a <> \$b	Not equal	TRUE if \$a is not equal to \$b.
\$a !== \$b	Not identical	TRUE if \$a is not equal to \$b, or they are not of the same type. (PHP 4 only)
\$a < \$b	Less than	TRUE if \$a is strictly less than \$b.
\$a > \$b	Greater than	TRUE if \$a is strictly greater than \$b.
\$a <= \$b	Less than or equal to	TRUE if \$a is less than or equal to \$b.
\$a >= \$b	Greater than or equal to	TRUE if \$a is greater than or equal to \$b.

■ Opérateurs Logiques

Example	Name	Result
\$a and \$b	And	TRUE if both \$a and \$b are TRUE .
\$a or \$b	Or	TRUE if either \$a or \$b is TRUE .
\$a xor \$b	Xor	TRUE if either \$a or \$b is TRUE , but not both.
! \$a	Not	TRUE if \$a is not TRUE .
\$a && \$b	And	TRUE if both \$a and \$b are TRUE .
\$a \$b	Or	TRUE if either \$a or \$b is TRUE .

■ Opérateurs sur les chaînes :

Example	Name	Result
\$s1 . \$s2	Concat	The string built by concatenating s1 and s2

Conditionnelles

■ **if :**

```
if ( $a < 10) {  
    print "$a inferieur a 10" ;  
} else {  
    print "$a supérieur ou égal à 10" ;  
}
```

```
if ( $a < 10) {  
    print "$a inferieur a 10" ;  
} elseif ($a < 20 ) {  
    print "$a inferieur a 20" ;  
} else {  
    print "$a supérieur ou égal à 20" ;  
}
```


■ SWITCH

```
switch ( $a ) {
```

```
  case v1 :
```

```
    <statements1>
```

```
    break;
```

exécuté si <expr> == v1

```
  case v2 :
```

```
    <statements2>
```

```
    break;
```

exécuté si <expr> == v2

```
  ...
```

```
  case vn :
```

```
    <statementsn>
```

```
    break;
```

exécuté si <expr> == vn

```
  default :
```

```
    <statementsd>
```

exécuté dans les autres cas

```
}
```

■ La boucle For :

```
for ($i = 1; $i <= 10; $i++) {  
    echo $i;  
}
```

■ La boucle while

```
$i = 1;  
while ($i <= 10) {  
    echo $i++;  
}
```

Inclusion de code

■ **require()** :

- inclut et évalue le code contenu dans le fichier spécifié

■ **require_once()**

Identique à **require()**, sauf que le fichier n'est inclu qu'une seule fois

```
<?php  
  
require 'prepend.php';  
  
require $somefile;  
  
require ('somefile.txt');  
  
require_once 'autoload.php';
```

conventions de code : PSR-1

php standard recommandation - 1

- Voir <http://www.php-fig.org/psr/psr-1/>
- utiliser <?php
- utf8 sans BOM
- noms de classes : **StudlyCaps**
- méthodes : **camelCase()**
- attributs, variables : **\$camelCase**, **\$StudlyCaps**, **\$under_score** à condition d'être homogène dans le projet

- un fichier déclare une classe ou des fonctions, **OU** produit des résultats, jamais les 2

```
<?php  
require_once 'Personne.php' ;  
$polo=new Personne('polo', 'marco', 42);  
$polo->affiche() ;
```

```
class Personne {  
  
    public $nom, $prenom, $age ;  
  
    public function __construct( ..) { ... }  
  
    public function affiche() { ... }  
  
}
```

~~`$p = new Personne('neplin', 'jean', 1337);`~~

Personne.ph
p

exercices

- TD n°2