30/12/2019

Lord Darwil

The dark mind

CCTL PHP

Fiches de révisions

# Définir les caractéristiques des variables

Une variable est un conteneur permettant de stocker des informations, en PHP elle commence par le signe **$** suivi du nom de la variable.

Exemple : $VISIO

En PHP elles sont dynamiquement typées c’est-à-dire qu’en fonction de la valeur entrée elles se munissent du type approprié.

Remarque : L’on n’a pas besoin de déclarer une variable avant de l’utiliser contrairement au langage comme le C.

Les variables PHP doivent respecter certaines règles à savoir :

* Une variable commence par le $signe, suivi du nom de la variable
* Un nom de variable doit commencer par une lettre ou le caractère de soulignement
* Un nom de variable ne peut pas commencer par un nombre
* Un nom de variable ne peut contenir que des caractères alphanumériques et des traits de soulignement (Az, 0-9 et \_)
* Les noms de variable sont sensibles à la casse ($age et $AGE sont deux variables différentes)

Elles peuvent être déclarées n’importe ou dans le script , en PHP l’on a 3 portée de variables différentes à savoir :

* Local : Ce sont les variables déclarées lors de la création de fonctions

Exemple :

<?php  
function myTest() {  
    $x = 5; // local scope  
    echo "<p>Variable x inside function is: $x</p>";  
}  
myTest();  
  
// using x outside the function will generate an error  
echo "<p>Variable x outside function is: $x</p>";  
?>

* Global : ce sont les variables déclarées en des fonctions. Elles peuvent être appelées grâce au mot clé **global**

Exemple :

<?php  
$x = 5; // global scope  
  
function myTest() {  
    // using x inside this function will generate an error  
    echo "<p>Variable x inside function is: $x</p>";  
}  
myTest();  
  
echo "<p>Variable x outside function is: $x</p>";  
?>

* Statique : cette portée est attribue pour assurer la continuité de vie des variables locales car lors de l’exécution d’une fonction les valeurs de variables sont écrasées donc on l’utilise pour les rendre statique , de manière superficiel immortel.

Exemple :

<?php  
function myTest() {  
    static $x = 0;  
    echo $x;  
    $x++;  
}  
  
myTest();  
myTest();  
myTest();  
?>

# Identifier les specificites des tableaux

Un tableau est une variable pouvant contenir plusieurs valeurs à la fois. On les crée en PHP grâce à la fonction **array().**

On distingue trois types de tableaux à savoir :

* Les tableaux indexés : se sont des tableaux a index numerique. Pour parcourir et imprimer toutes les valeurs d'un tableau indexé, vous pouvez utiliser une forbouclé

Exemple : $cars = array("Volvo", "BMW", "Toyota");

$cars[0] = "Volvo";  
 $cars[1] = "BMW";  
 $cars[2] = "Toyota";

* Les tableaux associatifs sont des tableaux qui utilisent des clés nommées que vous leur affectez. Pour parcourir et imprimer toutes les valeurs d'un tableau associatif, vous pouvez utiliser une bouclé **foreach**

Il existe deux façons de créer un tableau associatif : **$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");**

**Exemple :**

<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
  
foreach($age as $x => $x\_value) {  
    echo "Key=" . $x . ", Value=" . $x\_value;  
    echo "<br>";  
}  
?>

* Les tableaux multidimensionnels : se sont des tableaux contenant un ou plusieurs tableaux. Nous pouvons également mettre une bouclé for dans une autre bouclé for pour obtenir les éléments du tableau

Exemple :

$cars = array  
  (  
  array("Volvo",22,18),  
  array("BMW",15,13),  
  array("Saab",5,2),  
  array("Land Rover",17,15)  
  );

Pour fusionner les tableaux en php l’on peut utiliser :

* Array\_merge

<?php  
$array1 = array("color" => "red", 2, 4);  
$array2 = array("a", "b", "color" => "green", "shape" => "trapezoid", 4);  
$result = array\_merge($array1, $array2);  
print\_r($result);  
?>

* Array\_merge\_recursive
* L’operateur + : Exemple : $elfes +$miniferme

# Connaitre les fonctions natives

En PHP on distingue plusieurs fonctions natives tel que :

* **Echo**: affiche les chaines de caractères
* **\_\_toString()** :pour transtyper un objet en string
* **Print** : affiche une chaine de caractères
* **Implode** : rassemble les éléments d’un tableau dans une chaine

Exemple :

<?php  
  
$array = array('lastname', 'email', 'phone');  
$comma\_separated = implode(",", $array);  
  
echo $comma\_separated; // lastname,email,phone  
  
// Chaîne vide lors de l'emploi d'un tableau vide :  
var\_dump(implode('hello', array())); // string(0) ""  
  
?>

* **Explode**: scinde une chaine de caractères en segments
* **Var\_dump :** affiche les informations d’une variable (son type, sa valeur)
* **Print\_r :**  affiche les informations lisibles pour une variable

<pre>  
<?php  
$a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x', 'y', 'z'));  
print\_r ($a);  
?>  
</pre>

**Resultat :**

<pre>

Array

(

[a] => apple

[b] => banana

[c] => Array

(

[0] => x

[1] => y

[2] => z

)

)

</pre>

* **Printf**: afficheune chaine de caractere formatee.

<?php  
$n =  43951789;  
$u = -43951789;  
$c = 65; // ASCII 65 is 'A'  
  
// notez le double %%, cela affiche un caractère '%' littéral  
printf("%%b = '%b'\n", $n); // représentation binaire  
printf("%%c = '%c'\n", $c); // affiche le caractère ascii, comme la fonction chr()  
printf("%%d = '%d'\n", $n); // représentation standard d'un entier  
printf("%%e = '%e'\n", $n); // notation scientifique  
printf("%%u = '%u'\n", $n); // représentation entière non signée d'un entier positif  
printf("%%u = '%u'\n", $u); // représentation entière non signée d'un entier négatif  
printf("%%f = '%f'\n", $n); // représentation en virgule flottante  
printf("%%o = '%o'\n", $n); // représentation octale  
printf("%%s = '%s'\n", $n); // représentation chaîne de caractères  
printf("%%x = '%x'\n", $n); // représentation hexadécimal (minuscule)  
printf("%%X = '%X'\n", $n); // représentation hexadécimal (majuscule)  
  
printf("%%+d = '%+d'\n", $n); // indication du signe pour un entier positif  
printf("%%+d = '%+d'\n", $u); // indication du signe pour un entier négatif  
?>

L'exemple ci-dessus va afficher :

%b = '10100111101010011010101101'

%c = 'A'

%d = '43951789'

%e = '4.39518e+7'

%u = '43951789'

%u = '4251015507'

%f = '43951789.000000'

%o = '247523255'

%s = '43951789'

%x = '29ea6ad'

%X = '29EA6AD'

%+d = '+43951789'

%+d = '-43951789'

* Sprintf : retourne une chaine formatee

Exemple :

<?php  
$num = 5;  
$location = 'bananier';  
  
$format = 'Il y a %d singes dans le %s';  
echo sprintf($format, $num, $location);  
?>

Resultat : Il y a 5 singes dans le bananier

* Fprintf : ecrit une chaine formatee dans un flux

Exemple :

<?php  
if (!($fp = fopen('date.txt', 'w'))) {  
    return;  
}  
  
fprintf($fp, "%04d-%02d-%02d", $year, $month, $day);  
// écrira la date formatée ISO dans le fichier date.txt  
?>

* Vprintf : affiche une chaine formatee

Exemple :

<?php  
vprintf("%04d-%02d-%02d", explode('-', '1988-8-1'));  
?>

Resultat : 1988-04-01

* Vsprintf : retourne une chaine formatee dans un flux
* **Include :** permet d’inclure du texte /code / balisage d’un fichier dans un autre
* **Require :**  permet d’inclure un fichier php dans un autre de tel sorte qu’en cas d’echec cela produira une erreur fatale E\_COMPILE\_ERROR et arretera le script contrairement a include qui continuera et donnera juste un E\_WARNING

Exemple :

<?php  
// Exemple 1  
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";  
**$pieces = explode(" ", $pizza);  
echo $pieces[0]; // piece1  
echo $pieces[1]; // piece2  
  
// Exemple 2  
$data = "foo:\*:1023:1000::/home/foo:/bin/sh";  
list($user, $pass, $uid, $gid, $gecos, $home, $shell) = explode(":", $data);  
echo $user; // foo  
echo $pass; // \*  
  
?>**

* **Fdf\_create**: cree un nouveau document PDF
* **Fbsql\_create\_db** : cree une base de donnees

# Gerer les sessions

Les sessions sont le moyen de preserver des donnees entre plusieurs acces. Chaque visiteur accédant à votre page web se voit assigner un identifiant unique, appelé "identifiant de session". Il peut être stocké soit dans un cookie, soit propagé dans l'URL. Le support des sessions vous permet de stocker des données entre les requêtes dans le tableau super-globale [*$\_SESSION*](https://www.php.net/manual/fr/reserved.variables.session.php). Lorsqu'un visiteur accède à votre site, PHP va vérifier automatiquement (si [session.auto\_start](https://www.php.net/manual/fr/session.configuration.php" \l "ini.session.auto-start)est activé) ou sur demande (explicitement avec [session\_start()](https://www.php.net/manual/fr/function.session-start.php)) s'il existe une session du même nom. Si c'est le cas, l'environnement précédemment sauvé sera recréé.

Les sessions sont un moyen simple de stocker des données individuelles pour chaque utilisateur en utilisant un identifiant de session unique. Elles peuvent être utilisées pour faire persister des informations entre plusieurs pages. Les identifiants de session sont normalement envoyés au navigateur via des cookies de session, et l'identifiant est utilisé pour récupérer les données existantes de la session. L'absence d'un identifiant ou d'un cookie de session indique à PHP de créer une nouvelle session, et génère ainsi un nouvel identifiant de session.

Les sessions suivent une cinématique simple. Lorsqu'une session est démarrée, PHP va soit récupérer une session existante en utilisant l'identifiant de session passé (habituellement depuis un cookie de session) ou si aucun identifiant de session n'est passé, il va créer une nouvelle session. PHP va ainsi peupler la variable superglobale [*$\_SESSION*](https://www.php.net/manual/fr/reserved.variables.session.php) avec toutes les données de session une fois la session démarrée. Lorsque PHP s'arrête, il va prendre automatiquement le contenu de la variable superglobale [*$\_SESSION*](https://www.php.net/manual/fr/reserved.variables.session.php), le linéariser, et l'envoyer pour stockage au gestionnaire de sauvegarde de session.

Par défaut, PHP utilise en interne le gestionnaire de sauvegarde files qui est défini via la directive [session.save\_handler](https://www.php.net/manual/fr/session.configuration.php" \l "ini.session.save-handler). Les données de session seront sauvegardées sur le serveur à l'endroit spécifié par la directive de configuration [session.save\_path](https://www.php.net/manual/fr/session.configuration.php" \l "ini.session.save-path).

Les sessions peuvent être démarrées manuellement en utilisant la fonction [session\_start()](https://www.php.net/manual/fr/function.session-start.php). Si la directive de configuration [session.auto\_start](https://www.php.net/manual/fr/session.configuration.php" \l "ini.session.auto-start) est définie à 1, une session démarrera automatiquement lors du début de la demande.

Les sessions s'arrêtent automatiquement lorsque PHP a terminé d'exécuter un script, mais peuvent être stoppées manuellement en utilisant la fonction [session\_write\_close()](https://www.php.net/manual/fr/function.session-write-close.php).

Exemple :

<?php  
// Start the session  
session\_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// Set session variables  
$\_SESSION["favcolor"] = "green";  
$\_SESSION["favanimal"] = "cat";  
echo "Session variables are set.";  
?>  
  
</body>  
</html>

NB : Pour supprimer toutes les variables de session globales et détruire la session, utilisez session\_unset() et session\_destroy()

# Expliquer la mise en œuvre de la connexion a une base de donnees avec PDO

PDO : PHP DATA OBJECT est une interface qui permet au script PHP d’interroger une base de dpnnees via des requetes SQL.

PDO est une extension qui s’ajoute au PHP pour que ses différentes fonctionnalités soient disponibles dans le langage. Tout simplement c’est l’interface d’abstraction de la base de données. Autrement dit il permet de communiquer avec tous les SGBD.

Pour se connecter a une BDD via les PDo l’on commence par instancier l’objet PDO :

**$bdd = new PDO (« mysql :host=localhost ;dbname=mabase », « user», « password») ;**

* Le premier parametre est le **DSN** (data source name) c'est pratiquement le seul paramètre qui change quand on passe d'un SGBD à un autre (d'où le nom **abstraction** de la base de données). Il est possible d'ajouter d'autres mots clés comme **port** pour désigner un numéro de port particulier (sur lequel le SBGD écoute les requêtes) ou **charset** pour spécifier l'encodage (ou jeux de caractères). Ces différents paramètres sont espacés par point virgule.
* Le deuxieme est le nom d’utilisateur créer sur le serveur de bdd
* Le troisieme le mot de passe

si les paramètres ne son pas bien déclarés ou que le serveur de base de données n'est pas accessible alors une exception de type **PDOException** (vue dans la page précédente) est lancée automatiquement. Il faut donc la rattraper comme ceci:

**<?php**

**try{**

**$pdo=new PDO("mysql:host=localhost;dbname=mabase","user","1234");**

**}**

**catch(PDOException $e){**

**echo $e->getMessage();**

**}**

**?>**

# Manipuler les donnees et mecanismes d’une base de donnees avec PDO

Pour la manipulation des donnees telque les requetes et autres :

La classe PDO nous met une panoplie de fonction a nous disposition pour gerer les differentes actions a savoir :

* **Prepare()** : prepare une requete a l’execution et retourne un objet .

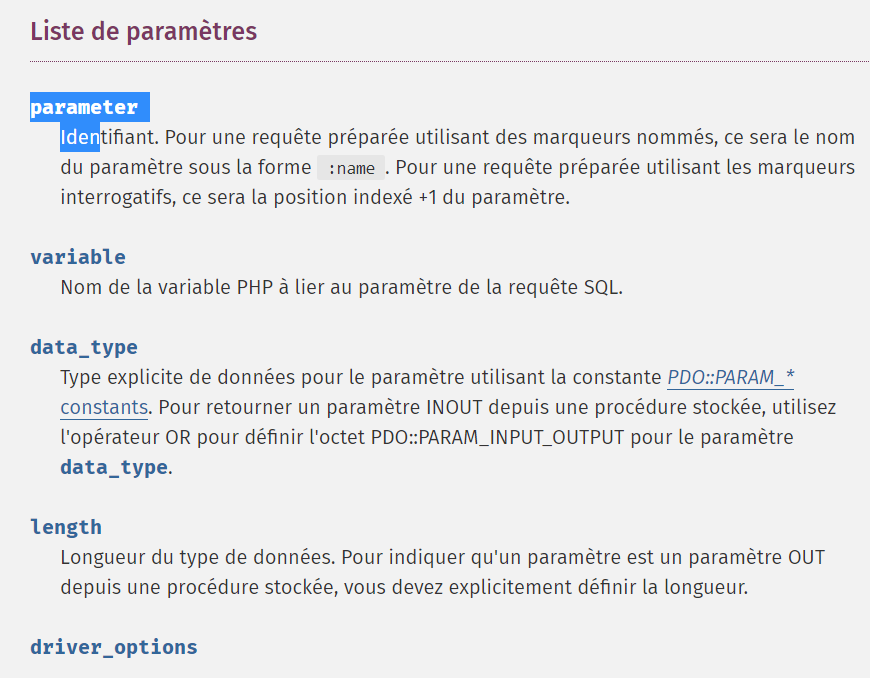
**PDO::prepare** ( string $statement [, array $driver\_options = array() ] ) : [PDOStatement](https://www.php.net/manual/fr/class.pdostatement.php)

* **Execute ()** pour executer une requeter preparer elle suit toujours PDO ::PREPARE
* **Query()** : execute une requet eSQL , retourne un jeu de resultats en tant qu’objet

public **PDO::query** ( string $statement ) : [PDOStatement](https://www.php.net/manual/fr/class.pdostatement.php)

* bindParam : lie un parametre a un nom de variable specifique

public **PDOStatement::bindParam** ( [mixed](https://www.php.net/manual/fr/language.pseudo-types.php#language.types.mixed) $parameter , [mixed](https://www.php.net/manual/fr/language.pseudo-types.php#language.types.mixed) &$variable [, int $data\_type = PDO::PARAM\_STR [, int $length [, [mixed](https://www.php.net/manual/fr/language.pseudo-types.php#language.types.mixed) $driver\_options ]]] ) : bool

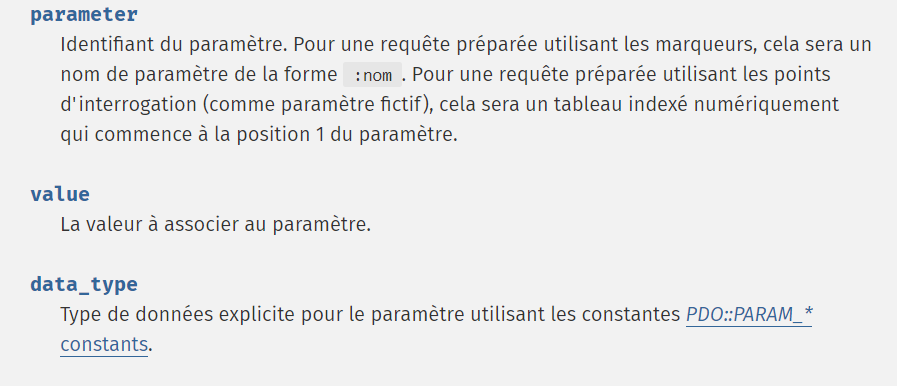


Exemple :

<?php  
/\* Exécution d'une requête préparée en liant des variables PHP \*/  
$calories = 150;  
$couleur = 'rouge';  
$sth = $dbh->prepare('SELECT nom, couleur, calories  
    FROM fruit  
    WHERE calories < :calories AND couleur = :couleur');  
$sth->bindParam(':calories', $calories, PDO::PARAM\_INT);  
$sth->bindParam(':couleur', $couleur, PDO::PARAM\_STR, 12);  
$sth->execute();  
?>

* **bindValue() :** associe une valeur a un parametre.

public **PDOStatement::bindValue** ( [mixed](https://www.php.net/manual/fr/language.pseudo-types.php#language.types.mixed) $parameter , [mixed](https://www.php.net/manual/fr/language.pseudo-types.php#language.types.mixed) $value [, int $data\_type = PDO::PARAM\_STR ] ) : bool



Exemple :

<?php  
/\* Exécute une requête préparée en associant des variables PHP \*/  
$calories = 150;  
$couleur = 'rouge';  
$sth = $dbh->prepare('SELECT nom, couleur, calories  
    FROM fruit  
    WHERE calories < :calories AND couleur = :couleur');  
$sth->bindValue(':calories', $calories, PDO::PARAM\_INT);  
$sth->bindValue(':couleur', $couleur, PDO::PARAM\_STR);  
$sth->execute();  
?>

* **fetch :** Récupère une ligne d'un ensemble de résultats associé à un objet PDOStatement. Le fetch\_styleparamètre détermine la façon dont PDO renvoie la ligne.

public **PDOStatement :: fetch** ([ int $fetch\_style [, int $cursor\_orientation= PDO :: FETCH\_ORI\_NEXT [, int $cursor\_offset= 0 ]]]): [mixte](https://www.php.net/manual/en/language.pseudo-types.php#language.types.mixed)

* **CloseCursor :** ferme le curseur, permettant a l’instruction d’etre executee a nouveau.

Exemple :

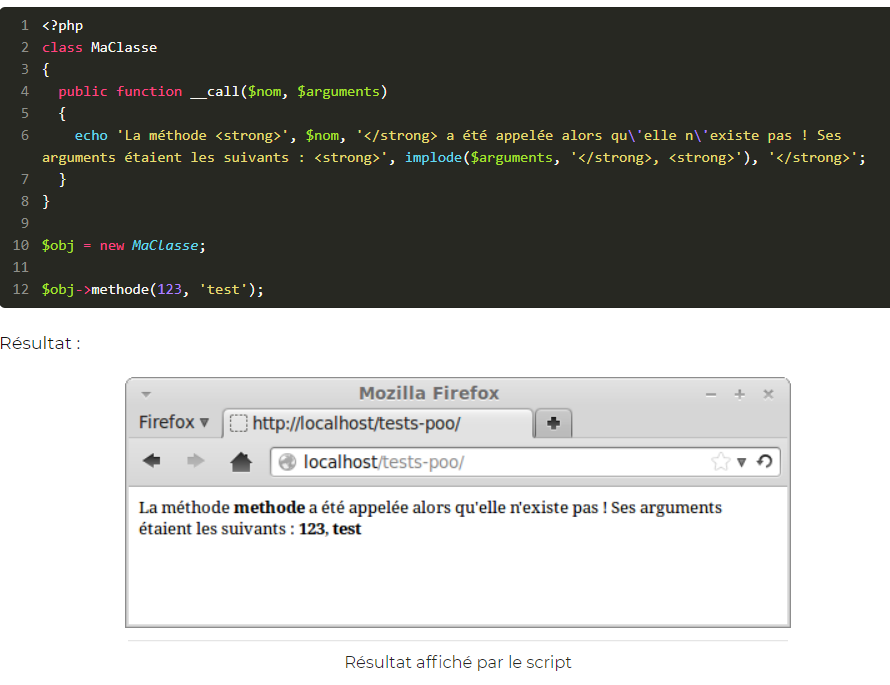
<?php  
/\* Create a PDOStatement object \*/  
$stmt = $dbh->prepare('SELECT foo FROM bar');  
  
/\* Create a second PDOStatement object \*/  
$otherStmt = $dbh->prepare('SELECT foobaz FROM foobar');  
  
/\* Execute the first statement \*/  
$stmt->execute();  
  
/\* Fetch only the first row from the results \*/  
$stmt->fetch();  
  
/\* The following call to closeCursor() may be required by some drivers \*/  
$stmt->closeCursor();  
  
/\* Now we can execute the second statement \*/  
$otherStmt->execute();  
?>

* **PDOStatement :: fetchAll -** Retourne un tableau contenant toutes les lignes du jeu de résultats

# PHP orientee objet :

Le php est un langage oriente objet et procédurale. Donc nous pouvons aussi retrouver les notions de classe , constructeur , héritage…

* Classe : Pour déclarer une classe en PHP on utilise le mot clé **class.** Une classe pouvant avoir une portée private ou publique
* Les méthodes en php se déclarent grâce au mot clé **Function**
* Pour instancier un objet on utilise le mot clé **new** suivi du nom de la classe.
* Le constructeur est la fonction **\_\_Construct**
* Il est conseillé de créer un fichier par classe Les classes peuvent être chargées dynamiquement (c'est-à-dire sans avoir explicitement inclus le fichier la déclarant) grâce à l'auto-chargement de classe (**utilisation despl\_autoload\_register)**Bottom of Form
* Pour déclarer les méthodes statiques on utilise le mot clé **static**
* Et les variables constantes on utilise le mot clé **Const**
* L'opérateur « -> » permet d'accéder à un élément de tel objet, tandis que l'opérateur **« :: »** permet d'accéder à un élément de telle classe.
* Au sein d'une méthode, on accède à l'objet grâce à la pseudo-variable **$this**, tandis qu'on accède à la classe grâce au mot-clé **self.**
* Les méthodes magiques sont des méthodes qui sont appelés automatiquement lorsqu’un certain évènement est déclenche.
* Toutes les méthodes magiques commencent par deux *underscores*, évitez donc d'appeler vos méthodes suivant ce même modèle.
* La méthode**\_\_call** sera appelée lorsque l'on essayera d'appeler une telle méthode. Elle prend deux arguments : le premier est le nom de la méthode que l'on a essayé d'appeler et le second est la liste des arguments qui lui ont été passés (sous forme de tableau).



* La méthode magique**\_\_toString** est appelée lorsque l'objet est amené à être converti en chaîne de caractères. Cette méthode doit retourner la chaîne de caractères souhaitée.
* ?php
* class MaClasse
* {
* protected $texte;
* public function \_\_construct($texte)
* {
* $this->texte = $texte;
* }
* public function \_\_toString()
* {
* return $this->texte;
* }
* }
* $obj = new *MaClasse*('Hello world !');
* // Solution 1 : le cast
* $texte = (string) $obj;
* var\_dump($texte); // Affiche : string(13) "Hello world !".
* // Solution 2 : directement dans un echo
* echo $obj; // Affiche : Hello world !
* Le mot clé **extends**  pour traduire les héritages
* Le mot clé **Interface** pour les interfaces