

Cours de programmation web

PAUL-ANTOINE BISGAMBIGLIA

2021-2022

[HTTP://PAUL-ANTOINE-BISGAMBIGLIA.UNIV-CORSE.FR/](http://PAUL-ANTOINE-BISGAMBIGLIA.UNIV-CORSE.FR/)

@PABISGAMBIGLIA



Licence



- ▶ Cours en licence libre
- ▶ Certaines images ont été récupérées sur google image avec le filtre « réutilisation utilisée sans but commercial »
- ▶ Certains exemples de code sont issus de sites web référencés en fin de chapitre via leurs urls

Objectifs

- ▶ Approfondir les notions de programmation web
- ▶ Apprendre à créer un site internet riche
- ▶ Technologies utilisées
 - ▶ HTML5
 - ▶ CSS3
 - ▶ JavaScript
 - ▶ PHP
 - ▶ MySQL
 - ▶ Des frameworks



Prérequis

Master 1

- ▶ Cours de L3
 - ▶ Algorithmique et programmation
 - ▶ Notion de réseau
 - ▶ Communication client serveur
 - ▶ Protocole HTTP
 - ▶ Techo web : HTML/CSS/JS/PHP

Licence 3

- ▶ Notion d'algorithmique et de programmation
- ▶ Notion de réseau
 - ▶ Communication client serveur
 - ▶ Protocole HTTP

Programme

- ▶ Introduction et révisions
 - ▶ Le web, et ses techno
- ▶ HTML 5
 - ▶ Langage de balise <body> </body>
- ▶ CSS 3 avec bootstrap
 - ▶ Mise en forme
 - ▶ Ergonomie du web
- ▶ JavaScript avec jQuery
 - ▶ Dynamisme coté client
 - ▶ Angular.js
 - ▶ Node.js
- ▶ PHP
 - ▶ Dynamisme coté serveur
 - ▶ Les frameworks (Zend, Symfony, cakephp)
- ▶ Le +
 - ▶ les Design Patterns pour le web
 - ▶ Sécuriser son code
 - ▶ Django le framework web en python
 - ▶ Framework html 5 pour mobile (IONIC, Mobile Angular UI, Intel XDK, Appcelerator Titanium, Sencha Touch)
 - ▶ Apache Cordova, Meteor

PHP

1. LE LANGAGE ET SES LIMITES
2. EXEMPLE D'ENVIRONNEMENT DE PROGRAMMATION 'WAMP'
3. BASES DE PHP
4. PHP, XML ET JSON
5. PHP ET BASES DE DONNÉES
6. PHP ET AJAX
7. BONNES PRATIQUES

PHP

Introduction

- ▶ Le PHP est un langage de script côté serveur
- ▶ Acronyme pour "PHP: Hypertext Preprocessor"
- ▶ PHP est open source
- ▶ Les scripts PHP sont exécutées sur le serveur
- ▶ PHP gratuit

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

PHP

Introduction

- ▶ Les fichiers PHP (.php) peuvent contenir de l'HTML, du CSS, du JS et du PHP
- ▶ Le PHP est exécuté coté serveur et renvoie un flux HTML
- ▶ Langage multiplateforme
- ▶ Compatible avec plusieurs SGBD

Pourquoi l'utiliser :

- ▶ pour générer du contenu dynamique
- ▶ pour créer, ouvrir, lire, écrire, supprimer, et fermer des fichiers sur un serveur
- ▶ pour collecter des données de formulaire
- ▶ pour gérer des cookies
- ▶ pour ajouter, supprimer, modifier des données dans votre base de données
- ▶ ...

PHP

Introduction

- ▶ Syntaxe proche du C
- ▶ Interpréteur souple
- ▶ Types de données classiques
- ▶ Typage à première affectation
- ▶ Conversion automatique de type
- ▶ Tableaux dynamiques à sélecteur
- ▶ Modèle objet intégré

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>
<?php
// This is a single-line comment
echo "Hello World!";
?>

</body>
</html>
```

PHP

ENVIRONNEMENT DE
DÉVELOPPEMENT, UN
EXEMPLE WAMP SERVEUR :

[HTTP://WWW.WAMPSERVER.COM/](http://WWW.WAMPSERVER.COM/)

PHP

Exécution

- ▶ Pour faire du PHP il faut un serveur (wamp par exemple)
- ▶ Pour exécuter du php il faut lancer les pages à partir de l'adresse du serveur (localhost)

URL :

file:///C:/wamp/www/test_php/test.php

Local le php ne marche pas

URL :

http://localhost/test_php/test.php

Sur le serveur le php marche

PHP

file:///C:/wamp/www/test_php/test.php

Bonjour tout le monde en html

Bonjour tout le monde en php

```
"; echo $maVar; ?>  
lien
```

http://localhost/test_php/test.php

Bonjour tout le monde en html

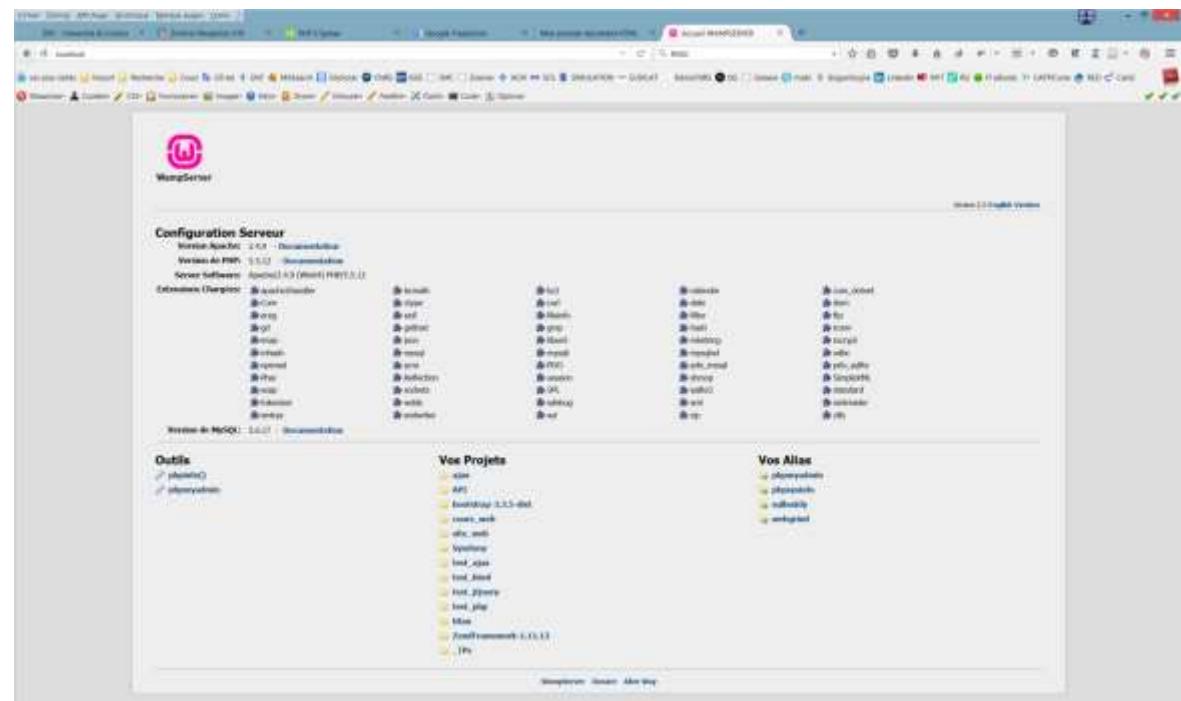
Bonjour tout le monde en php

lien

PHP

Exécution

- ▶ Wamp :
 - ▶ Serveur web
 - ▶ Dossier : www
 - ▶ url : localhost
 - ▶ SGBD mySQL



PHP

Syntaxe

- ▶ Plusieurs formes de commentaire
- ▶ Extension des fichier .php
- ▶ Ligne d'instruction se termine par ;
- ▶ Mots clés non sensible à la casse
 - ▶ Echo = echo = ECHO
- ▶ Variable sensible à la casse

```
<!DOCTYPE html>
<html>
<body>
<?php
// This is a single-line comment
# This is also a single-line comment
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
// You can also use comments to leave out parts of a code
line
$x = 5 /* + 15 */ + 5;
echo $x;
?>
</body>
</html>
```

Les bonnes pratiques php

<http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>

camelCase

- ▶ \$maVariableNumerique = 3;
- ▶ function sommeDesEntiers(\$maximum)
{...}

snake_case

- ▶ \$user = ['username' => 'monadresse@mail.com', 'first_name' => 'John', 'last_name' => 'Doe', 'active' => 1];
- ▶ Tables SQL

Les bonnes pratiques php

<http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>

PascalCase

- ▶ class Aventurier { ... }
- ▶ Chaque classe est enregistrée dans un fichier qui porte le même nom

MAJUSCULE

- ▶ define('CHIFFRE PORTE_BONHEUR', 13);

Les bonnes pratiques php

<http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>

{ }

- ▶ Les accolades sont à la ligne suivant la déclaration d'une classe, d'une fonction ou d'une méthode.
- ▶ Les accolades sont à la suite, précédée d'un espace, pour un if, switch, while, foreach, etc. et sont obligatoires.

Exemple :

```
class Aventurier
{
    // Nom de mon Aventurier
    public $nom = '';
    // Constructeur
    public function __construct($nom)
    {
        $this->nom = $nom;
    }
}
```

Les bonnes pratiques php

<http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>

()

- ▶ On ne met pas d'espace entre le nom d'une fonction et l'appel de ses paramètres. Même chose à la création d'un objet. Seul echo fait exception.

Exemple :

```
$maSomme =  
calculSommeNombresPairs(200);  
echo $maSomme;  
  
$monAventurier = new  
Aventurier('Actarus');  
echo $monAventurier->getNom();
```

Les bonnes pratiques php

<http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>

```
// Au lieu de faire
function trouvePairesDeValeurs($valeurs)
{
    // Je recherche 2 valeurs identiques dans un tableau de valeurs

    // Si j'en ai trouvé, je retourne la valeur de ces paires
    if ($pairesTrouves) {
        return $valeursPairesTrouvees;
    }

    return false;
}

// Préférez faire
function trouvePairesDeValeurs($valeurs)
{
    // Je recherche 2 valeurs identiques dans un tableau de valeurs

    // De cette manière ma fonction retourne toujours la même chose
    return [
        'paires_trouves' => $pairesTrouves,
        'valeurs_paires_trouvees' => $valeursPairesTrouvees
    ];
}
```

Un tableau ne doit contenir qu'un seul type d'informations.
Si vous avez besoin de stocker plusieurs types
d'informations, utilisez un tableau associatif

Les bonnes pratiques php

<http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>

Nommage

- ▶ Les noms des variables, des fonctions, méthodes et classes doivent être le plus explicites possibles.
- ▶ // Evitez absolument les abréviations, qui laissent place à l'interprétation
- ▶ // Pas bon :
- ▶ \$numUser;
- ▶ // Bon :
- ▶ \$numeroTelephoneUser;

Quotes

```
$prenom = 'Jean';  
$piece = 'Cuisine';  
$phrase = $prenom . ' est dans la ' . $piece;  
// Au lieu de :  
$phrase = "$prenom est dans la $piece";
```

Les bonnes pratiques php

<http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>

test

- ▶ Utilisez des constantes pour réaliser des comparaisons de nombre ou de chaînes de caractères. Ce principe est très intéressant pour avoir un code beaucoup plus lisible. De plus, si vous faites une faute de frappe sur une constante, PHP déclenchera une erreur, alors que si vous faites une faute de frappe dans une chaîne de caractères, PHP ne déclenchera pas d'erreur

Quotes

```
// Imaginons cette fois-ci je teste la présence d'un langage dans un tableau de compétences
if (in_array('html', $competences)) {
    // Certaines instructions
}

// Le code ci-dessous n'est pas bon mais ne génère pas d'erreur :
if (in_array('hmtl', $competences)) {
    // Certaines instructions
}

// Alors qu'avec une constante, disons LANGAGE_HTML, ce code génère une erreur :
if (in_array(LANGAGE_HMTL, $competences)) {
    // Certaines instructions
}
```

Les bonnes pratiques php

<http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>

fonction

- ▶ Limiter le nombre de paramètres d'une fonction et favoriser des options
- ▶

```
/** * Counts the number of items in the
provided array. ** @param mixed[]
$items Array structure to count the
elements of. ** @return int Returns the
number of elements. */ function
count(array $items) { <...> }
```

Exemple:

```
function input($name, $type = 'text',
$value = '', $placeholder = '', $id = '',
|required = false)
```

```
{ // Détails de ma fonction }
```

```
function input($name, $options = []) {
```

PHP

Variables

Règles pour les variables PHP:

- ▶ 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 (AZ, 0-9 et _)
- ▶ Les noms de variables sont sensibles à la casse (\$age et \$Age sont deux variables différentes)

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 5;
$y = 4;
echo $x + $y;
$txt = "PHP";
echo "I love " . $txt . "!";
?>
</body>
</html>
```

Variables

Langage faiblement type

Typé dynamiquement

Il y a trois portés de variable

- ▶ local
- ▶ global
- ▶ Static

Une variable déclarée en dehors d'une fonction avec une portée globale ne peut être accessible dans la fonction

```
<?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>";  
?>
```

Variables

Une variable déclarée dans une fonction avec une portée locale ne peut être accessible qu'au sein de cette fonction

```
<!DOCTYPE html>
<html>
<body>

<?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>";
?>

</body>
</html>
```

PHP

Variables

Le mot-clé **global** est utilisé pour accéder à une variable globale dans une fonction

Le langage stocke toutes les variables globales dans un tableau

`$GLOBALS[index]`

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest(); // run function
echo $y; // output the new value for variable $y
?>

</body>
</html>
```

Variables

En PHP, quand une fonction est terminé toutes ses variables sont supprimés. Si, nous souhaitons conserver une variable locale il faut utiliser le mot clé **statique** lorsque l'on déclare la variable

```
<!DOCTYPE html>
<html>
<body>
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}
myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
</body>
</html>
```

Sorties

- ▶ echo
- ▶ print ou print()

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with
multiple parameters.";
?>

</body>
</html>
```

PHP

Types de données

- ▶ String
- ▶ Integer
- ▶ Float (floating point numbers - also called double)
- ▶ Boolean (true false)
- ▶ Array
- ▶ Object
- ▶ NULL
- ▶ Resource

```
<?php  
// String  
  
$x = "Hello world!";  
$y = 'Hello world!';  
  
// Integer  
  
$z = 5985;  
var_dump($z);  
  
// Float  
  
$w = 10.365;  
var_dump($w);
```

Types de données

- ▶ **String**
 - ▶ `strlen`
 - ▶ `str_word_count`
 - ▶ `strrev`
 - ▶ `strpos`
 - ▶ `str_replace`

```
<!DOCTYPE html><html>
<body>
<?php
echo strlen("Hello world!");
// outputs 12
echo str_word_count("Hello world!");
// outputs 2
echo strrev("Hello world!");
// outputs !dlrow olleH
echo strpos("Hello world!", "world");
// outputs 6
echo str_replace("world", "Dolly", "Hello
world!");
// outputs Hello Dolly!
?>
</body></html>
```

PHP

String

addcslashes : Ajoute des slash dans une chaîne, à la mode du langage C

addslashes : Ajoute des antislashes dans une chaîne

chr : Retourne un caractère à partir de son code ASCII

count_chars : Retourne des statistiques sur les caractères utilisés dans une chaîne

crc32 : Calcule la somme de contrôle CRC32

crypt : Hachage à sens unique (indéchiffrable)

html_entity_decode : Convertit toutes les entités HTML en caractères normaux

htmlentities : Convertit tous les caractères éligibles en entités HTML

htmlspecialchars_decode : Convertit les entités HTML spéciales en caractères

htmlspecialchars : Convertit les caractères spéciaux en entités HTML

lcfirst : Met le premier caractère en minuscule

ltrim : Supprime les espaces (ou d'autres caractères) de début de chaîne

md5_file : Calcule le md5 d'un fichier

md5 : Calcule le md5 d'une chaîne

PHP

String

ord : Retourne le code ASCII d'un caractère

parse_str : Analyse une requête HTTP

rtrim : Supprime les espaces (ou d'autres caractères) de fin de chaîne

str_repeat : Répète une chaîne

str_replace : Remplace toutes les occurrences dans une chaîne

strip_tags : Supprime les balises HTML et PHP d'une chaîne

stripcslashes : Décode une chaîne encodée avec addslashes

stripos : Recherche la position de la première occurrence dans une chaîne, sans tenir compte de la casse

stripslashes : Supprime les antislashes d'une chaîne

strstr : Trouve la première occurrence dans une chaîne

String

strtolower : Renvoie une chaîne en minuscules

strtoupper : Renvoie une chaîne en majuscules

trim : Supprime les espaces (ou d'autres caractères) en début et fin de chaîne

substr_compare : Compare deux chaînes depuis un offset jusqu'à une longueur en caractères

substr_count : Compte le nombre d'occurrences de segments dans une chaîne

substr_replace : Remplace un segment dans une chaîne

substr : Retourne un segment de chaîne

PHP

Types de données

- ▶ String
- ▶ Integer
- ▶ Float (floating point numbers - also called double)
- ▶ Boolean (true false)
- ▶ **Array**
- ▶ Object
- ▶ NULL
- ▶ Resource

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>

</body>
</html>
```

Types de données

- ▶ String
- ▶ Integer
- ▶ Float (floating point numbers - also called double)
- ▶ Boolean (true false)
- ▶ Array
- ▶ **Object**
- ▶ NULL
- ▶ Resource

```
<!DOCTYPE html>
<html>
<body>

<?php
class Car {
    function Car() {
        $this->model = "Toyota";
    }
}
// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>

</body>
</html>
```

PHP

Types de données

- ▶ String
- ▶ Integer
- ▶ Float (floating point numbers - also called double)
- ▶ Boolean (true false)
- ▶ Array
- ▶ Object
- ▶ **NULL**
- ▶ Resource

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>

</body>
</html>
```

Constante

- ▶ Pour créer une constante, il faut utiliser le mot clé `define`
 - ▶ `define(name, value, case-insensitive)`

```
<!DOCTYPE html>
<html>
<body>

<?php
define("GREETING", "Bonjour à Tous!");
echo GREETING;
function myTest() {
    echo GREETING;
}

myTest();
?>
</body></html>
```

Variables globales

- ▶ **\$GLOBALS**
- ▶ **\$_SERVER**
- ▶ **\$_REQUEST**
- ▶ **\$_POST**
- ▶ **\$_GET**
- ▶ **\$_FILES**
- ▶ **\$_ENV**
- ▶ **\$_COOKIE**
- ▶ **\$_SESSION**

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>

</body>
</html>
```

PHP

Variables globales

- ▶ `$GLOBALS`
- ▶ `$_SERVER`
- ▶ `$_REQUEST`
- ▶ `$_POST`
- ▶ `$_GET`
- ▶ `$_FILES`
- ▶ `$_ENV`
- ▶ `$_COOKIE`
- ▶ `$_SESSION`

```
<!DOCTYPE html>
<html><body>

<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>

</body></html>
```

Variables globales

- ▶ `$GLOBALS`
- ▶ `$_SERVER`
- ▶ `$_REQUEST`
- ▶ `$_POST`
- ▶ `$_GET`
- ▶ `$_FILES`
- ▶ `$_ENV`
- ▶ `$_COOKIE`
- ▶ `$_SESSION`

```
<!DOCTYPE html>
<html><body>
<form method="post" action=<?php echo
$_SERVER['PHP_SELF'];?>>
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body></html>
```

Variables globales

- ▶ `$GLOBALS`
- ▶ `$_SERVER`
- ▶ `$_REQUEST`
- ▶ `$_POST`
- ▶ `$_GET`
- ▶ `$_FILES`
- ▶ `$_ENV`
- ▶ `$_COOKIE`
- ▶ `$_SESSION`

```
<html><body>

<a href="test_get.php?subject=PHP&web=mon site">Test $GET</a>

</body></html>

<html><body>

<?php
echo "Study " . $_GET['subject'] . " at " .
$_GET['web'];
?>

</body></html>
```

Operateurs

- ▶ Les opérateurs **arithmétiques**
- ▶ Les opérateurs d'affectation
- ▶ Les opérateurs de comparaison
- ▶ Opérateurs d'incrémentation / décrémentation
- ▶ Les opérateurs logiques
- ▶ Opérateurs de chaînes
- ▶ Opérateurs de tableaux

+	Addition	$$x + y
-	Subtraction	$$x - y
*	Multiplication	$$x * y
/	Division	$$x / y
%	Modulus	$$x \% y
**	Exponentiation	$$x ** y

Operateurs

- ▶ Les opérateurs arithmétiques
- ▶ Les opérateurs **d'affectation**
- ▶ Les opérateurs de comparaison
- ▶ Opérateurs d'incrémentation / décrémentation
- ▶ Les opérateurs logiques
- ▶ Opérateurs de chaînes
- ▶ Opérateurs de tableaux

$x = y$	$x = y$	Affectation
$x += y$	$x = x + y$	Addition
$x -= y$	$x = x - y$	Subtraction
$x *= y$	$x = x * y$	Multiplication
$x /= y$	$x = x / y$	Division
$x \% y$	$x = x \% y$	Modulus

Operateurs

- ▶ Les opérateurs arithmétiques
- ▶ Les opérateurs d'affectation
- ▶ Les opérateurs de **comparaison**
- ▶ Opérateurs d'incrémentation / décrémentation
- ▶ Les opérateurs logiques
- ▶ Opérateurs de chaînes
- ▶ Opérateurs de tableaux

<code>==</code>	Equal	<code>\$x == \$y</code>
<code>===</code>	Identical	<code>\$x === \$y</code>
<code>!=</code>	Not equal	<code>\$x != \$y</code>
<code><></code>	Not equal	<code>\$x <> \$y</code>
<code>!==</code>	Not identical	<code>\$x !== \$y</code>
<code>></code>	Greater than	<code>\$x > \$y</code>
<code>>=</code>	Greater than or equal to	<code>\$x >= \$y</code>

Operateurs

- ▶ Les opérateurs arithmétiques
- ▶ Les opérateurs d'affectation
- ▶ Les opérateurs de comparaison
- ▶ Opérateurs **d'incrémantation** / décrémentation
- ▶ Les opérateurs logiques
- ▶ Opérateurs de chaînes
- ▶ Opérateurs de tableaux

/

<code>++\$x</code>	Pre-increment	Increments \$x by one, then returns \$x
<code>\$x++</code>	Post-increment	Returns \$x, then increments \$x by one
<code>--\$x</code>	Pre-decrement	Decrements \$x by one, then returns \$x
<code>\$x--</code>	Post-decrement	Returns \$x, then decrements \$x by one

Operateurs

- ▶ Les opérateurs arithmétiques
- ▶ Les opérateurs d'affectation
- ▶ Les opérateurs de comparaison
- ▶ Opérateurs d'incrémentation décrémentation
- ▶ Les opérateurs **logiques**
- ▶ Opérateurs de chaînes
- ▶ Opérateurs de tableaux

/

and	And	\$x and \$y
or	Or	\$x or \$y
xor	Xor	\$x xor \$y
&&	And	\$x && \$y
	Or	\$x \$y
!	Not	!\$x

Operateurs

- ▶ Les opérateurs arithmétiques
- ▶ Les opérateurs d'affectation
- ▶ Les opérateurs de comparaison
- ▶ Opérateurs d'incrémentation / décrémentation
- ▶ Les opérateurs logiques
- ▶ Opérateurs de **chaînes**
- ▶ Opérateurs de tableaux

.	Concatenation	\$txt1 . \$txt2
=	Concatenation assignment	\$txt1 .= \$txt2

Conditions

- ▶ if statement
- ▶ if...else statement
- ▶ if...elseif....else statement
- ▶ switch statement

```
<!DOCTYPE html>
<html><body>

<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
</body></html>
```

Conditions

► switch statement

```
<!DOCTYPE html>
<html>
<body>

<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>

</body>
</html>
```

PHP

Boucles

- ▶ while
- ▶ do...while
- ▶ for
- ▶ foreach

```
<!DOCTYPE html>
<html>
<body>
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
?>
</body>
</html>
```

PHP

Fonctions

- ▶ Une fonction est un bloc d'instructions qui peut être utilisé à plusieurs reprises dans un programme.
- ▶ Il n'est pas exécuté immédiatement quand une page se charge, mais une fois que la fonction est appelée.

```
<!DOCTYPE html>
<html><body>

<?php
function setHeight($minheight = 50) {
    echo 'The height is : $minheight <br>';
}

setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>
</body>
</html>
```

PHP

Fonctions

▶ function functionName()
 code to be executed;
}

```
<!DOCTYPE html>
<html><body>

<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>
</body></html>
```

Tableaux

- ▶ En PHP il y a 3 types de tableaux
 - ▶ **Les tableaux indexés**
 - ▶ Les tableaux associatifs
 - ▶ Les tableaux multi dimensions

```
▶ $cars = array("Volvo", "BMW", "Toyota");  
▶ echo count($cars);  
▶ for($x = 0; $x < $arrlength; $x++) {  
    echo $cars[$x];  
    echo "<br>";  
}
```

Tableaux

- ▶ En PHP il y a 3 types de tableaux
 - ▶ Les tableaux indexés
 - ▶ **Les tableaux associatifs**
 - ▶ Les tableaux multi dimensions

```
$age = array("Peter"=>"35", "Ben"=>"37",  
"Joe"=>"43");
```

```
foreach($age as $x => $x_value) {  
    echo "Key=" . $x . ", Value=" . $x_value;  
    echo "<br>";  
}
```

Tableau multiple

```
<?php  
echo $cars[0][0].": In stock: ".$cars[0][1].",  
sold: ".$cars[0][2].".<br>";  
echo $cars[1][0].": In stock: ".$cars[1][1].",  
sold: ".$cars[1][2].".<br>";  
echo $cars[2][0].": In stock: ".$cars[2][1].",  
sold: ".$cars[2][2].".<br>";  
echo $cars[3][0].": In stock: ".$cars[3][1].",  
sold: ".$cars[3][2].".<br>";  
?>
```

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

PHP

Tableaux

- ▶ sort() - sort arrays in ascending order
- ▶ rsort() - sort arrays in descending order
- ▶ asort() - sort associative arrays in ascending order, according to the value
- ▶ ksort() - sort associative arrays in ascending order, according to the key
- ▶ arsort() - sort associative arrays in descending order, according to the value
- ▶ krsort() - sort associative arrays in descending order, according to the key

```
<!DOCTYPE html>
<html><body>
<?php
$age = array("Peter"=>"35", "Ben"=>"37",
"Joe"=>"43");
asort($age);

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
</body></html>
```

Formulaires

- ▶ Code html <form>
- ▶ Get ou Post ?
- ▶ Validation et sécurité ?

```
<!DOCTYPE HTML>
<html>
<body>
<form action="myForm.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

Formulaires

- ▶ Changer Get par Post dans les deux fichiers !

```
<html>  
<body>
```

```
Welcome <?php echo $_GET["name"];  
?><br>
```

```
Your email address is: <?php echo  
$_GET["email"]; ?>
```

```
</body>  
</html>
```

PHP

Formulaires

► Validation et sécurité ?

[http://www.monsite.com/test_form.php/%22%3E%3Cscript%3Ealert\('hacked'\)%3C/script%3E](http://www.monsite.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E)

```
<!DOCTYPE HTML>
<html>
<body>
<form method="post" action=<?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>
</body>
</html>
```

Formulaires

- ▶ Validation et **sécurité** ?

```
<?php  
  
function test_input($data) {  
    $data = trim($data);  
    $data = stripslashes($data);  
    $data = htmlspecialchars($data);  
    return $data;  
}
```

Formulaires

► Validation et sécurité ?

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

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

PHP

mdp

```
password_hash ( string $password , int $algo [, array $options  
] ) : string
```

Dans votre base de données scinder en 2 tables le login et le mot de passe.

- ▶ PASSWORD_DEFAULT - Utilisation de l'algorithme bcrypt (par défaut depuis PHP 5.5.0). Notez que cette constante est concue pour changer dans le temps, au fur et à mesure que des algorithmes plus récents et plus forts sont ajoutés à PHP. Pour cette raison, la longueur du résultat issu de cet algorithme peut changer dans le temps, il est donc recommandé de stocker le résultat dans une colonne de la base de données qui peut contenir au moins 60 caractères (255 caractères peut être un très bon choix).
- ▶ PASSWORD_BCRYPT - Utilisation de l'algorithme CRYPT_BLOWFISH pour créer la clé de hachage. Ceci va créer une clé de hachage standard [crypt\(\)](#) utilisant l'identifiant "\$2y\$". Le résultat sera toujours une chaîne de 60 caractères, ou FALSE si une erreur survient.
- ▶ PASSWORD_ARGON2I - Utilise l'algorithme de hachage Argon2i pour créer le hachage. Cet algorithme est seulement disponible si PHP a été compilé avec le support d'Argon2
- ▶ PASSWORD_ARGON2ID - Utilise l'algorithme de hachage Argon2id pour créer le hachage. Cet algorithme est seulement disponible si PHP a été compilé avec le support d'Argon2

PHP 7.1 les évolutions du typage

```
function fono01(int $a) { var_dump($a); }

function fono01(?int $a) { var_dump($a); } // ?int indique que la valeur null est acceptée

function fono02(?int $a, ?int $b) : ?int
{
    if ($a === null || $b === null) { return null; }
    return $a + $b;
}

function returns_nothing() : void { // Cette fonction retourne null return; }

function fono03(iterable $data) { foreach ($data as $key => $val) {
    var_dump($val); // ... } } // fono03([10, 20, 30]);
```

PHP 8

À lire

<https://kinsta.com/fr/blog/php-8/#constructor-property-promotion>

<https://www.php.net/releases/8.0/fr.php>

The screenshot shows two side-by-side code snippets from the PHP documentation. On the left, under 'PHP 7', is the original constructor definition:

```
class Point {
    public float $x;
    public float $y;
    public float $z;

    public function __construct(
        float $x = 0.0,
        float $y = 0.0,
        float $z = 0.0
    ) {
        $this->x = $x;
        $this->y = $y;
        $this->z = $z;
    }
}
```

An arrow points from this code to the right, where it is shown how the code has changed in 'PHP 8' due to constructor property promotion:

```
class Point {
    public function __construct(
        public float $x = 0.0,
        public float $y = 0.0,
        public float $z = 0.0,
    ) {}
}
```

At the bottom left, the email address `bisgambiglia@univ-e` is visible. At the bottom right, the date and time `24/09/2025 07:07:26` are displayed.

PHP 8

Types d'union [RFC Doc](#)

PHP 7

```
class Number {
    /** @var int|float */
    private $number;

    /**
     * @param float|int $number
     */
    public function __construct($number) {
        $this->number = $number;
    }

    new Number('NaN'); // Ok
}
```

PHP 8

```
class Number {
    public function __construct(
        private int|float $number
    ) {}

    new Number('NaN'); // TypeError
}
```



PHP

BD

Connexion BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = new mysqli($servername,  
$username, $password);  
  
// Check connection  
if (mysqli_connect_error()) {  
    die("Database connection failed: " .  
    mysqli_connect_error());  
}  
?>
```

PHP

Connexion BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = mysqli_connect($servername,  
$username, $password);  
  
// Check connection  
if (!$conn) {  
    die("Connection failed: " .  
    mysqli_connect_error());  
}  
echo "Connected successfully";  
?>
```

PHP

Connexion BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new
        PDO("mysql:host=$servername;dbname=myDB",
        $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
        PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
}
catch(PDOException $e)
{
    echo "Connection failed: " . $e->getMessage();
?>
```

Sélection BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername,  
$username, $password, $dbname);  
// Check connection  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn-  
>connect_error);  
}  
  
...
```

Sélection BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

...
```

Sélection BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new
    PDO("mysql:host=$servername;dbname=$dbname",
    $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
    PDO::ERRMODE_EXCEPTION);
    $sql = "INSERT INTO MyGuests (firstname, lastname,
    email)
    VALUES ('John', 'Doe', 'john@example.com')";
    // use exec() because no results are returned
    $conn->exec($sql);
    echo "New record created successfully";
}
catch(PDOException $e)
```

Création BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username,
$password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}
$conn->close();
?>
```

PHP

Création BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username,
$password);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP

Création BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB",
    $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
    PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE myDBPDO";
    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Database created successfully<br>";
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

Création table BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
// Create connection
// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY
KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
```

PHP

Création table BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
// Create connection
// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY
KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " .
    mysqli_error($conn);
}

mysqli_close($conn);
```

PHP

Création table BD

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
try {  
    //connexion  
  
    // sql to create table  
    $sql = "CREATE TABLE MyGuests (  
        id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
        firstname VARCHAR(30) NOT NULL,  
        lastname VARCHAR(30) NOT NULL,  
        email VARCHAR(50),  
        reg_date TIMESTAMP  
    )";  
  
    // use exec() because no results are returned  
    $conn->exec($sql);  
    echo "Table MyGuests created successfully";  
}  
catch(PDOException $e)  
{  
    echo $sql . "<br>" . $e->getMessage();  
}  
  
$conn = null;
```

Ajout de données

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
// Create connection

$sql = "INSERT INTO MyGuests (firstname, lastname,
email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
```

PHP

Ajout de données

- ▶ MySQLi (object-oriented)
- ▶ [MySQLi \(procedural\)](#)
- ▶ PDO

```
// Create connection

$sql = "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('John', 'Doe',
'john@example.com')";

if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" .
    mysqli_error($conn);
}

mysqli_close($conn);
```

Ajout de données

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
try {  
    //connexion  
  
    $sql = "INSERT INTO MyGuests (firstname,  
        lastname, email)  
        VALUES ('John', 'Doe', 'john@example.com');  
        // use exec() because no results are returned  
        $conn->exec($sql);  
        echo "New record created successfully";  
    }  
catch(PDOException $e)  
{  
    echo $sql . "<br>" . $e->getMessage();  
}  
  
$conn = null;
```

PHP

Ajout multiple de données

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}
```

```
$sql = "INSERT INTO MyGuests (firstname, lastname,
email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
```

Sélection de données

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
// Create connection

$conn = new mysqli("localhost", "root", "", "test");

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"] . " - Name: " .
        $row["firstname"] . " " . $row["lastname"] . "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
```

Sélection de données

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
// connexion
$sql = "SELECT id, firstname, lastname FROM
MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: ".
$row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
```

Sélection de données

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
try {  
    $conn = new  
    PDO("mysql:host=$servername;dbname=$dbname",  
        $username, $password);  
    $conn->setAttribute(PDO::ATTR_ERRMODE,  
        PDO::ERRMODE_EXCEPTION);  
    $stmt = $conn->prepare("SELECT id, firstname, lastname  
    FROM MyGuests");  
    $stmt->execute();  
  
    // set the resulting array to associative  
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);  
    foreach(new TableRows(new  
        RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {  
        echo $v;  
    }  
}  
catch(PDOException $e) {  
    echo "Error: " . $e->getMessage();  
}  
$conn = null;
```

Mise à jour de données

- ▶ **MySQLi (object-oriented)**
- ▶ MySQLi (procedural)
- ▶ PDO

```
// Create connection  
...  
  
$sql = "UPDATE MyGuests SET  
lastname='Doe' WHERE id=2";  
  
if ($conn->query($sql) === TRUE) {  
    echo "Record updated successfully";  
} else {  
    echo "Error updating record: " . $conn-  
>error;  
}
```

Mise à jour de données

- ▶ MySQLi (object-oriented)
- ▶ **MySQLi (procedural)**
- ▶ PDO

```
// Create connection  
...  
  
$sql = "UPDATE MyGuests SET  
lastname='Doe' WHERE id=2";  
  
if (mysqli_query($conn, $sql)) {  
    echo "Record updated successfully";  
} else {  
    echo "Error updating record: " .  
    mysqli_error($conn);  
}
```

Mise à jour de données

- ▶ MySQLi (object-oriented)
- ▶ MySQLi (procedural)
- ▶ PDO

```
...  
try {  
    $conn = new  
    PDO("mysql:host=$servername;dbname=$dbname",  
        $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR_ERRMODE,  
        PDO::ERRMODE_EXCEPTION);  
  
    $sql = "UPDATE MyGuests SET lastname='Doe' WHERE  
        id=2";  
  
    // Prepare statement  
    $stmt = $conn->prepare($sql);  
  
    // execute the query  
    $stmt->execute();  
  
    // echo a message to say the UPDATE succeeded  
    echo $stmt->rowCount() . " records UPDATED  
        successfully";  
}
```

PHP

Prepared Statements

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDB";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password,  
$dbname);  
  
// Check connection  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn->connect_error);  
}
```

```
// prepare and bind  
$stmt = $conn->prepare("INSERT INTO  
MyGuests (firstname, lastname, email)  
VALUES (?, ?, ?)");  
  
$stmt->bind_param("sss", $firstname,  
$lastname, $email);  
  
// set parameters and execute  
$firstname = "John";  
$lastname = "Doe";  
$email = "john@example.com";  
$stmt->execute();
```

PHP

- ▶ PDO (PHP Data Objects) : interface parue avec PHP 5 qui vise à utiliser des bases de données sans avoir à s'occuper du SGBD utilisé derrière. Ainsi, il est tout à fait possible de faire un code qui marchera avec MySQL mais aussi avec Oracle, ODBC, etc.
- ▶ **Avantages de PDO :**
 - ▶ interface pour SGBD : plus besoin de s'occuper de savoir quelle SGBD est derrière (en théorie) ;
 - ▶ orienté objet : les objets PDO et PDOStatement peuvent être étendus, il est donc tout à fait possible de personnaliser et remodeler une partie du comportement initial ;
 - ▶ exception : les objets de l'interface PDO utilisent des exceptions, il est donc tout à fait possible d'intégrer facilement un système de gestion des erreurs.

PHP

Activation :

- ▶ Avec PHP 5, cette extension n'est pas activée par défaut. Afin de remédier au problème, ouvrez le 'php.ini' que vous utilisez pour PHP 5 puis rendez-vous dans la partie où vous avez la liste des extensions. Faites une recherche sur 'Windows Extensions', vous devriez y arriver directement. Bien : maintenant, vérifiez que vous avez la ligne suivante :
 - ▶ ;extension=php_pdo.dll
- ▶ Si c'est le cas, décommentez la ligne afin d'avoir :
 - ▶ extension=php_pdo.dll
- ▶ Sinon, rajoutez la ligne à la main.
- ▶ Vous venez donc d'activer PDO mais cela n'est pas suffisant : maintenant, il vous faut activer le driver correspondant au type de BDD que vous souhaitez utiliser.
- ▶ Pour cela rien de plus simple : imaginons que vous vouliez activer MySQL, rajoutez pour cela la ligne suivante (ou activez-la si elle est présente) :
 - ▶ extension=php_pdo_mysql.dll
- ▶ Bien, enregistrez, relancez le serveur et vous voilà prêts à travailler avec l'extension PDO qui utilise MySQL

PHP

► Création d'une connexion :

```
<?php  
$PARAM_hote='localhost'; // le chemin vers le serveur  
$PARAM_port='3306';  
$PARAM_nom_bd='sdz'; // le nom de votre base de données  
$PARAM_utilisateur='root'; // nom d'utilisateur pour se connecter  
$PARAM_mot_passe=""; // mot de passe de l'utilisateur pour se connecter  
$connexion = new PDO('mysql:host='.$PARAM_hote.';port='.$PARAM_port.';dbname='.$PARAM_nom_bd  
, $PARAM_utilisateur, $PARAM_mot_passe);  
?>
```

PHP

```

<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // 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();

    // insert another row
    $firstname = "Mary";
    $lastname = "Moe";
    $email = "mary@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Julie";
    $lastname = "Dooley";
    $email = "julie@example.com";
    $stmt->execute();

    echo "New records created successfully";
}
catch(PDOException $e)
{
    echo "Error: " . $e->getMessage();
}

$conn = null;
?>

```



PHP

BD NOSQL

NoSQL

- ▶ <http://www.stephane-raymond.com/blog/nosql/debuter-avec-cassandra-et-php/>
- ▶ <http://www.lafermeduweb.net/billet/nosql-mongodb-et-php-premiere-approche-781.html>
- ▶ <http://www.w3resource.com/mongodb/nosql.php>



PHP

SESSION

Session

Définition

- ▶ Les sessions sont un moyen simple de stocker des données individuelles pour chaque utilisateur.
- ▶ 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

```
<?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>
```

Session

Fonctionnement

- ▶ Lorsqu'une session est démarrée, PHP va soit récupérer une session existante en utilisant l'identifiant de session passé (cookie de session) ou si aucun identifiant de session n'est passé, il va créer une nouvelle session.
- ▶ la variable superglobale `$_SESSION` est initialisée avec toutes les données de session

```
<?php  
session_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
print_r($_SESSION);  
?>  
  
</body>  
</html>
```

Session

Fonctionnement

- ▶ Les données de session sont sauvegardées sur le serveur à l'endroit spécifié par la directive de configuration `session.save_path`.
- ▶ Les sessions peuvent être démarrées manuellement en utilisant la fonction `session_start()`. Si la directive de configuration `session.auto_start` est définie à 1, une session démarra 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()`.

```
<?php  
session_start();  
?>  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
// Echo session variables that were set on previous  
page  
echo "Favorite color is " . $_SESSION["favcolor"] .  
".<br>";  
echo "Favorite animal is " . $_SESSION["favanimal"]  
".  
?>  
</body>  
</html>
```

Session

Exemple

// Enregister une variable

```
<?php
session_start();
if (!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
} else {
    $_SESSION['count']++;
}
?>
```

// Supprimer

```
<?php
session_start();
unset($_SESSION['count']);
?>
```

Cookie

- ▶ Création
 - ▶ setcookie
- ▶ Stockage
 - ▶ \$_COOKIE[]
- ▶ Destruction
 - ▶ setcookie("user", "", time() - 3600);

```
<?php  
$cookie_name = "user";  
$cookie_value = "John Doe";  
setcookie($cookie_name, $cookie_value, time() + (86400  
* 30), "/"); // 86400 = 1 day  
?>  
<html>  
<body>  
  
<?php  
if(!isset($_COOKIE[$cookie_name])) {  
    echo "Cookie named ".$cookie_name . " is not set!";  
} else {  
    echo "Cookie " . $cookie_name . " is set!<br>";  
    echo "Value is: " . $_COOKIE[$cookie_name];  
}  
?>  
</body>  
</html>
```

Mail

Exemple 1

- ▶ Pour envoyer un mail
 - ▶ `mail(to,subject,message,headers,parameters);`
- ▶ Il faut configurer le fichier `php.ini` et renseigner le serveur SMTP

```
<?php
// the message
$msg = "First line of text\nSecond line of
text";

// use wordwrap() if lines are longer than
70 characters
$msg = wordwrap($msg,70);

// send email
mail("someone@example.com","My
subject",$msg);
?>
```

Mail

Exemple 2

- ▶ Pour envoyer un mail
 - ▶ `mail(to,subject,message,headers,parameters);`
- ▶ Il faut configurer le fichier `php.ini` et renseigner le serveur SMTP

```
<?php  
$to = "somebody@example.com";  
$subject = "My subject";  
$txt = "Hello world!";  
$headers = "From:  
webmaster@example.com" . "\r\n" .  
"CC: somebodyelse@example.com";  
  
mail($to,$subject,$txt,$headers);  
?>
```

PHP

XML
EXTENSIBLE MARKUP
LANGUAGE

PHP et XML

XML

- ▶ langage permettant de mettre en forme des documents/données grâce à des balises
- ▶ La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité
- ▶ Il permet de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

```
<annuaire>
<personne class = "etudiant">
<nom>Pillou</nom>
<prenom>Jean-Francois</prenom>
<telephone>555-123456</telephone>
<email>webmaster@site.fr</email>
</personne>
<personne>
...
</personne>
</annuaire>
```

PHP et XML

XML

- ▶ les balises XML décrivent le contenu plutôt que la présentation (contrairement à HTML)
- ▶ XML permet de séparer le contenu de la présentation
- ▶ XML fournit des moyens de vérifier la syntaxe d'un document grâce aux DTD (Document Type Definition)

```
<?php
require "connect.php";
$Fnm = "mon_dossier/mon_fichier.xml";
$reponse = mysql_query("SELECT * FROM voiture ") or
    die(mysql_error()); // Requête SQL
$xml = '<?xml version="1.0" encoding="ISO-8859-1"?>';
$xml .= '<articles>';
while( $row = mysql_fetch_assoc($reponse) )
{
    $xml .= '<article id="" . $row['idd'] . ">';
    $xml .= '<titre>' . htmlspecialchars($row['marque']) .
    '</titre>';
    $xml .= '<contenu>' . htmlspecialchars($row['modele']) .
    '</contenu>';
    $xml .= '</article>';
}
$xml .= '</articles>';
$inF = fopen($Fnm,"w");
fwrite($inF,$xml);
fclose($inF);
?>
```

PHP et XML

XML

- ▶ XML permet de définir un format d'échange et des mécanismes pour vérifier la validité du document produit.
- ▶ Un parseur permet d'extraire les données d'un document et de vérifier éventuellement la validité du document.

```
test.xml
<?xml version="1.0" encoding="ISO-8859-1" ?>

<continents>
    <europe>
        <pays>France</pays>
        <pays>Belgique</pays>
        <pays>Espagne</pays>
    </europe>
    <asie>
        <pays>Japon</pays>
        <pays>Inde</pays>
    </asie>
</continents>
```

PHP et XML

XML

► Format

- Lisible
- Auto descriptif et extensible
- structure arborescente
- Portable

```
<?php  
$dom = new DomDocument;  
$dom->load("test.xml");  
$listePays = $dom->getElementsByTagName('pays');  
foreach($listePays as $pays)  
    echo $pays->firstChild->nodeValue . "<br />";  
    echo "---<br />";  
$europe = $dom->getElementsByTagName('europe')->item(0);  
$listePaysEurope = $europe->getElementsByTagName('pays');  
foreach($listePaysEurope as $pays)  
    echo $pays->firstChild->nodeValue . "<br />";  
?>
```

PHP et XML

Parser XML

- ▶ Pour lire, mettre à jour, créer et manipuler un document XML, on utilise un parseur XML.
- ▶ En PHP, il y a deux types de parseurs XML:
 - ▶ Tree-Based Parsers
 - ▶ Event-Based Parsers

Les Tree-Based Parsers (SimpleXML, DOM) chargent l'intégralité du document en mémoire et transforme le document XML en structure arborescente. Il analyse l'ensemble du document, et donne accès aux éléments de l'arbre (DOM). Ce type d'analyseur est une meilleure option pour les documents XML de petites taille.

Les Event-Based Parsers (XMLReader) ne conservent pas la totalité du document en mémoire, ils lisent dans un seul nœud à la fois. Adapté pour les gros documents XML. Il analyse plus rapide et consomme moins de mémoire.

PHP et XML

Parser XML

- ▶ Exemple pour lire de l'XML avec la fonction **simplexml_load_string()**
- ▶ SimpleXMLElement Object ([to] => Pierre [from] => Richard [heading] => Reminder [body] => Don't forget me this weekend!)

```
<!DOCTYPE html>
<html>
<body>
<?php
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Pierre</to>
<from>Richard</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>";

$xml=simplexml_load_string($myXMLData)
die("Error: Cannot create object");
print_r($xml);
?>
</body>
</html>
```

or

PHP et XML

Parser XML

- ▶ Exemple pour lire de l'XML avec la fonction **simplexml_load_string()**
- ▶ Avec fichier externe
- ▶ SimpleXMLElement Object ([to] => Pierre [from] => Richard [heading] => Reminder [body] => Don't forget me this weekend!)

```
<!DOCTYPE html>
<html>
<body>
<?php
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Pierre</to>
<from>Richard</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>";

$xml=simplexml_load_file("note.xml") or die("Error:
Cannot create object");
print_r($xml);
?>
</body></html>
```

PHP et XML

Parser XML

- ▶ Gestion des erreurs avec ***libxml***
- ▶ Failed loading XML:
Opening and ending tag mismatch:
user line 3 and wronguser
Opening and ending tag mismatch:
email line 4 and wrongemail

```
<?php
libxml_use_internal_errors(true);
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<document>
<user>John Doe</wronguser>
<email>john@example.com</wrongemail>
</document>";

$xml = simplexml_load_string($myXMLData);
if ($xml === false) {
    echo "Failed loading XML: ";
    foreach(libxml_get_errors() as $error) {
        echo "<br>", $error->message;
    }
} else {
    print_r($xml);
}
?>
```

PHP et XML

Parser XML

- ▶ Lire dans l'arbre
- ▶ Avec plusieurs éléments
 - ▶ echo \$xml->body[i]->souselement;

```
<?php  
$xml=simplexml_load_file("note.xml")    or  
die("Error: Cannot create object");  
echo $xml->to . "<br>";  
echo $xml->from . "<br>";  
echo $xml->heading . "<br>";  
echo $xml->body;  
?>
```

```
<?xml version="1.0" encoding="utf-8"?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en-us">XQuery Kick Start</title>
    <author>James McGovern</author>
    <year>2003</year>
    <price>49.99</price>
  </book>
  <book category="WEB">
    <title lang="en-us">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

```
<?php
$xml=simplexml_load_file("books.xml") or die("Error:
Cannot create object");
foreach($xml->children() as $books) {
  echo $books->title . ", ";
  echo $books->author . ", ";
  echo $books->year . ", ";
  echo $books->price . "<br>";
}
?>
```

PHP et XML

Parser DOM

► XML

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

print $xmlDoc->saveXML();
?>

<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item) {
    print $item->nodeName . " = " . $item-
        >nodeValue . "<br>";
}
?>
```

PHP

AJAX
ASYNCHRONOUS
JAVASCRIPT AND XML

AJAX

AJAX

- ▶ XMLHttpRequest object (to exchange data asynchronously with a server)
- ▶ JavaScript/DOM (to display/interact with the information)
- ▶ CSS (to style the data)
- ▶ XML (often used as the format for transferring data)

Méthodes de l'objet XMLHttpRequest	
Méthode	Description
open()	Met fin à la requête en cours et en prépare une nouvelle, en indiquant la méthode (GET ou POST) et l'URL.
send()	Lance la requête
abort()	Met fin à la requête en cours
setRequestHeader()	Assigne un couple nom/valeur à l'en-tête accompagnant la requête
getResponseHeader()	Récupère la valeur d'une chaîne de l'en-tête de réponse
getAllResponseHeader()	Récupère l'ensemble des en-têtes de réponse

```

<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
$a[] = "Ophelia";
$a[] = "Petunia";
$a[] = "Amanda";
$a[] = "Raquel";
$a[] = "Cindy";
$a[] = "Doris";
$a[] = "Eve";
$a[] = "Evita";
$a[] = "Sunniva";
$a[] = "Tove";
$a[] = "Unni";
$a[] = "Violet";
$a[] = "Liza";
$a[] = "Elizabeth";
$a[] = "Ellen";

```

```

        $a[] = "Wenche";
        $a[] = "Vicky";
        // get the q parameter from URL
        $q = $_REQUEST["q"];
        $hint = "";
        // lookup all hints from array if $q is different from ""
        if ($q !== "") {
            $q = strtolower($q);
            $len=strlen($q);
            foreach($a as $name) {
                if (stristr($q, substr($name, 0, $len))) {
                    if ($hint === "") {
                        $hint = $name;
                    } else {
                        $hint .= ", $name";
                    }
                }
            }
        }
        // Output "no suggestion" if no hint was found or
        // output correct values
        echo $hint === "" ? "no suggestion" : $hint;
    ?>

```

src : http://www.w3schools.com/php/php_ajax_php.asp

```
<html>
<head>
<script>
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML
= "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status
== 200) {
                document.getElementById("txtHint").inner
HTML = xmlhttp.responseText;
            }
        };
        xmlhttp.open("GET", "gethint.php?q=" + str,
true);
        xmlhttp.send();
    }
}
</script>
</head>
<body>
<p><b>Start typing a name in the input field
below:</b></p>
<form>
First name: <input type="text"
onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

PHP

BONNES PRATIQUES

Les commandements du développement PHP

- ▶ echo est plus rapide que print
- ▶ Mettre ses chaînes de caractères entre simple quotes '...' est plus rapide qu'entre des doubles quotes "..." car PHP analyse s'il y'a des variables entre les doubles quotes. Utiliser les simple quote pour du texte pur.
- ▶ Utiliser sprintf au lieu de mettre des variables dans des double quotes, C'est 10x plus rapide
- ▶ Utiliser les paramètres multiples dans un echo au lieu de la concaténation des chaînes.
- ▶ Utiliser le plus possible des variables pour les calculs, éviter de les mettre dans les boucles
- ▶ Pensez à unset ou rendre null vos variables, en particulier les gros tableaux

Les commandements du développement PHP

- ▶ Eviter les méthodes magiques comme __get, __set, __autoload
- ▶ Utiliser require() au lieu de require_once() quand c'est possible
- ▶ Utilisez des chemins complets dans vos include et require. C'est du temps gagné pour la résolution du chemin au niveau de votre OS
- ▶ require() et include() sont identiques à part que require arrete le script si le fichier n'est pas trouvé. Les performances sont quasi identiques
- ▶ Depuis PHP5, l'heure de démarrage d'un script peut être trouvé grâce à \$_SERVER['REQUEST_TIME'], à utiliser à la place de time() ou microtime()

Les commandements du développement PHP

- ▶ PCRE regex est plus rapide que EREG, mais il faut toujours regarder s'il n'est pas possible d'utiliser une fonction native comme strncasecmp, strpbrk et stripas à la place
- ▶ Quand vous parsez du XML en PHP essayez xml2array, qui permet d'utiliser les fonctions PHP XML, pour du HTML vous pouvez essayer DOM document
- ▶ La suppression d'erreurs avec @ est très lent
- ▶ str_replace est plus rapide que preg_replace, str_replace est globalement le meilleur dans tous les cas, même si quelques fois strstr est plus rapide avec des chaînes longues. Utiliser un array() dans str_replace est plus rapide que d'utiliser plusieurs str_replace
- ▶ "else if" est plus rapide qu'un case/switch

Les commandements du développement PHP

- ▶ Fermer les connexions aux BDD après les avoir utilisé
- ▶ L'utilisation d'un code strict permettant de supprimer toutes les erreurs, warning etc est conseillé. `error_reporting(E_ALL)` doit toujours être activé
- ▶ Les scripts PHP sont rendus 2 à 10 fois moins rapidement par Apache qu'une page statique. Essayez d'utiliser au maximum des pages statiques
- ▶ `$row['id']` est 7 fois plus rapide que `$row[id]`, car si vous ne mettez pas les quotes, PHP Pense qu'il va s'agir d'une constante
- ▶ Les scripts PHP sont compilés à la volée (si pas de cache). Installez un système de cache PHP (comme memcached, eAccelerator ou Turck MMCache) permet d'augmenter de 25-100% les performances

Les commandements du développement PHP

- ▶ Utilisez `isset` où c'est possible au lieu de `strlen`. (ie: `if (strlen($foo) < 5) { echo "Foo is too short"; } vs. if (!isset($foo{5})) { echo "Foo is too short"; }`)
- ▶ `++$i` est plus rapide que `$ i++`, donc utilisez le pre-increment quand c'est possible
- ▶ Analysez votre code (Profiler). Utilisez **Xdebug** debugger pour profiler du code PHP
- ▶ Documentez votre code.
- ▶ <http://www.sitepoint.com/good-and-bad-php-code/>
- ▶ Séparez les couches: Contenu, PHP et HTML. HTML dans un autre fichier que le PHP
- ▶ Ne jamais avoir confiance en les variables utilisateurs: `$_POST` et `$_GET`. Utilisez **mysql_real_escape_string** quand vous utilisez MySQL, et **htmlspecialchars** quand vous rendez du HTM

Les commandements du développement PHP

- ▶ Pour des raisons de sécurité, ne dévoillez jamais d'infos concernant vos paths, extensions et configuration, comme utiliser `display_errors` ou `phpinfo()`
- ▶ Ne jamais utiliser du texte clair pour stocker les mots de passe ou les comparer. Utilisez un hash md5 au minimum
- ▶ Utilisez `ip2long()` et `long2ip()` pour stocker les adresses IP en INT plutôt qu'en STRING
- ▶ Quand vous utilisez `header('Location: '.$url);` n'oubliez pas d'y faire suivre un `die();` car le script continue de tourner même après l'instruction
- ▶ Avec POO, si une méthode peut être static, alors déclarez la en static. Elle sera 4 fois plus rapide
- ▶ Incrémenter une variable locale dans une méthode POO est le plus rapide
- ▶ Incrémenter une propriété d'un objet (eg. `$this->prop++`) est 3 fois plus lent qu'une variable locale

Les commandements du développement PHP

- ▶ incrémenter une variable indéfinie est 9-10 fois plus lent qu'une variable pré définie
- ▶ Déclarer une variable globale dans une fonction sans l'utiliser ralenti les choses. PHP doit vérifier qu'elle existe bien
- ▶ Le nombre de méthodes dans une classe ne change rien aux performances d'appel d'une méthode
- ▶ Les méthodes d'une classe dérivée vont plus vite que celles de la classe mère
- ▶ Tout ne doit pas être objet, chaque méthode et propriété consomme de la mémoire
- ▶ Echappez les chaînes provenant de l'extérieur avec `mysql_real_escape_string`, au lieu de `mysql_escape_string` ou `addslashes`. Si `magic_quotes_gpc` est activé, mieux vaut utiliser `stripslashes` en premier

Les commandements du développement PHP

- ▶ Attention lors de l'utilisation de mail() et de ses headers, il y'a des failles de sécurité
 - ▶ Il faut vider (unset) les variables dont on ne se sert plus après s'être connecté à la BDD
 - ▶ Utiliser les tags <?php et ?>
 - ▶ Spécifier l'encodage dans la balise meta des pages HTML (ex: <meta charset= "utf-8" />)
 - ▶ Content-type: text/html; charset=utf-8
-
- ▶ <http://blog.netapsys.fr/audit-php-securite-et-bonnes-pratiques/>
 - ▶ <http://gregwar.com/php/practices.html>