BASE

Insertion PHP :
```
<?php
        echo " Hello world ";
        print " Hello world ";
?>
```

Commentaire :
```
// Commentaire
/* commentaire */
<!--commentaire -->
```

Concaténation :
```
echo "Bonjour,". " ". "tout le monde ". "!";
Ou simplement :
$txt = "You";
echo "I love $txt!";
Ou :
$x = 5;
$y = 4;
echo $x + $y;
```

Scope globale et local:
```
$x = 5; // global scope
function myTest() {
  $x = 0; // local scope
}
```

VARIABLES

Déclaration de variable
```
$nomVariable = "Chaîne";
$nomVariable = 1234;
```

Affichage de variable
```
echo $x; //affichage de la valeur
print $x; //affichage de la valeur
var_export($x); //affichage de la représentation PHP
print_r($x); //affichage du contenu
var_dump($x); //affichage du type et du contenu
```

Variable globale
```
//Une variable n'est visible que dans lafonction dans laquelle elle a été définie. Global permet d'outrepasser cette restriction.
$x = 5;
$y = 10;
function myTest() {
   global $x, $y;
   $y = $x + $y;
}
myTest();
echo $y; // outputs 15
```

Variable static
```
function myTest() {
   static $x = 0;
   echo $x;
   $x++;
}
myTest(); //en sortie : 0
myTest(); //en sortie : 1
myTest(); //en sortie : 2
```

Constante :
```
define("GREETING", "Welcome to W3Schools.com!", true);
echo GREETING;
// True pour activer lecase-insensitive
// Les constantes sont Globale
```

PHP Object :
```
class Car {
   function Car() {
      $this->model = "VW";
   }
}
// create an object
$herbie = new Car();
// show object properties
echo $herbie->model;
```

GESTION DE STRING

```
strlen("winny"); //Taille d'un String
strpos("winny"); //Inverse une String
str_word_count ("winny"); //Nombre de mots
strtoupper( $maPhrase) ; //Met en majuscule
strtolower( $maPhrase) ; //Met en miniscule
substr("cadeau", 2, 4); //Donne "deau"
strpos("winny", "in"); //Donne 1/False si non trouvé
str_replace("world", "Dolly", "Hello world!");
// outputs Hello Dolly!
```

Fonctions mathématiques :
```
print round(5.2456);  //Affiche l'arrondi à l'unité
print round(5.2456, 3); //Affiche l'arrondi à 3 chiffres apres la ",".
```

New operator :

**	Exponentiation	$x ** $y	*Result of raising $x to the $y'th power (Introduced in* **PHP 5.6**)

CONDITIONS

Syntaxe du if…else if…else
```
if ($t < "10") {
   echo "Have a good morning!";
} elseif ($t < "20") {
   echo "Have a good day!";
} else {
   echo "Have a good night!";
}
```

Syntaxe du SWITCH :
```
// CLASSIQUE
switch ($i) {
```

```
case "red":
        echo "Your favorite color is red!";
        break;
case "blue":
        echo "Your favorite color is blue!";
        break;
default:
        echo "Your favorite color is black";
  }
//ALTERNATIVE
switch ($i):
            //code
endswitch;
```

Syntaxe ternaire:

```
echo (1==1) ? 'vrai' : 'faux';
echo (1==2) ? 'vrai' : 'faux';
//EX
$id = isset($_GET['id']) ? (int)$_GET['id'] : 0;
//la valeur entière de $_GET['id'] ou zéro si la variable n'existe pas
```

BOUCLES

Syntaxe WHILE :

```
$lol = true;
while($lol) {
echo "Hého !";
$lol = false;   }
```

SyntaxeWHILE alternative:

```
while(cond):
   // le code de la boucle est ici
endwhile;
```

Syntaxe DO - WHILE:

```
$i = 0;
do {
    echo $i;
} while ($i > 0);
```

Syntaxe FOR:

```
for ($i = 1; $i < 100; $i += 4) {
        echo "<p>$i</p>";
      }
```

Syntaxe FOREACH:

```
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
//OU
foreach($membres as $i => $membre)
{
//affiche tour à tour "0 BrYs" puis "1 mathieu" puis "2 Yogui"
echo $i . ' ' . $membres[$i];
}
```

TABLEAU/ARRAY

Déclaration de tableau:

```
$tableau = array("Oeuf", "Tomate", 52);
```

Affichage du tableau:

```
print_r($positions);
```

Parcourir un tableau :

```
foreach($tableau as $tab) {
        print "<p>$tab</p>";
      }
//ou
foreach($nombres as $i => $nombre)
{
echo $i.' '.$nombre.'<br/>';
}
//ou
while(list($i, $nombre) = each($nombres))
{
echo $i.' '.$nombre.'<br/>';
}
```

Tableaux associatifs :

```
$myAssocArray = array('year' => 2012,
                      'colour' => 'blue',
                      'doors' => 5);
//Chercher dans ce tableau
echo $myAssocArray['colour'];
```

Parcourir un tableau associatif:

```
foreach ($nomTableau as $Cle=>$Valeur) {
  echo 'Le téléphone de ' . $ Cle . ' est ' . $ Valeur . '. <br>';
}
```

Afficher un élément d'un tableau :

```
echo $monTableau[0] ; //0 est le 1er élément
// OU
print $dizaines{2} ;
```

Supprimer l'élément d'un tableau :

```
unset($tableau[4]);
```

Tableaux multidimensionnels :

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );//Chercher dans ce tableau
echo 'La voiture' . $cars[2][0] . '!';
```

Fonction de tableau :

```
array_push ($nomTableau, "nomElement"); //Ajoute un élément au tableau
count($couleursPreferes); //Nombre d'éléments du tableau
sort($monTableau); //Range dans l'ordre croissant
rsort($monTableau); //Range dans l'ordre décroissant
join(",", $monTableau); //Affiche les éléments du tableau en les séparant de ","
asort ( ) // tri des tableaux associatifs dans l'ordre croissant , selon la valeur
ksort ( ) // tri des tableaux associatifs dans l'ordre croissant , selon la clé
arsort ( ) // tri des tableaux associatifs dans l'ordre décroissant , selon la valeur
krsort ( ) // tri des tableaux associatifs dans l'ordre décroissant , selon la clé
list() : Assigne plusieurs valeurs en une opération (habituellement depuis un tableau) ;
current() : Retourne l'élément du tableau désigné par son pointeur interne ;
reset() : Réinitialise le pointeur interne du tableau ;
```

next() : *Avance le pointeur interne puis agit comme current() ;*
prev() : *Recule le pointeur interne puis agit comme current().*

INCLUDE

• **include()** : Inclure le code du script indiqué, lancer un avertissement si le fichier est introuvable ;
• **require()** : Inclure le code du script indiqué, lancer une erreur fatale si le fichier est introuvable ;
Chacune de ces instructions se décline en une instruction ***_once()** qui oblige PHP à vérifier si le script demandé a déjà été inclus au cours de la requête actuelle (très pratique pour les déclarations de fonctions et de classes).
Les fonctions *_once() ne sont pas nécessairement plus lentes à l'exécution que leurs grandes soeurs, mais elles consomment légèrement plus de mémoire. La différence étant dérisoire, il est inutile de s'en soucier.

AVANCE

SUPERGLOBALES

$_GET : Les valeurs provenant de l'URL ;
$_POST : Les valeurs envoyées par formulaire ;
$_FILE : Les fichiers envoyés par formulaire ;
$_SERVER : Les valeurs mises en place par le serveur Web (elles peuvent donc changer d'une configuration à l'autre) ;
$_ENV : Les variables d'environnement (système d'exploitation) ;
$_SESSION : Les valeurs mises dans le magasin des sessions ;
$_COOKIE : Les valeurs transmises au moyen de cookies par le navigateur ;
$GLOBALS : L'ensemble des variables du script.
$_REQUEST

CONSTANTES MAGIQUES

__LINE__ : La ligne de code en cours ;
__FILE__ : Le nom complet du script en cours ;
__DIR__ : Le nom du répertoire du script en cours (depuis les versions 5.3 et 6.0 de PHP) ;
__FUNCTION__ : La fonction en cours ;
__CLASS__ : La classe en cours, similaire à **get_class($this)** ;
__METHOD__ : La méthode en cours ;
__NAMESPACE__ : L'espace de noms en cours (depuis les versions 5.3 et 6.0 de PHP).

FONCTIONS UTILES

function_exists() : Est-ce que la fonction existe ?
get_defined_functions() : Liste des fonctions définies
func_num_args() : Nombre de paramètres transmis à la fonction courante
func_get_arg() : Un paramètre de la fonction courante
func_get_args() : Tous les paramètres de la fonction courante
create_function() : Créer une fonction utilisateur à partir du code PHP fourni en commentaires.
define() déclare une nouvelle constante
defined() permet de savoir si une constante est déjà définie
get_defined_constants() retourne toutes les constantes définies
contant() retourne la valeur d'une constante

PHP $_SERVER

$_SERVER['REQUEST_METHOD']
La méthode d'appel
POST
$_SERVER['SERVER_NAME']
Nom du serveur
localhost
$_SERVER['SERVER_ADMIN']
L'email de l'administrateur du serveur
email@domaine.tld
$_SERVER['SERVER_ADDR']
L'Adresse IP du serveur
195.14.0.256
$_SERVER['QUERY_STRING']
Les paramètres indiquées à votre script
url=toto.html&id=234
$_SERVER['REMOTE_PORT']
Port HTTP de la requête
80
$_SERVER['REMOTE_ADDR']
Adresse IP de l'internaute
88.101.2.255
$_SERVER['REQUEST_URI']
Chemin du script
/exemple.php
$_SERVER['PATH_TRANSLATED']
Chemin physique (complet) du script
/home/www/domain.fr/example.php
$_SERVER['HTTP_USER_AGENT']
User agent du navigateur du client
Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.1) Gecko/20061010 Firefox/2.0
$_SERVER['HTTP_REFERER']
L'URL de la page d'où provient l'internaute
http://oseox.fr/exemple.php
$_SERVER['HTTP_HOST']
Le nom de domaine où est exécuté le script
example.fr
$_SERVER['HTTP_ACCEPT_LANGUAGE']
Langue acceptée par le navigateur de l'internaute
fr
$_SERVER['DOCUMENT_ROOT']
Adresse de la racine du serveur
/var/www/exemple.fr

AFFICHER/MODIFIER LE TYPE D'UNE VARIABLE

var_dump() function returns the data type and value.

```
echo gettype($variable);
 //affiche le type de la variable
echo get_resource_type($variable);
//affiche le type de la ressource
settype($x, 'int'); //Modifie le type de la variable
intval('5.2 ans'); //Idem
floatval('5.2 ans'); // Idem
strval('5.2 ans'); // Idem
```

```
var_dump((int) 5.2);
var_dump((bool) 5.2);
```

```
var_dump((float) 5);
var_dump((string) 5.2);
// Transtypage (modifier le type d'une valeur de manière ponctuelle)
```

NAMESPACE

Déclaration:
namespace <Nom>;
Utilisation:
use <Nom> as <Alias>;
Namespaces composés:
namespace Cours::Models;
use Cours::Models; //*Sans alias*
use Cours::Models as M; //*Avec alias*
Namespaces non composés:
namespace Models;
use Models as M;

AUTOLOAD

Une fonction particulièrement utile pour les classes est __autoload(). Si cette fonction magique est déclarée dans vos scripts, alors toute classe utilisée mais n'ayant pas été chargée jusque-là, est chargée à l'aide de cette fonction.

```
function __autoload($class)
{
require_once $class.'.php';
}
$object = new MyClass();
//chargement automatique de "MyClass.php"
```

VALIDATION DES DONNEES

```
if(!empty($_GET['id']) and ctype_digit($_GET['id']))
//OU
if(!empty($_POST['password']) and
ctype_print($_POST['password']))
```

FILTRAGE DES DONNEES

```
$id = (int)$_GET['id'];
//on transtype "id" au type numérique entier
$clean['id'] = (int)$_GET['id'];
//on transtype "id" au type numérique entier. Les variables filtrées dans un tableau PHP afin de mettre en valeur le fait qu'elles sont filtrées.
```

UTILISATION DES DONNEES

Si l'on souhaite afficher la valeur dans une page Web, il faut utiliser une des fonctions suivantes :
• utf8_encode() : Pour afficher au format UTF-8 (approche recommandée pour les chaînes UTF-8) ;
• htmlentities() : Pour convertir tous les caractères en leur entité HTML correspondante, attention à bien utiliser les deux paramètres optionnels ;
• htmlspecialchars() : Pour convertir uniquement les entités HTML fondamentales (fonction insuffisante si elle est utilisée seule).

EX CONTRE FAILLE XSS (cross-site scripting):

```
echo htmlentities("Je suis développeur PHP", ENT_QUOTES, 'UTF-8');
```

EX CONTRE FAILLE CSRF (cross-site request forgery):

EXPLIQUER DANS LES FORMULAIRES !!

EX CONTRE FAILLE INJECTION:

```
// Dans MySQL
$string = mysql_real_escape_string($string);
//Ou à l'aide de API OO comme PDO
```

COOKIE

Créer un cookie :

```
setcookie(name, value, expire, path, domain, secure, httponly);

$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
```

Modifier un cookie :
To modify a cookie, just set (again) the cookie using the setcookie() function

Supprimer un cookie :
To delete a cookie, use the setcookie() function with an expiration date in the past

```
setcookie("user", "", time() - 3600);
```

SESSION

Démarrer la session :

```
session_start();
```

Créer une variable de session :

```
$_SESSION["favcolor"] = "green";
```

Afficher toutes les variables de session :

```
print_r($_SESSION);
```

Détruire la session session :

```
// remove all session variables
session_unset();
// destroy the session
session_destroy();
```

FILTER

Constant	ID	Description
FILTER_VALIDATE_BOOLEAN	258	Validates a boolean
FILTER_VALIDATE_EMAIL	274	Validates an e-mail address
FILTER_VALIDATE_FLOAT	259	Validates a float
FILTER_VALIDATE_INT	257	Validates an integer
FILTER_VALIDATE_IP	275	Validates an IP address
FILTER_VALIDATE_REGEXP	272	Validates a regular expression
FILTER_VALIDATE_URL	273	Validates a URL
FILTER_SANITIZE_EMAIL	517	Removes all illegal characters from an e-mail address
FILTER_SANITIZE_ENCODED	514	Removes/Encodes special characters

FILTER_SANITIZE_MAGIC_QUOTES	521	Apply addslashes()
FILTER_SANITIZE_NUMBER_FLOAT	520	Remove all characters, except digits, +- and optionally .,eE
FILTER_SANITIZE_NUMBER_INT	519	Removes all characters except digits and + -
FILTER_SANITIZE_SPECIAL_CHARS	515	Removes special characters
FILTER_SANITIZE_FULL_SPECIAL_CHARS		
FILTER_SANITIZE_STRING	513	Removes tags/special characters from a string
FILTER_SANITIZE_STRIPPED	513	Alias of FILTER_SANITIZE_STRING
FILTER_SANITIZE_URL	518	Removes all illegal character from s URL
FILTER_UNSAFE_RAW	516	Do nothing, optionally strip/encode special characters
FILTER_CALLBACK	1024	Call a user-defined function to filter data

filter_var () de PHP Fonction

La fonction filter_var () à la fois de valider et de désinfecter les données et elle filtre une seule variable avec un filtre spécifique. Il faut deux éléments de données:

La variable que vous souhaitez vérifier

Le type de contrôle à utiliser

Nettoyer une chaîne :

```
$str = "<h1>Hello World!</h1>";
$newstr = filter_var($str, FILTER_SANITIZE_STRING);
```

Valider un int :

```
if (filter_var($int, FILTER_VALIDATE_INT) === 0 ||
!filter_var($int, FILTER_VALIDATE_INT) === false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
```

Nettoyer et valider une @mail :

```
$email = "john.doe@example.com";

// Remove all illegal characters from email
$email = filter_var($email, FILTER_SANITIZE_EMAIL);

// Validate e-mail
if (!filter_var($email, FILTER_VALIDATE_EMAIL) ===
false) {
    echo("$email is a valid email address");
} else {
    echo("$email is not a valid email address");
}
```

Nettoyer et valider une URL :

```
$url = "http://www.w3schools.com";

// Remove all illegal characters from a url
$url = filter_var($url, FILTER_SANITIZE_URL);

// Validate url
if (!filter_var($url, FILTER_VALIDATE_URL) === false) {
    echo("$url is a valid URL");
} else {
    echo("$url is not a valid URL");
}
```

ERROR

Gestion d'erreur simple avec Die( ):

```
if(!file_exists("welcome.txt")) {
  die("File not found");
} else {
  $file=fopen("welcome.txt","r");
}
```

Gestion d'erreur personnalisée:

```
error_function(error_level,error_message,
error_file,error_line,error_context)
EX :
if (!Ora_Logon($username, $password)) {
  error_log("Base Oracle indisponible !", 0);
}
```

EXCEPTION

Gestion d'exception simple:

```
try {
    $dbh = new PDO('mysql:host = localhost;dbname=test',
$user, $pass);
            //Instructions
} catch (PDOException $e) {
    print "Erreur !: " . $e->getMessage() . "<br/>";
    die();
}
```

DATE

```
echo "Today is " . date("Y/m/d") . "<br>";
&copy; 2010-<?php echo date("Y")?>

date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");

$d=strtotime("tomorrow");
echo date("Y-m-d h:i:sa", $d) . "<br>";
$d=strtotime("next Saturday");
echo date("Y-m-d h:i:sa", $d) . "<br>";
$d=strtotime("+3 Months");
echo date("Y-m-d h:i:sa", $d) . "<br>";
```

MAIL TEXT

Eléments complet :
http://a-pellegrini.developpez.com/tutoriels/php/mail/ - L1

```
<?php
$sujet = 'Sujet de l\'email';
$message = "Bonjour,
```

```
Ceci est un message texte envoyé grâce à
php.
merci :)";
$destinataire =
'destinataire@domaine.com';
$headers = "From: \"expediteur
moi\"<moi@domaine.com>\n";
$headers .= "Reply-To:
moi@domaine.com\n";
$headers .= "Content-Type: text/plain;
charset=\"iso-8859-1\"";
if(mail($destinataire,$sujet,$message,$he
aders))
{
echo "L'email a bien été envoyé.";
}
else
{
echo "Une erreur s'est produite lors de
l'envois de l'email.";
}
?>
```

MAIL HTML

```
<?php
$sujet = 'Sujet de l\'email';
$message = "Bonjour,<br />
<strong>Ceci est un message html envoyé
grâce à php.</strong><br />
merci :)";
$destinataire =
'destinataire@domaine.com';
$headers = "From: \"expediteur
moi\"<moi@domaine.com>\n";
$headers .= "Reply-To:
moi@domaine.com\n";
$headers .= "Content-Type: text/html;
charset=\"iso-8859-1\"";
if(mail($destinataire,$sujet,$message,$he
aders))
{
echo "L'email a bien été envoyé.";
}
else
{
echo "Une erreur s'est produite lors de
l'envois de l'email.";
}
?>
```

POO/FONCTION/CLASSE

POO

Syntaxe Classe :

```
class Nomdeclasse {
}
$variable1 = new Nomdeclasse(); //Création d'un new objet
```

Syntaxe Constructeur :

```
public function __construct($param1,$param2) {
  $this->prop1 = $param1 ;
  $this->prop2 = $param2 ;}
```

Affiche le contenu d'une propriété d'un objet :

```
print $nomObjet -> nomParametre;
```

Syntaxe METHODE dans une classe :

```
public function MaMethode($monParametre) {
   // Faire quelque chose
};
//L'appeler
$objet->MaMethode($parametre);
```

Fonctions pour Objet:

```
is_a($nomObjet, "NomClasse"); //Vérifie si c'est bien l'objet de la
classe donnée
property_exists($me, "name"); //Vérifie si un objet possède la
propriété donnée
method_exists($me, "dance"); //Vérifie si un objet à bien la
méthode donnée
```

Héritage :

```
<?php
      class Vehicle {
        public function honk() {
          return "HONK HONK!";
        }
      }

      class Bicycle extends Vehicle {
          public function honk() {
          return "Beep beep!";
        }
      }

      $bicycle = new Bicycle();
      print $bicycle -> honk();

    ?>
//Affichera Beep beep!
```

FINAL :

```
// En reprenant l'exercice précédent, si la méthode publique est définit en
final, elle ne peut être redéfinit plutard
      class Vehicle {
        final public function honk() {
          return "HONK HONK!";
        }
      }
```

Constante :

```
const alive = true; //Déclaration d'une constante
Immortel::alive //Utilisation de la constante (alive)en dehors de sa
classe (Immortel)
```

Static :

```
//Autorisez son utilisation sans avoir besoin d'instance cette classe pour
obtenir un objet.
class King {
       public static function proclaim() {
          echo "A kingly proclamation!";
        }
      }
      King::proclaim();
```

FUNCTION

Syntaxe fonctions :

```
function NomDeMaFonction(paramètres) {
  //instructions;
}
NomDeMaFonction(); //Appelle la fonction
//EX :
function pourcent($x, $y)
{
return $x * 100 / $y;
}
var_dump(pourcent(3, 8));
```

CLASSE

class : Déclaration de classe ;
const : Déclaration de constante de classe ;
function : Déclaration d'une méthode ;
public/**protected**/**private** : Accès (par défaut "public" si aucun accès n'est explicitement défini) ;
new : Création d'objet ;
self : Résolution de portée (la classe elle-même) ;
parent : Résolution de portée (la classe "parent") ;
static : Résolution de portée (appel statique) disponible depuis PHP 5.3 et 6.0 ;
extends : Héritage de classe ;
implements : Implémentation d'une interface (dont il faut déclarer toutes les méthodes).

Les mots clefs "self" et "parent" sont utiles pour accéder à une propriété ou méthode (statique ou non) de la classe elle-même ou de son parent.
Le mot clef "static" a la même utilité mais il résout la portée au moment de l'exécution du script (*cf.* plus loin).

Les **méthodes magiques** sont des méthodes qui, si elles sont déclarées dans une classe, ont une fonction déjà prévue par le langage.

__construct() : Constructeur de la classe ;
__destruct() : Destructeur de la classe ;
__set() : Déclenchée lors de l'accès en écriture à une propriété de l'objet ;
__get() : Déclenchée lors de l'accès en lecture à une propriété de l'objet ;
__call() : Déclenchée lors de l'appel d'une méthode inexistante de la classe (appel non statique) ;
__callstatic() : Déclenchée lors de l'appel d'une méthode inexistante de la classe (appel statique) : disponible depuis PHP 5.3 et 6.0 ;
__isset() : Déclenchée si on applique **isset()** à une propriété de l'objet ;
__unset() : Déclenchée si on applique **unset()** à une propriété de l'objet ;
__sleep() : Exécutée si la fonction **serialize()** est appliquée à l'objet ;
__wakeup() : Exécutée si la fonction **unserialize()** est appliquée à l'objet ;
__toString() : Appelée lorsque l'on essaie d'afficher directement l'objet : **echo $object;** ;
__set_state() : Méthode statique lancée lorsque l'on applique la fonction **var_export()** à l'objet ;
__clone() : Appelés lorsque l'on essaie de cloner l'objet ;
__autoload() : Cette fonction n'est pas une méthode, elle est déclarée dans le scope global et permet d'automatiser les "include/require" de classes PHP.
class_parents() : Retourne un tableau de la classe parent et de tous ses parents ;
class_implements() : Retourne un tableau de toutes les interfaces implémentées par la classe et par tous ses parents ;
get_class() : Retourne la classe de l'objet passé en paramètre ;
get_called_class() : À utiliser dans une classe, retourne la classe appelée explicitement dans le code PHP et non au sein de la classe ;
class_exists() : Vérifie qu'une classe a été définie ;
get_declared_classes() : Liste des classes définies ;
get_class_methods() : Liste des méthodes d'une classe ;
get_class_vars() : Liste des propriétés d'une classe.

```
//EX
class Caniche
{
private $nbPattes;
public function __construct()
{
$this->nbPattes = 4;
}
public function nbPattes() // "getter"
{
return $this->nbPattes; //ok depuis l'intérieur de la classe
}
}
$froufrou = new Caniche();
echo $froufrou->nbPattes(); //affiche "4"
//EX AVEC RESOLUTION DE PORTEE
class Chien
{
protected function aboyer()
{
return 'Je suis un chien';
}
}
class Chien_Labrador extends Chien
{
protected function aboyer()
{
return 'Je suis un labrador';
}
public function identifierParent()
{
return parent::aboyer();
}
public function identifierSelf()
{
return self::aboyer();
}
}
$médor = new Chien_Labrador();
echo $médor->identifierParent().'<br/>';
echo $médor->identifierSelf().'<br/>';
```

INTERFACES

Une interface est un ensemble de méthodes que les classes doivent définir si elles veulent l'implémenter.

```
//EX
interface Joueur
```

```
{
// Une classe qui veut implémenter l'interface Joueur
// doit définir la méthode jouer()
public function jouer();
}
class Labrador implements Joueur
{
public function jouer()
{
echo 'Ouah!';
}}
$médor = new Labrador();
$médor->jouer();
```

REFERENCES ET CLONAGE

Depuis PHP 5, les objets sont tous des *références*. Ainsi, copier un objet vers un autre au moyen de l'opérateur "=" ne duplique pas l'objet, au contraire il créé une deuxième référence vers le même objet.

```
$object_1 = new stdClass();
$object_2 = $object_1;
unset($object_1); //il reste une référence, l'objet persiste donc en mémoire
unset($object_2); //l'objet n'est plus référencé par aucune variable, il est donc détruit

$obj_1 = new Test();
$obj_2 = clone $obj_1;
//Pour le clonage, chaque élément est indépendant et peut être détruit indépendamment.
```

FORMULAIRE / FICHIER

FORM

Formulaire avec envoi sécurisé :

```
<form action="welcome.php" method=" <?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
```

Vérification des données reçus :

```
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = valid_data($_POST["name"]);
  $email = valid_data($_POST["email"]);
  $website = valid_data($_POST["website"]);
  $comment = valid_data($_POST["comment"]);
  $gender = valid_data($_POST["gender"]);
}
function valid_data($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
```

Vérification d'un nom:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

Vérification d'un email:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

Vérification d'une URL:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

FILE

Lire un fichier :

```
readfile("nomdufichier.txt");
```

r

Open a file for read only. File pointer starts at the beginning of the file

w

Open a file for write only. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file

a

Open a file for write only. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist

x

Creates a new file for write only. Returns FALSE and an error if file already exists

r+

Open a file for read/write. File pointer starts at the beginning of the file

w+

Open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file

a+

Open a file for read/write. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist

x+

Creates a new file for read/write. Returns FALSE and an error if file already exists

Ouvrir un fichier :

```
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
```

Lire un fichier ouvert :

```
echo fread($myfile,filesize("webdictionary.txt"));
```

Fermer un fichier :

```
fclose($myfile);
```

Lire une ligne du fichier :

```
echo fgets($myfile);
```

Lire ligne par ligne jusqu'à la fin du fichier:

```
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
```

Lire caractère par caractère jusqu'à la fin du fichier:

```
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
```

Créer un fichier:

```
$myfile = fopen("testfile.txt", "w")
```

Ecrire dans un fichier:

```
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
```

First, ensure that PHP is configured to allow file uploads. In your "php.ini" file, search for the file_uploads directive, and set it to On:

```
file_uploads = On
```

Upload de fichiers:

```
<?php

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Vérifier si le fichier est bien une image ou un fake
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
// Vérification si le fichier existe déjà
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
// Vérification de la taille du fichier
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Autoriser seulement certains formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
// Vérifier si $uploadOk est set à 0 par une erreur
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
//Si tout est OK, essayer d'upload le fichier
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

PDO

PDO (PhpDataObject)

//EXEMPLE COMPLET:

```
$select_users = $db->prepare('SELECT id, name FROM user');
$select_user = $db->prepare('SELECT id, name FROM user WHERE user_id = ?');
$insert_user = $db->prepare('INSERT INTO user (name, password) VALUES (?, ?)');
//récupère tous les utilisateurs (aucun puisque la table est vide)
$select_users->execute();
$users = $select_users->fetchAll();
$insert_user->execute(array('BrYs', '4321'));
$insert_user->execute(array('mathieu', '4321'));
$insert_user->execute(array('Yogui', '4321'));
$select_users->execute(); //récupère tous les utilisateurs (les 3)
$users = $select_users->fetchAll();
$select_user->execute(array(1)); //récupère l'utilisateur numéro 1
$user_1 = $select_user->fetchAll();
$select_user->execute(array(2)); //récupère l'utilisateur numéro 2
$user_2 = $select_user->fetchAll();
$select_user->execute(array(3)); //récupère l'utilisateur numéro 3
$user_3 = $select_user->fetchAll();
```

//BONNE PRATIQUE

```
$sql = 'SELECT m.id, m.title, u.name AS author
FROM message AS m
INNER JOIN user AS u ON m.user_id = u.id
ORDER BY post_date';
$select_messages = $db->prepare($sql);
$select_messages->setFetchMode(PDO::FETCH_ASSOC);
$select_messages->execute();
header('Content-Type: text/html; charset=utf-8');
foreach($select_messages->fetchAll() as $message)
{
echo utf8_encode(htmlspecialchars(
$message['title'].' par '.$message['author'], ENT_QUOTES))
.'<br/>';
}
```

Ouvrir la connexion en MySql en PDO

```
$serveur = 'localhost';
 $user = 'root';
 $pass  = 'mysql';
 $bdd = 'm2l_manager';

try {
```

```
   $cnx = new PDO('mysql:host='.$serveur.';dbname='.$bdd,
$user, $pass);
  $cnx->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
   $cnx->exec("set names utf8");
}
catch(PDOException $e)
{
   echo $e->getMessage();
}
```

Fermer la connexion en MySql en PDO

```
$cnx = null;
```

SELECT en PDO

```
$select = $db->prepare("SELECT * FROM nom_table
WHERE nom_champ = '".$variable."'");
                        $select->execute();
                        $result = $select->fetchAll();

                        foreach($result as $val)
                        {
                                echo $val["nom_champ"];
                        }
```

```
$select = $cnx->prepare("SELECT * FROM m2lm_user");
$select->execute();
$liste = $select->fetchAll();

foreach ($liste as $val) {
echo '<option value="'.$val["email"].'"> '.$val["firstname"]."
".$val["lastname"]." ".$val["email"].'</option>';
                        }
```

```
$q = array('id'=>$id);

$sql = 'SELECT champ FROM m2lm_user WHERE id = :id';
$req = $cnx->prepare($sql);
$req->execute($q);

$req->setFetchMode(PDO::FETCH_OBJ);
$resultat = $req->fetch();

$validation = $resultat->champ;
```

```
$q = array('id'=>$id);

$sql2 = 'SELECT * FROM m2lm_user WHERE id = :id';
$req2 = $cnx->prepare($sql2);
$req2->execute($q);
$req2->setFetchMode(PDO::FETCH_OBJ);
$resultat = $req2->fetch();

//Stockage des données dans les variables de session
$_SESSION['Auth'] = array(
        'id' => $resultat->id,
        'login' => $resultat->login,
        'email' => $resultat->email,
        'password' => $password,
        'firstname' => $resultat->firstname,
        'connection' => 1
                        );
```

INSERT en PDO

```
// prepare sql and bind parameters
   $stmt = $conn->prepare("INSERT INTO MyGuests
(firstname, lastname, email)
   VALUES (:firstname, :lastname, :email)");
   $stmt->bindParam(':firstname', $firstname);
   $stmt->bindParam(':lastname', $lastname);
   $stmt->bindParam(':email', $email);

   // insert a row
   $firstname = "John";
   $lastname = "Doe";
   $email = "john@example.com";
   $stmt->execute();
```

```
$q = array(
'login'     =>$login,
'email'     =>$email,
'password'  =>$password,
'firstname' =>$firstname,
'lastname'  =>$lastname,
'job'       =>$job,
'domain'    =>$domain,
'token'     =>$token);

$sql = 'INSERT INTO m2lm_user (login, email, password,
firstname, lastname, job, domain, token)
VALUES (:login, :email, :password, :firstname, :lastname, :job,
:domain, :token)';

$req = $cnx->prepare($sql);
$req->execute($q);
```

VERIF en PDO

```
$selectverif = $db->prepare("SELECT * FROM nom_table
WHERE nom_champ = '".$variable."'
 AND nom_champ = '".$variable."'
 AND nom_champ = '".$variable."'
 AND nom_champ = '".$variable."'");
$selectverif->execute();
$val = $selectverif->fetchAll(); //ou fetch();

if(count($val) != 0)
{
$test = " Ligne déjà Existante !";
}
```

```
$q = array('email'=>$email,'token'=>$token);
$sql = 'SELECT email,token FROM m2lm_user WHERE
email = :email AND token = :token';
$req = $cnx->prepare($sql);
$req->execute($q);
$count = $req->rowCount($sql);

if($count == 1){
//instructions...
}
```

UPDATE en PDO

```
$u = array('email'=>$email, 'active'=>'1');
$sql = 'UPDATE m2lm_user SET active = :active WHERE
email = :email';
        $req = $cnx->prepare($sql);
        $req->execute($u);
```

DELETE en PDO

```
$delete = $cnx->prepare("DELETE FROM m2lm_room
WHERE id = '".$_GET['id']."'");
$delete->execute();
```

```
$TRUNCATE = $db->prepare("TRUNCATE TABLE
`nom_table`");
```

$TRUNCATE->execute();

PROCEDURES STOCKEES en PDO

```
$update = $cnx->query("call upUser('$loginup', '$emailup',
'$passwordup', '$firstnameup', '$lastnameup', '$jobup',
'$domainup', '$rankup', '$tokenup', '$activeup', '$idup')");
```

```
$req = $cnx->query("call addUser('$login', '$email', '$password',
'$firstname', '$lastname', '$job', '$domain', '.$token.')");
```

```
$id = $_GET['id'];
$delete = $cnx->query("call delUser('$id')"
```