PHP

DOCUMENTATION DE PHP : [DOC](http://www.php.net/manual/fr/funcref.php)

**Table des matières**

[Premiers pas avec PHP 8](#_Toc468093916)

[Forme d’une balise PHP : 8](#_Toc468093917)

[Echo : 8](#_Toc468093918)

[Les commentaires : 9](#_Toc468093919)

[Inclure des portions de page 9](#_Toc468093920)

[Les variables 10](#_Toc468093921)

[Les types de variables : 10](#_Toc468093922)

[Déclaration de variable : 10](#_Toc468093923)

[Afficher et concaténer des variables : 10](#_Toc468093924)

[Les opérations : 11](#_Toc468093925)

[Les conditions 12](#_Toc468093926)

[If … else : 12](#_Toc468093927)

[Conditions multiples : 13](#_Toc468093928)

[Le switch-case : 14](#_Toc468093929)

[Les ternaires : 15](#_Toc468093930)

[Les boucles 15](#_Toc468093931)

[Le while : 15](#_Toc468093932)

[Le for : 15](#_Toc468093933)

[Les fonctions 16](#_Toc468093934)

[Quelques fonctions utiles : 16](#_Toc468093935)

[Strlen() : 16](#_Toc468093936)

[Str\_replace() : 16](#_Toc468093937)

[Str\_shuffle() : 16](#_Toc468093938)

[Strlower() : 17](#_Toc468093939)

[Récupérer les dates : 17](#_Toc468093940)

[Créer ses propres fonctions : 18](#_Toc468093941)

[Les tableaux 18](#_Toc468093942)

[Les tableaux numérotés : 18](#_Toc468093943)

[Récupérer une valeur : 19](#_Toc468093944)

[Le tableau associatif : 19](#_Toc468093945)

[Récupérer une valeur : 20](#_Toc468093946)

[Parcourir tout le tableau : for, foreach, print\_r : 20](#_Toc468093947)

[La boucle for : 20](#_Toc468093948)

[Foreach : 20](#_Toc468093949)

[Print\_r : 21](#_Toc468093950)

[Recherche dans un tableau : 21](#_Toc468093951)

[Vérifier que la clé existe dans l’array () : 21](#_Toc468093952)

[Vérifier qu’une valeur existe dans array() : 22](#_Toc468093953)

[Récupérer la clé d’une valeur : 22](#_Toc468093954)

[Transmettre des données dans l’URL 23](#_Toc468093955)

[Envoyer des données : 23](#_Toc468093956)

[Récupérer les données : 23](#_Toc468093957)

[Contrôler les valeurs des paramètres : 24](#_Toc468093958)

[Transmettre des données avec les formulaires 25](#_Toc468093959)

[Créer la base du formulaire : 25](#_Toc468093960)

[Petites zones de texte : 25](#_Toc468093961)

[Les grandes zones de texte : 26](#_Toc468093962)

[Les listes déroulantes : 26](#_Toc468093963)

[Les checkbox : 26](#_Toc468093964)

[Les champs cachés : 27](#_Toc468093965)

[La faille XSS : 27](#_Toc468093966)

[L’envoi des fichiers : 27](#_Toc468093967)

[Variable superglobale, session et cookie 29](#_Toc468093968)

[Les variables superglobales : 29](#_Toc468093969)

[Les sessions : 30](#_Toc468093970)

[Les cookies : 30](#_Toc468093971)

[Afficher un cookie : 31](#_Toc468093972)

[Modifier un cookie : 31](#_Toc468093973)

[Lire et écrire dans un fichier 32](#_Toc468093974)

[Autoriser l’écriture de fichier : 32](#_Toc468093975)

[Ouvrir et fermer un fichier : 32](#_Toc468093976)

[Lire et écrire dans un fichier : 33](#_Toc468093977)

[Lire un fichier : 33](#_Toc468093978)

[Ecrire dans un fichier : 33](#_Toc468093979)

[Les bases de données 34](#_Toc468093980)

[PhpMyAdmin 34](#_Toc468093981)

[Créer une table : 34](#_Toc468093982)

[Modifier une table : 35](#_Toc468093983)

[Lire des données SQL 35](#_Toc468093984)

[Se connecter à la base de données en PHP : 35](#_Toc468093985)

[Se connecter à mysql avec PDO : 35](#_Toc468093986)

[Tester la présence d’erreur : 36](#_Toc468093987)

[Récupérer les données : 36](#_Toc468093988)

[Filtrer les champs : 38](#_Toc468093989)

[Where : 39](#_Toc468093990)

[ORDER BY : 39](#_Toc468093991)

[LIMIT : 40](#_Toc468093992)

[Construire des requêtes en fonction de variables : 40](#_Toc468093993)

[Avec des marqueurs « ? » : 41](#_Toc468093994)

[Traquer les erreurs : 42](#_Toc468093995)

[Ecrire des données 42](#_Toc468093996)

[INSERT : ajoute des données : 42](#_Toc468093997)

[Insertion avec des requêtes séparées : 43](#_Toc468093998)

[Update pour modifier des valeurs : 43](#_Toc468093999)

[En requête séparée : 43](#_Toc468094000)

[DELETE : supprimer des données 44](#_Toc468094001)

[Les fonction SQL 44](#_Toc468094002)

[La fonction scalaire : 44](#_Toc468094003)

[Quelques fonctions utiles : 45](#_Toc468094004)

[Les fonctions d’agrégats : 45](#_Toc468094005)

[Quelques fonctions utiles : 45](#_Toc468094006)

[Compter uniquement le nombre de champ non nul : 46](#_Toc468094007)

[Distinct : 46](#_Toc468094008)

[Group by et having : 46](#_Toc468094009)

[Group by : 46](#_Toc468094010)

[HAVING : 46](#_Toc468094011)

[Les dates en SQL 47](#_Toc468094012)

[Les champs de type date : 47](#_Toc468094013)

[Utilisation : 47](#_Toc468094014)

[Fonctions de gestion de date : 48](#_Toc468094015)

[La date actuelle : 48](#_Toc468094016)

[DATE\_ADD et DATE\_SUBB : 48](#_Toc468094017)

[TP blog 49](#_Toc468094018)

[Les jointures entre table 51](#_Toc468094019)

[Modéliser une relation : 51](#_Toc468094020)

[Qu’est-ce qu’une jointure ? 51](#_Toc468094021)

[Les jointures internes : 52](#_Toc468094022)

[Jointures externes : 53](#_Toc468094023)

[Créer des images en php 54](#_Toc468094024)

[Activer la bibliothèque GD : 54](#_Toc468094025)

[Bases de la création d’image : 54](#_Toc468094026)

[Texte et couleurs : 55](#_Toc468094027)

[Ecrire du texte : 56](#_Toc468094028)

[Dessiner une forme : 56](#_Toc468094029)

[ImageSetPixel : 56](#_Toc468094030)

[ImageLine : 57](#_Toc468094031)

[ImageEllipse : 57](#_Toc468094032)

[ImageRectange : 57](#_Toc468094033)

[ImgePolygon 57](#_Toc468094034)

[Fonctions plus puissantes : 57](#_Toc468094035)

[Rendre une image transparente : 57](#_Toc468094036)

[Mélanger deux images : 58](#_Toc468094037)

[Redimensionner une image : 59](#_Toc468094038)

[Les expressions régulières 60](#_Toc468094039)

[Où utiliser une REGEX ? 60](#_Toc468094040)

[Grep\_match : 60](#_Toc468094041)

[Des recherches simples : 61](#_Toc468094042)

[La casse : 61](#_Toc468094043)

[Le OU : 61](#_Toc468094044)

[Début et fin de chaîne : 62](#_Toc468094045)

[Les classes de caractères : 62](#_Toc468094046)

[Des classes simples : 62](#_Toc468094047)

[Les intervalles de classes : 62](#_Toc468094048)

[Les quantificateurs : 63](#_Toc468094049)

[Les plus courants : 63](#_Toc468094050)

[Plus de précision : les accolades : 64](#_Toc468094051)

[Les métacaractères : 64](#_Toc468094052)

[Les classes abrégées : 65](#_Toc468094053)

[Construire une REGEX complète : 66](#_Toc468094054)

[Capture et remplacement : 67](#_Toc468094055)

[Les parenthèses captures : 67](#_Toc468094056)

[La POO 70](#_Toc468094057)

[Créer une classe : 70](#_Toc468094058)

[Getter et Setters : 71](#_Toc468094059)

[Autres fonctions : 71](#_Toc468094060)

[Constructeurs et destructeurs : 72](#_Toc468094061)

[Les fonctions magiques : 72](#_Toc468094062)

[Les autres fonctions magiques : 73](#_Toc468094063)

[L’héritage : 73](#_Toc468094064)

[Droits d’accès et encapsulation : 73](#_Toc468094065)

[Le droit d’accès : 73](#_Toc468094066)

[Organiser son code selon l’architecture MVC 74](#_Toc468094067)

[MVC : 74](#_Toc468094068)

[Code du TP blog et ses défauts : 75](#_Toc468094069)

[Amélioration : 76](#_Toc468094070)

[Le modèle : 76](#_Toc468094071)

[Le contrôleur : 77](#_Toc468094072)

[La vue : 77](#_Toc468094073)

[Le contrôleur global du site 78](#_Toc468094074)

[Résumé des fichiers : 78](#_Toc468094075)

[TP créer espace membre 79](#_Toc468094076)

[Les pages principales : 79](#_Toc468094077)

[Mettre son site en ligne 80](#_Toc468094078)

[Coder proprement 81](#_Toc468094079)

[Créer un dossier d’admin du site 81](#_Toc468094080)

[Créer le dossier htaccess : 81](#_Toc468094081)

[Créer le .htpasswd : 82](#_Toc468094082)

# Premiers pas avec PHP

## Forme d’une balise PHP :

On insère une balise PHP dans du code html

<!DOCTYPE html>

<html>

<Head>

<title>Ceci est une page de test avec des balises PHP</title>

<meta charset="utf-8" />

</head>

<body>

<h2>Page de test</h2>

<p>

Cette page contient du code HTML avec des balises PHP.<br />

<?php /\* Insérer du code PHP ici \*/ ?>

Voici quelques petits tests :

</p>

<ul>

<li style="color: blue;">Texte en bleu</li>

<li style="color: red;">Texte en rouge</li>

<li style="color: green;">Texte en vert</li>

</ul>

<?php

/\* Encore du PHP

Toujours du PHP \*/

?>

</body>

</html>

On peut placer le code PHP n’importe où, même dans l’entête :

<!DOCTYPE html>

<html>

<head>

<title>Ceci est une page de test <?php /\* Code PHP \*/ ?></title>

<meta charset="utf-8" />

</head>

<!DOCTYPE html>

<html>

<head>

<title>Ceci est une page de test</title>

<meta <?php /\* Code PHP \*/ ?> charset="utf-8" />

</head>

## Echo :

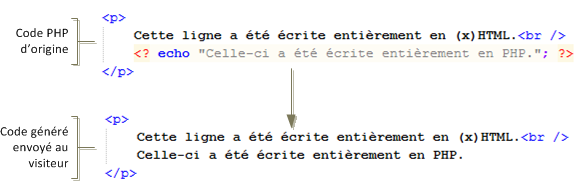
<?php echo "Ceci est du <strong>texte</strong>"; ?>

On peut aussi insérer du code PHP dans une balise comme :

<?php echo "Cette ligne a été écrite \"uniquement\" en PHP."; ?

Pour exécuter du code, on l’enregistre dans www de Wamp, sinon ça ne marche pas, c’est un langage serveur et non interprété.

Si on oublie le point-virgule à la fin de la ligne, on aura « parse error ».



### Les commentaires :

* Les commentaires monolignes : //
* Les commentaires multilignes /\*…\*/

# Inclure des portions de page

Avec html on doit recopier le même code sur chaque page, avec PHP on fait des include de partie de code, on code donc qu’une seule fois.

<?php include("menus.php"); ?>

Ici on a menus.php que l’on a codé, et index.php qui est la page principale. On fait un include sur la page index.

Ainsi une structure type ressemble à :

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Mon super site</title>

</head>

<body>

<?php include("entete.php"); ?>

<?php include("menus.php"); ?>

<!-- Le corps -->

<div id="corps">

<h1>Mon super site</h1>

<p>

Bienvenue sur mon super site !<br />

Vous allez adorer ici, c'est un site génial qui va parler de... euh... Je cherche encore un peu le thème de mon site. :-D

</p>

</div>

<!-- Le pied de page -->

<?php include("pied\_de\_page.php"); ?>

</body>

</html>

# Les variables

## Les types de variables :

* String : chaine de caractère
* Int : nombre entier
* Float : décimaux
* Bool : vrai ou faux
* NULL : rien

## Déclaration de variable :

<?php

$age\_du\_visiteur = 17;

?>

Pour stocker des chaînes de caractères on utilise des doubles guillemets :

<?php

$nom\_du\_visiteur = "Mateo21";

$nom\_du\_visiteur = 'Mateo21';

?>

## Afficher et concaténer des variables :

<?php

$age\_du\_visiteur = 17;

echo $age\_du\_visiteur;

?>

Pour afficher une variable on utilise un simple echo.

Pour concaténer :

<?php

$age\_du\_visiteur = 17;

echo "Le visiteur a ";

echo $age\_du\_visiteur;

echo " ans";

?>

Ou concaténer dans une seule ligne avec des guillemets doubles :

<?php

$age\_du\_visiteur = 17;

echo "Le visiteur a $age\_du\_visiteur ans";

?>

Pour concaténer avec des guillemets simples il faut sortir la variable de la parenthèse et les séparer à l’aide de point :

<?php

$age\_du\_visiteur = 17;

echo 'Le visiteur a ' . $age\_du\_visiteur . ' ans';

?>

C’est aussi la méthode la plus rapide car PHP n’a pas besoin de chercher où est la variable, il sait direct.

## Les opérations :

Comme pour les autres langages, c’est la même méthode … rappel : le modulo renvoie le reste de la division.

# Les conditions

| **Symbole** | **Signification** |
| --- | --- |
| == | Est égal à |
| > | Est supérieur à |
| < | Est inférieur à |
| >= | Est supérieur ou égal à |
| <= | Est inférieur ou égal à |
| != | Est différent de |

## If … else :

<?php

$age = 8;

if ($age <= 12) // SI l'âge est inférieur ou égal à 12

{

echo "Salut gamin ! Bienvenue sur mon site !<br />";

$autorisation\_entrer = "Oui";

}

else // SINON

{

echo "Ceci est un site pour enfants, vous êtes trop vieux pour pouvoir entrer. Au revoir !<br />";

$autorisation\_entrer = "Non";

}

echo "Avez-vous l'autorisation d'entrer ? La réponse est : $autorisation\_entrer";

?>

Autre structure :

<?php

if ($autorisation\_entrer == "Oui") // SI on a l'autorisation d'entrer

{

// instructions à exécuter quand on est autorisé à entrer

}

elseif ($autorisation\_entrer == "Non") // SINON SI on n'a pas l'autorisation d'entrer

{

// instructions à exécuter quand on n'est pas autorisé à entrer

}

else // SINON (la variable ne contient ni Oui ni Non, on ne peut pas agir)

{

echo "Euh, je ne connais pas ton âge, tu peux me le rappeler s'il te plaît ?";

}

?>

Inutile de mettre ==true pour certain cas :

<?php

if ($autorisation\_entrer)

{

echo "Bienvenue petit nouveau. :o)";

}

else

{

echo "T'as pas le droit d'entrer !";

}

?>

Et le même modèle avec la négation :

On utilise le « ! » pour la négation

<?php

if (! $autorisation\_entrer)

{

}

?>

## Conditions multiples :

| **Mot-clé** | **Signification** | **Symbole équivalent** |
| --- | --- | --- |
| AND | Et | && |
| OR | Ou | || |

Exemple :

<?php

if ($age <= 12 AND $langue == "français")

{

echo "Bienvenue sur mon site !";

}

elseif ($age <= 12 AND $langue == "anglais")

{

echo "Welcome to my website!";

}

?>

Astuce : les 2 suivants sont identiques

<?php

if ($variable == 23)

{

echo '<strong>Bravo !</strong> Vous avez trouvé le nombre mystère !';

}

?>

<?php

if ($variable == 23)

{

?>

<strong>Bravo !</strong> Vous avez trouvé le nombre mystère !

<?php

}

?>

Inutile de s’embêter avec le « echo » et les anti-slash, on peut directement insérer du html.

## Le switch-case :

<?php

$note = 10;

switch ($note) // on indique sur quelle variable on travaille

{

case 0: // dans le cas où $note vaut 0

echo "Tu es vraiment un gros nul !!!";

break;

case 5: // dans le cas où $note vaut 5

echo "Tu es très mauvais";

break;

case 7: // dans le cas où $note vaut 7

echo "Tu es mauvais";

break;

case 10: // etc. etc.

echo "Tu as pile poil la moyenne, c'est un peu juste…";

break;

case 12:

echo "Tu es assez bon";

break;

case 16:

echo "Tu te débrouilles très bien !";

break;

case 20:

echo "Excellent travail, c'est parfait !";

break;

default:

echo "Désolé, je n'ai pas de message à afficher pour cette note";

}

?>

## Les ternaires :

On teste une valeur dans une condition et on affecte une valeur à la variable en fonction de sa réponse true ou false :

<?php

$age = 24;

$majeur = ($age >= 18) ? true : false;

?>

Ce code est équivalent à true si $age >=18 et false sinon.

# Les boucles

## Le while :

<?php

$nombre\_de\_lignes = 1;

while ($nombre\_de\_lignes <= 100)

{

echo 'Je ne dois pas regarder les mouches voler quand j\'apprends le PHP.<br />';

$nombre\_de\_lignes++; // $nombre\_de\_lignes = $nombre\_de\_lignes + 1

}

?>

## Le for :

<?php

for ($nombre\_de\_lignes = 1; $nombre\_de\_lignes <= 100; $nombre\_de\_lignes++)

{

echo 'Ceci est la ligne n°' . $nombre\_de\_lignes . '<br />';

}

?>

# Les fonctions

Exemple de déclaration de fonction :

<?php

fonctionImaginaire(17, 'Vert', true, 41.7);

?>

## Quelques fonctions utiles :

### Strlen() :

Mesure la longueur d’une chaîne de caractère. Strlen($variable)

<?php

$phrase = 'Bonjour tout le monde ! Je suis une phrase !';

$longueur = strlen($phrase);

echo 'La phrase ci-dessous comporte ' . $longueur . ' caractères :<br />' . $phrase;

?>

### Str\_replace() :

Rechercher et remplacer.

Exemple : on cherche les b, on les remplace par des p dans bim bam boum.

<?php

$ma\_variable = str\_replace('b', 'p', 'bim bam boum');

echo $ma\_variable;

?>

### Str\_shuffle() :

Mélange les lettres d’une chaîne de caractère. Str\_shuffle($variable)

<?php

$chaine = 'Cette chaîne va être mélangée !';

$chaine = str\_shuffle($chaine);

echo $chaine;

?>

### Strlower() :

Ecrit en minuscule.

<?php

$chaine = 'COMMENT CA JE CRIE TROP FORT ???';

$chaine = strtolower($chaine);

echo $chaine;

?>

### Récupérer les dates :

| **Paramètre** | **Description** |
| --- | --- |
| H | Heure |
| i | Minute |
| d | Jour |
| m | Mois |
| Y | Année |

Il faut absolument respecter les majuscules.

Exemple :

<?php

// Enregistrons les informations de date dans des variables

$jour = date('d');

$mois = date('m');

$annee = date('Y');

$heure = date('H');

$minute = date('i');

// Maintenant on peut afficher ce qu'on a recueilli

echo 'Bonjour ! Nous sommes le ' . $jour . '/' . $mois . '/' . $annee . 'et il est ' . $heure. ' h ' . $minute;

?>

## Créer ses propres fonctions :

Exemple :

<?php

function DireBonjour($nom)

{

echo 'Bonjour ' . $nom . ' !<br />';

}

DireBonjour('Marie');

DireBonjour('Patrice');

DireBonjour('Edouard');

DireBonjour('Pascale');

DireBonjour('François');

DireBonjour('Benoît');

DireBonjour('Père Noël');

?>

Autre exemple :

<?php

// Ci-dessous, la fonction qui calcule le volume du cône

function VolumeCone($rayon, $hauteur)

{

$volume = $rayon \* $rayon \* 3.14 \* $hauteur \* (1/3); // calcul du volume

return $volume; // indique la valeur à renvoyer, ici le volume

}

$volume = VolumeCone(3, 1);

echo 'Le volume d\'un cône de rayon 3 et de hauteur 1 est de ' . $volume;

?>

# Les tableaux

## Les tableaux numérotés :

Ce sont les array().

| **Clé** | **Valeur** |
| --- | --- |
| 0 | François |
| 1 | Michel |
| 2 | Nicole |
| 3 | Véronique |

Pour le construire :

<?php

// La fonction array permet de créer un array

$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique', 'Benoît');

?>

Ou alors manuellement :

<?php

$prenoms[] = 'François'; // Créera $prenoms[0]

$prenoms[] = 'Michel'; // Créera $prenoms[1]

$prenoms[] = 'Nicole'; // Créera $prenoms[2]

?>

Ici on laisse PHP mettre les numéros automatiquement. En écrivant les numéros soit même ça marche aussi.

### Récupérer une valeur :

<?php

echo $prenoms[1];

?>

## Le tableau associatif :

Permet de mettre une étiquette devant la case au lieu d’un numéro. Pratique pour différencier des paramètres comme nom, prénom …

<?php

// On crée notre array $coordonnees

$coordonnees = array (

'prenom' => 'François',

'nom' => 'Dupont',

'adresse' => '3 Rue du Paradis',

'ville' => 'Marseille');

?>

On met l’étiquette devant la valeur et séparé d’une flèche =>.

Rappel : ici on n’a qu’une seule instruction, un seul point-virgule, on aurait pu tout écrire sur une seule ligne aussi. On utilise aussi array().

Pour le créer case par case, même procédé :

<?php

$coordonnees['prenom'] = 'François';

$coordonnees['nom'] = 'Dupont';

$coordonnees['adresse'] = '3 Rue du Paradis';

$coordonnees['ville'] = 'Marseille';

?>

### Récupérer une valeur :

Même procédé que le tableau numéroté :

<?php

echo $coordonnees['ville'];

?>

## Parcourir tout le tableau : for, foreach, print\_r :

### La boucle for :

<?php

// On crée notre array $prenoms

$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique', 'Benoît');

// Puis on fait une boucle pour tout afficher :

for ($numero = 0; $numero < 5; $numero++)

{

echo $prenoms[$numero] . '<br />'; // affichera $prenoms[0], $prenoms[1] etc.

}

?>

### Foreach :

Comme la boucle for mais à chaque ligne, va stocker la valeur dans une variable temporaire.

<?php

$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique', 'Benoît');

foreach($prenoms as $element)

{

echo $element . '<br />'; // affichera $prenoms[0], $prenoms[1] etc.

}

?>

Ici on stocke dans la variable temporaire $element. La syntaxe est : $variable as $autreVariablePourStocker.

On peut aussi récupérer la clé de l’élément :

<?php

$coordonnees = array (

'prenom' => 'François',

'nom' => 'Dupont',

'adresse' => '3 Rue du Paradis',

'ville' => 'Marseille');

foreach($coordonnees as $cle => $element)

{

echo '[' . $cle . '] vaut ' . $element . '<br />';

}

?>

La syntaxe a changé, on met as $laClé => $element.

### Print\_r :

Surtout utilisé pour déboguer, print\_r est une sorte de « echo » spécialisé dans les talbeaux. On veut juste savoir ce qu’il contient.

On l’utilise surtout pour déboguer car elle ne renvoie pas de code HTML.

<?php

$coordonnees = array (

'prenom' => 'François',

'nom' => 'Dupont',

'adresse' => '3 Rue du Paradis',

'ville' => 'Marseille');

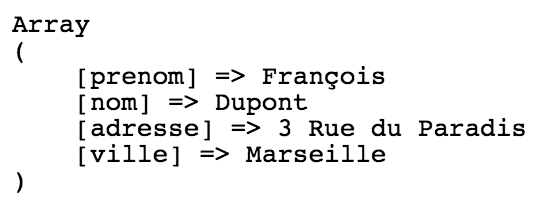
echo '<pre>';

print\_r($coordonnees);

echo '</pre>';

?>

On utilise <pre></pre> pour avoir le code HTML correspondant. On aura un affichage comme suit :



## Recherche dans un tableau :

### Vérifier que la clé existe dans l’array () :

Array\_key\_exists(‘cle’,$tableau). Il renvoie un Booléen.

Exemple :

<?php

$coordonnees = array (

'prenom' => 'François',

'nom' => 'Dupont',

'adresse' => '3 Rue du Paradis',

'ville' => 'Marseille');

if (array\_key\_exists('nom', $coordonnees))

{

echo 'La clé "nom" se trouve dans les coordonnées !';

}

if (array\_key\_exists('pays', $coordonnees))

{

echo 'La clé "pays" se trouve dans les coordonnées !';

}

?>

### Vérifier qu’une valeur existe dans array() :

In\_array(«’valeur’, $tableau). Renvoie un Booléen.

<?php

$fruits = array ('Banane', 'Pomme', 'Poire', 'Cerise', 'Fraise', 'Framboise');

if (in\_array('Myrtille', $fruits))

{

echo 'La valeur "Myrtille" se trouve dans les fruits !';

}

if (in\_array('Cerise', $fruits))

{

echo 'La valeur "Cerise" se trouve dans les fruits !';

}

?>

Syntaxe assez similaire.

### Récupérer la clé d’une valeur :

Array\_search(‘valeur’, $tableau). Renvoie un numéro en cas de tableau numéroté, une clé en cas de tableau associatif, false s’il n’a rien trouvé.

<?php

$fruits = array ('Banane', 'Pomme', 'Poire', 'Cerise', 'Fraise', 'Framboise');

$position = array\_search('Fraise', $fruits);

echo '"Fraise" se trouve en position ' . $position . '<br />';

$position = array\_search('Banane', $fruits);

echo '"Banane" se trouve en position ' . $position;

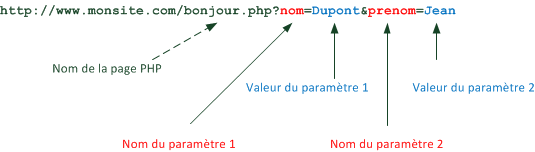
?>

# Transmettre des données dans l’URL

## Envoyer des données :

SI notre site est monsite.com, pour accéder à la page bonjour.php, on fera <http://www.monsite.com/bonjour.php>

Pour transmettre des infos :



Mais la longueur de 256 caractères est à respecter, tous les navigateurs ne gèrent pas au-delà de cette taille. La solution, créer des paramètres dans un fichier et les transmettre dans un autre.

SI on a deux fichier, index.php(page d’accueil) et bonjour.php ;= :

On veut que des paramètres de index.php arrivent dans bonjour.php.

Dans index.php :

<a href="bonjour.php?nom=Dupont&amp;prenom=Jean">Dis-moi bonjour !</a>

Si on écrit & au lieu de &amp ; le code html ne passera pas la validation W3C.

## Récupérer les données :

La page bonjour.php va automatiquement créer un tableau associatif comme suit :

| **Nom** | **Valeur** |
| --- | --- |
| $\_GET['nom'] | Dupont |
| $\_GET['prenom'] | Jean |

On a les clés $\_GET pour chaque élément passé d’une page à l’autre.

Exemple :

On a un fichier index avec la ligne pour passer les paramètres en URL, et un fichier bonjour avec la ligne :

<p>Bonjour <?php echo $\_GET['prenom'] . ' ' . $\_GET['nom']; ?> !</p>

Donc quand on clique sur le lien dans index.php, on arrive sur bonjour.php avec le texte là.

## Contrôler les valeurs des paramètres :

On veut boucler et afficher 8 fois les paramètres. S’il manque quelque chose, on ne fait rien.

Exemple :

<?php

if (isset($\_GET['prenom']) AND isset($\_GET['nom']) AND isset($\_GET['repeter']))

{

for ($i = 0 ; $i < $\_GET['repeter'] ; $i++)

{

echo 'Bonjour ' . $\_GET['prenom'] . ' ' . $\_GET['nom'] . ' !<br />';

}

}

else

{

echo 'Il faut renseigner un nom, un prénom et un nombre de répétitions !';

}

?>

On dit que l’on fait l’action si les paramètres sont définis, sinon on ne fait rien.

On utilise isset() qui peut se définir par : s’il existe.

Il faut être prévoyant, si on modifie l’url avec en valeur 1245679 ? On peut faire planter la page ou attendre la fin de la boucle .. De même si on met « grenouille » dans l’url ?

Au final :

<?php

if (isset($\_GET['prenom']) AND isset($\_GET['nom']) AND isset($\_GET['repeter']))

{

// 1 : On force la conversion en nombre entier

$\_GET['repeter'] = (int) $\_GET['repeter'];

// 2 : Le nombre doit être compris entre 1 et 100

if ($\_GET['repeter'] >= 1 AND $\_GET['repeter'] <= 100)

{

for ($i = 0 ; $i < $\_GET['repeter'] ; $i++)

{

echo 'Bonjour ' . $\_GET['prenom'] . ' ' . $\_GET['nom'] . ' !<br />';

}

}

}

else

{

echo 'Il faut renseigner un nom, un prénom et un nombre de répétitions !';

}

?>

* On a forcé la conversion avec (int)
* On a vérifié que le nombre était raisonnable.

On a sécurisé notre URL.

# Transmettre des données avec les formulaires

## Créer la base du formulaire :

Rappel : on utilise <form> pour créer un formulaire.

<form method="post" action="cible.php">

<p>

On insèrera ici les éléments de notre formulaire.

</p>

</form>

On met en action « cible.php » comparé à un simple formulaire en HTML.

* Il y a deux méthodes pour faire transiter les données :
  + GET pour passer les paramètres par l’URL
  + POST pour les faire transiter hors-URL.
* Action :
  + Définie la page appelé par le formulaire, c’est la page qui traitera les infos.

On aura donc : fomulaire.php qui contiendra le formulaire. Au submit c’est cible.php qui traitera les infos. On peut aussi appelé la page courante si celle-ci est aussi capable de traiter les infos.

On recevra un array POST au lieu de recevoir un array GET. Même procédé.

### Petites zones de texte :

Dans la page formulaire.php :

<p>

Cette page ne contient que du HTML.<br />

Veuillez taper votre prénom :

</p>

<form action="cible.php" method="post">

<p>

<input type="text" name="prenom" /> < !—name est obligatoire-->

<input type="submit" value="Valider" />

</p>

</form>

Dans cible.php :

<p>Bonjour !</p>

<p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <?php echo $\_POST['prenom']; ?> !</p>

<p>Si tu veux changer de prénom, <a href="formulaire.php">clique ici</a> pour revenir à la page formulaire.php.</p>

### Les grandes zones de texte :

<textarea name="message" rows="8" cols="45">

Votre message ici.

</textarea>

### Les listes déroulantes :

<select name="choix">

<option value="choix1">Choix 1</option>

<option value="choix2">Choix 2</option>

<option value="choix3">Choix 3</option>

<option value="choix4">Choix 4</option>

</select>

On peut rajouter selected = « selected » pour définir celui sélectionné par défaut.

### Les checkbox :

<input type="checkbox" name="case" id="case" /> <label for="case">Ma case à cocher</label>

On peut rajouter checked = « checked » pour définir une case cochée par défaut.

Exemple :

Aimez-vous les frites ?

<input type="radio" name="frites" value="oui" id="oui" checked="checked" /> <label for="oui">Oui</label>

<input type="radio" name="frites" value="non" id="non" /> <label for="non">Non</label>

Il faut bien penser à renseigner value, car c’est la valeur de la variable récupérée avec $\_POST[‘frites’] dans cible.php. Il vaudra donc oui ou non.

### Les champs cachés :

<input type="hidden" name="pseudo" value="Mateo21" />

Ce champ ne sera pas visible pour le visiteur mais sera récupérable avec $\_POST[‘pseudo’] et vaudra Mateo21. On l’utilisera plus tard.

## La faille XSS :

Avant-propos :

Quelqu’un peut copier le formulaire et le faire pointer vers la même cible.php. En clair, tout le monde peut modifier le code.

La faille XSS : cross-site-scripting. Dans un champ, on peut insérer du code HTML ou JavaScript et ça fonctionne.

La solution : transformer les <> en &lt et &gt : htmlspecialchars()

<p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <?php echo htmlspecialchars($\_POST['prenom']); ?> !</p>

Ou strip\_tags() pour les retirer au lieu de les convertir.

## L’envoi des fichiers :

Ça se passe en deux temps :

* Une page HTML avec le formulaire à remplir pour choisir le fichier à uploader
* Une page PHP récupère le fichier et l’enregistre sur le serveur.

On utilise déjà enctype pour que le navigateur sache qu’il doit envoyer un fichier :

<form action="cible\_envoi.php" method="post" enctype="multipart/form-data">

<p>

Formulaire d'envoi de fichier :<br />

<input type="file" name="monfichier" /><br />

<input type="submit" value="Envoyer le fichier" />

</p>

</form>

A partir de là, le fichier est dans un dossier temporaire, on décide si on l’accepte définitivement sur le serveur ou non avec move\_upload\_file.

Une variable $\_FILES[‘nom\_du\_fichier’]. Cette variable est un tableau qui contient plusieurs informations.

| **Variable** | **Signification** |
| --- | --- |
| $\_FILES['monfichier']['name'] | Contient le nom du fichier envoyé par le visiteur. |
| $\_FILES['monfichier']['type'] | Indique le type du fichier envoyé. Si c'est une image gif par exemple, le type seraimage/gif. |
| $\_FILES['monfichier']['size'] | Indique la taille du fichier envoyé. **Attention**: cette taille est en octets. Il faut environ 1 000 octets pour faire 1 Ko, et 1 000 000 d'octets pour faire 1 Mo. **Attention**: la taille de l'envoi est limitée par PHP. Par défaut, impossible d'uploader des fichiers de plus de 8 Mo. |
| $\_FILES['monfichier']['tmp\_name'] | Juste après l'envoi, le fichier est placé dans un répertoire temporaire sur le serveur en attendant que votre script PHP décide si oui ou non il accepte de le stocker pour de bon. Cette variable contient l'emplacement temporaire du fichier (c'est PHP qui gère ça). |
| $\_FILES['monfichier']['error'] | Contient un code d'erreur permettant de savoir si l'envoi s'est bien effectué ou s'il y a eu un problème et si oui, lequel. La variable vaut 0 s'il n'y a pas eu d'erreur. |

Pour vérifier si le fichier est bon ou non, 3 étapes :

* Tester s’il a bien été envoyé
* Vérifier sa taille
* Vérifier l’extension
* Valider l’upload du fichier

Exemple :

<?php

// Testons si le fichier a bien été envoyé et s'il n'y a pas d'erreur

if (isset($\_FILES['monfichier']) AND $\_FILES['monfichier']['error'] == 0)

{

// Testons si le fichier n'est pas trop gros

if ($\_FILES['monfichier']['size'] <= 1000000)

{

// Testons si l'extension est autorisée

$infosfichier = pathinfo($\_FILES['monfichier']['name']);

$extension\_upload = $infosfichier['extension'];

$extensions\_autorisees = array('jpg', 'jpeg', 'gif', 'png');

if (in\_array($extension\_upload, $extensions\_autorisees))

{

// On peut valider le fichier et le stocker définitivement

move\_uploaded\_file($\_FILES['monfichier']['tmp\_name'], 'uploads/' . basename($\_FILES['monfichier']['name']));

echo "L'envoi a bien été effectué !";

}

}

}

?>

# Variable superglobale, session et cookie

Les variables superglobales :

Sont des variables sont générés par le code.

Exemple :

$\_POST et $\_GET sont générés par PHP et on peut les récupérer dans le fichier ensuite.

Les sessions :

Sont mémorisés le temps d’une visite sur un site, sans avoir besoin de retaper une information par exemple.

Cookie :

Sont mémorisés dans l’ordinateur, pour se faciliter la vie au quotidien.

## Les variables superglobales :

Elles commencent toutes par un underscore et sont en majuscule. Ce sont des array.

Le plus simple pour vérifier le contenu d’une superglobale reste print\_r.

Quelques exemples :

* $\_SERVER comme $\_SERVER[‘REMOTE\_ADDR’] qui demande l’adresse ip de celui qui a demandé la page, pratique pour identifier.
* $\_ENV, ce sont des variables d’environnement souvent issu de server Linux.
* $\_SESSION, variable stockés le temps d’une visite
* $\_COOKIE variables stockés dans le pc
* $\_GET
* $\_POST
* $\_FILE

## Les sessions :

1. Le visiteur reçoit un code de session unique en hexadécimal. (PHPSESSID)
2. On créer des variables comme $\_SESSION[‘nom’] …
3. L’utilisateur se déconnecte, quitte, ou est victime du timout.

Pour cela :

* Session\_start(), à mettre en haut de chaque page pour démarrer la session.
* Session\_destroy() s’applique après le timout ou alors on créer un page déconnexion.

Il faut écrire session\_start() avant même d’écrire le < !DOCTYPE html>.

Exemple :

<?php

// On démarre la session AVANT d'écrire du code HTML

session\_start();

// On s'amuse à créer quelques variables de session dans $\_SESSION

$\_SESSION['prenom'] = 'Jean';

$\_SESSION['nom'] = 'Dupont';

$\_SESSION['age'] = 24;

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Titre de ma page</title>

</head>

<body>

<p>

Salut <?php echo $\_SESSION['prenom']; ?> !<br />

Tu es à l'accueil de mon site (index.php). Tu veux aller sur une autre page ?

</p>

<p>

<a href="mapage.php">Lien vers mapage.php</a><br />

<a href="monscript.php">Lien vers monscript.php</a><br />

<a href="informations.php">Lien vers informations.php</a>

</p>

</body>

</html>

## Les cookies :

Pour créer un cookie on utilise setcookie(‘nom du cookie’, ‘valeur du cookie’,’date d’expiration’).

L’expiration est en seconde et augmente de 1 toutes les secondes. On accède à cette variable avec time().

On créer donc un cookie ainsi :

<?php setcookie('pseudo', 'M@teo21', time() + 365\*24\*3600); ?>

Là le cookie expirera dans un an.

Recommandé d’activé httpOnly qui sécurisera le site en cas de code JavaScript si on a oublié de mettre htmlspecialchars().

On le créera plutôt ainsi :

<?php setcookie('pseudo', 'M@teo21', time() + 365\*24\*3600, null, null, false, true); ?>

Où True sert à rendre httpOnly activé.

Comme setcookie il ne marche que s’il est placé AVANT du code HTML.

<?php

setcookie('pseudo', 'M@teo21', time() + 365\*24\*3600, null, null, false, true); // On écrit un cookie

setcookie('pays', 'France', time() + 365\*24\*3600, null, null, false, true); // On écrit un autre cookie...

// Et SEULEMENT MAINTENANT, on peut commencer à écrire du code html

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Ma super page PHP</title>

</head>

<body>

etc.

### Afficher un cookie :

On accède aux valeurs avec la superglobale $\_COOKIE :

<p>

Hé ! Je me souviens de toi !<br />

Tu t'appelles <?php echo $\_COOKIE['pseudo']; ?> et tu viens de <?php echo $\_COOKIE['pays']; ?> c'est bien ça ?

</p>

### Modifier un cookie :

On refait un setcookie avec les mêmes coordonnées et cela écrasera l’ancien.

# Lire et écrire dans un fichier

## Autoriser l’écriture de fichier :

Sous windows ça n’existe pas, mais sous Linux, il faut utiliser CHMOD (par exemple 777).

Sous FileZilla : clic droit sur le fichier CHMOD ou permission de fichier et on entre 777.

## Ouvrir et fermer un fichier :

Ici un test avec un fichier text.txt que l’on envoie sur le serveur et auquel on applique le Chmod. On veut ouvrir le fichier texte et incrémenter une valeur à chaque visite du site.

<?php

// 1 : on ouvre le fichier

$monfichier = fopen('compteur.txt', 'r+');

// 2 : on fera ici nos opérations sur le fichier...

// 3 : quand on a fini de l'utiliser, on ferme le fichier

fclose($monfichier);

?>

| **Mode** | **Explication** |
| --- | --- |
| r | Ouvre le fichier en lecture seule. Cela signifie que vous pourrez seulement lire le fichier. |
| r+ | Ouvre le fichier en lecture et écriture. Vous pourrez non seulement lire le fichier, mais aussi y écrire (on l'utilisera assez souvent en pratique). |
| a | Ouvre le fichier en écriture seule. Mais il y a un avantage : si le fichier n'existe pas, il est automatiquement créé. |
| a+ | Ouvre le fichier en lecture et écriture. Si le fichier & n'existe pas, il est créé automatiquement. Attention : le répertoire doit avoir un CHMOD à 777 dans ce cas ! À noter que si le fichier existe déjà, le texte sera rajouté à la fin. |

Pour le fermer on utilise fclose($monfichier);

## Lire et écrire dans un fichier :

### Lire un fichier :

Pour la lecture on peut lrie :

* Caractère par caractère avec fgetc
* Ligne par ligne avec fgets

Avec l’ancien exemple, pour lire la valeur de nombre de vue dans compteur.txt :

<?php

// 1 : on ouvre le fichier

$monfichier = fopen('compteur.txt', 'r+');

// 2 : on lit la première ligne du fichier

$ligne = fgets($monfichier);

// 3 : quand on a fini de l'utiliser, on ferme le fichier

fclose($monfichier);

?>

### Ecrire dans un fichier :

Une seule méthode : fputs

<?php fputs($monfichier, 'Texte à écrire'); ?>

Mais, si on fait fopen, puis gets, on a lu la première ligne et le curseur et à la fin de cette ligne. Si on fait un fputs, on écrira à la fin de la ligne.

Pour placer le curseur où on veut on utilise fseek :

fseek($monfichier, 0); Ici 0 pour indiquer le début de la ligne

Le texte sera donc écrit par-dessus l’ancien. Il le remplacera.

D’où le code final :

 :

<?php

$monfichier = fopen('compteur.txt', 'r+');

$pages\_vues = fgets($monfichier); // On lit la première ligne (nombre de pages vues)

$pages\_vues += 1; // On augmente de 1 ce nombre de pages vues

fseek($monfichier, 0); // On remet le curseur au début du fichier

fputs($monfichier, $pages\_vues); // On écrit le nouveau nombre de pages vues

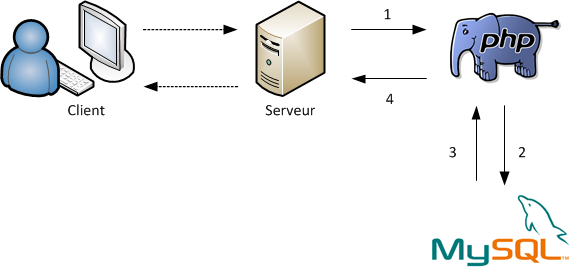
fclose($monfichier);

echo '<p>Cette page a été vue ' . $pages\_vues . ' fois !</p>';

?>

# Les bases de données

SGBD Système de Gestion de Base de Données : ici on utilisera MySQL.



Vocabulaire :

* La base est « la base de donné ».
* Dans la base on a des tables.
* Dans les tables on a des lignes : des entrées, et des colonnes : des champs.

Dans WAMP, les BDD sont dans data.

# PhpMyAdmin

## Créer une table :

On va dans phpmyadmin pour gérer ses bases de données. On peut en créer une directement.

Pendant la création d’une table, AUTO\_INCREMENT permet de générer d’incrémenter automatiquement l’id suivant à +1.

Pour les types ils sont classés par catégories.

* NUMERIC : ce sont les nombres. On y trouve des types dédiés aux petits nombres entiers (TINYINT), aux gros nombres entiers (BIGINT), aux nombres décimaux, etc.
* DATE and TIME : ce sont les dates et les heures. De nombreux types différents permettent de stocker une date, une heure, ou les deux à la fois.
* STRING : ce sont les chaînes de caractères. Là encore, il y a des types adaptés à toutes les tailles.
* SPATIAL : cela concerne les bases de données spatiales, utiles pour ceux qui font de la cartographie. Ce ne sera pas notre cas, donc nous n'en parlerons pas ici.

En fait, phpMyAdmin a eu la bonne idée de proposer au tout début de cette liste les quatre types de données les plus courants :

* INT : nombre entier ;
* VARCHAR : texte court (entre 1 et 255 caractères) ;
* TEXT : long texte (on peut y stocker un roman sans problème) ;
* DATE : date (jour, mois, année).

## Modifier une table :

On peut insérer ou modifier la table dans la partie Insérer de phpmyadmin.

# Lire des données SQL

## Se connecter à la base de données en PHP :

Première chose à faire : demander une connexion.

On utilise l’extension PDO avec lequel on peut interroger n’importe quelle BDD.

Pour vérifier que PDO est activé : clic sur WAMP et dans les extension PHP, vérifier que php\_pdo\_mysql est bien coché.

### Se connecter à mysql avec PDO :

4 choses à renseigner :

* **le nom de l'hôte** : c'est l'adresse de l'ordinateur où MySQL est installé (comme une adresse IP). Le plus souvent, MySQL est installé sur le même ordinateur que PHP : dans ce cas, mettez la valeurlocalhost(cela signifie « sur le même ordinateur »). Néanmoins, il est possible que votre hébergeur web vous indique une autre valeur à renseigner (qui ressemblerait à ceci :sql.hebergeur.com). Dans ce cas, il faudra modifier cette valeur lorsque vous enverrez votre site sur le Web ;
* **la base** : c'est le nom de la base de données à laquelle vous voulez vous connecter. Dans notre cas, la base s'appelle test. Nous l'avons créée avec phpMyAdmin dans le chapitre précédent ;
* **le login** : il permet de vous identifier. Renseignez-vous auprès de votre hébergeur pour le connaître. Le plus souvent (chez un hébergeur gratuit), c'est le même login que vous utilisez pour le FTP ;
* **le mot de passe** : il y a des chances pour que le mot de passe soit le même que celui que vous utilisez pour accéder au FTP. Renseignez-vous auprès de votre hébergeur.

Pour le moment comme on travaille en local, on utilisera localhost.

Pour se connecter on fera donc :

<?php

// Sous WAMP (Windows)

$bdd = new *PDO*('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');

// Sous MAMP (Mac)

$bdd = new *PDO*('mysql:host=localhost;dbname=test;charset=utf8', 'root', 'root');

?>

(mdp : 2588).

On créé donc un objet dans cet ordre :

* le nom d'hôte (localhost) ;
* la base de données (test) ;
* le login (root) ;
* le mot de passe (sous WAMP il n'y a pas de mot de passe, j'ai donc mis une chaîne vide, sous MAMP le mot de passe est root).

Le premier paramètre qui commence par mysql esst un DSN : Data Source Name.

### Tester la présence d’erreur :

<?php

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=jeux\_video;charset=utf8', 'root', '');

}

catch (Exception $e)

{

die('Erreur : ' . $e->getMessage());

}

?>

## Récupérer les données :

On doit parler avec mysqk :

<?php

$reponse = $bdd->query('SELECT \* FROM jeux\_video');

?>

Query signiei requête et est obligatoire.

Pour récupérer une entrée on exécute fetch() qui nous renvoie la première ligne :

<?php

$donnees = $reponse->fetch();

?>

Exemple complet :

<?php

try

{

// On se connecte à MySQL

$bdd = new *PDO*('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');

}

catch(Exception $e)

{

// En cas d'erreur, on affiche un message et on arrête tout

die('Erreur : '.$e->getMessage());

}

// Si tout va bien, on peut continuer

// On récupère tout le contenu de la table jeux\_video

$reponse = $bdd->query('SELECT \* FROM jeux\_video');

// On affiche chaque entrée une à une

while ($donnees = $reponse->fetch())

{

?>

<p>

<strong>Jeu</strong> : <?php echo $donnees['nom']; ?><br />

Le possesseur de ce jeu est : <?php echo $donnees['possesseur']; ?>, et il le vend à <?php echo $donnees['prix']; ?> euros !<br />

Ce jeu fonctionne sur <?php echo $donnees['console']; ?> et on peut y jouer à <?php echo $donnees['nbre\_joueurs\_max']; ?> au maximum<br />

<?php echo $donnees['possesseur']; ?> a laissé ces commentaires sur <?php echo $donnees['nom']; ?> : <em><?php echo $donnees['commentaires']; ?></em>

</p>

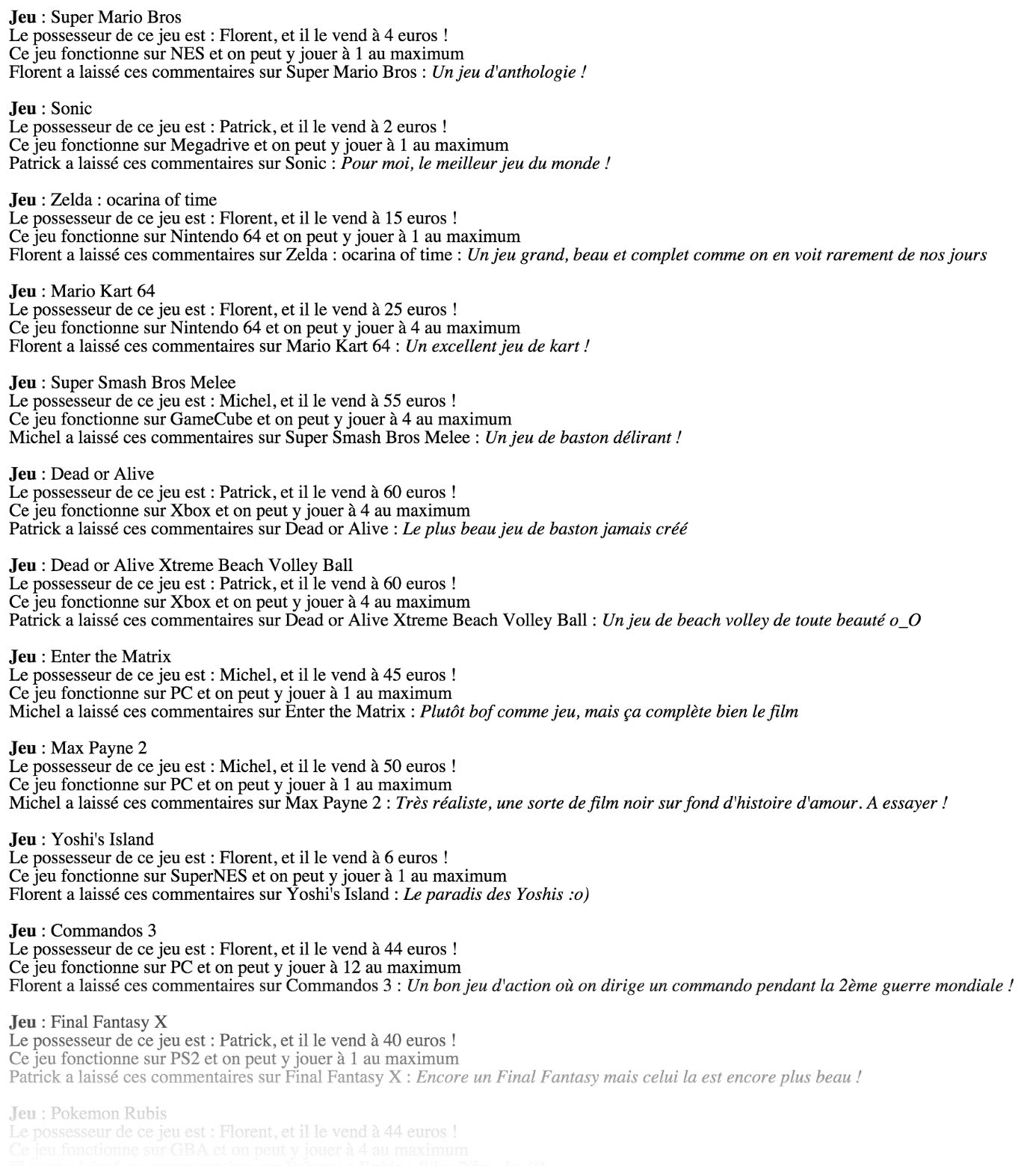
<?php

}

$reponse->closeCursor(); // Termine le traitement de la requête

?>

Ce qui renvoie :



## Filtrer les champs :

<?php

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

$reponse = $bdd->query('SELECT nom FROM jeux\_video');

while ($donnees = $reponse->fetch())

{

echo $donnees['nom'] . '<br />';

}

$reponse->closeCursor();

?>

### Where :

<?php

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=jeux\_video;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

$reponse = $bdd->query('SELECT nom, possesseur FROM jeux\_video WHERE possesseur=\'Patrick\'');

while ($donnees = $reponse->fetch())

{

echo $donnees['nom'] . ' appartient à ' . $donnees['possesseur'] . '<br />';

}

$reponse->closeCursor();

?>

On a trié les jeux qui appartiennent à Patrick.

Ou des requêtes à plusieurs critères :

SELECT \* FROM jeux\_video WHERE possesseur='Patrick' AND prix < 20

Il existe aussi le mot « OR ».

### ORDER BY :

<?php

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=jeux\_video;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

$reponse = $bdd->query('SELECT nom, prix FROM jeux\_video ORDER BY prix');

while ($donnees = $reponse->fetch())

{

echo $donnees['nom'] . ' coûte ' . $donnees['prix'] . ' EUR<br />';

}

$reponse->closeCursor();

?>

Sera forcément ascendant. Pour faire en descendant :

SELECT \* FROM jeux\_video ORDER BY prix DESC

### LIMIT :

Permet d’afficher seulement qu’une partie des résultats. Exemple : du 0 au 10 ème ou du 5 ay 15 ème. On écrit d’abords le point de départ puis le nombre de résultat souhaité.

SELECT \* FROM jeux\_video LIMIT 0, 20

Exemple : les 10 premiers jeux de la table.

<?php

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=jeux\_video;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

$reponse = $bdd->query('SELECT nom FROM jeux\_video LIMIT 0, 10');

echo '<p>Voici les 10 premières entrées de la table jeux\_video :</p>';

while ($donnees = $reponse->fetch())

{

echo $donnees['nom'] . '<br />';

}

$reponse->closeCursor();

?>

Exemple : avec les 3 :

SELECT nom, possesseur, console, prix FROM jeux\_video WHERE console='Xbox' OR console='PS2' ORDER BY prix DESC LIMIT 0,10

## Construire des requêtes en fonction de variables :

<?php

$reponse = $bdd->query('SELECT nom FROM jeux\_video WHERE possesseur=\'' . $\_GET['possesseur'] . '\'');

?>

Est le schéma QU’IL NE FAUT PAS FAIRE. La solution : les requêtes séparées.

### Avec des marqueurs « ? » :

<?php

$req = $bdd->prepare('SELECT nom FROM jeux\_video WHERE possesseur = ?');

?>

Ici on a préparé la requête sans sa partie variable.

<?php

$req = $bdd->prepare('SELECT nom FROM jeux\_video WHERE possesseur = ?');

$req->execute(array($\_GET['possesseur']));

?>

Maintenant la requête est exécutée avec les paramètres passés grace à execute sous forme de tableau (array()).

Si on a plusieurs paramètres :

<?php

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=jeux\_video;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

$req = $bdd->prepare('SELECT nom, prix FROM jeux\_video WHERE possesseur = ? AND prix <= ? ORDER BY prix');

$req->execute(array($\_GET['possesseur'], $\_GET['prix\_max']));

echo '<ul>';

while ($donnees = $req->fetch())

{

echo '<li>' . $donnees['nom'] . ' (' . $donnees['prix'] . ' EUR)</li>';

}

echo '</ul>';

$req->closeCursor();

?>

On peut remplacer les « ? » par des marqueurs nominatifs. Exemple : :prixMax ou :possesseur.

## Traquer les erreurs :

Pour afficher des messages d’erreurs plus précis, on doit remplacer quelques variables dans la phase de connexion à la BDD.

<?php

$bdd = new *PDO*('mysql:host=localhost;dbname=jeux\_video;charset=utf8', 'root', '', array(*PDO*::ATTR\_ERRMODE => *PDO*::ERRMODE\_EXCEPTION));

?>

# Ecrire des données

## INSERT : ajoute des données :

INSERT INTO jeux\_video(ID, nom, possesseur, console, prix, nbre\_joueurs\_max, commentaires) VALUES('', 'Battlefield 1942', 'Patrick', 'PC', 45, 50, '2nde guerre mondiale')

Seul les textes ont besoin d’être entouré de simple guillemet.

On peut même se passer de mettre ID vu qu’il est automatiquement incrémenté.

Une deuxième méthode en se passant de donner le détail des colonnes mais en renseignant tous les champs obligatoirement :

Exec permet de signifier qu’on va modifier la table.

INSERT INTO jeux\_video VALUES('', 'Battlefield 1942', 'Patrick', 'PC', 45, 50,

'2nde guerre mondiale')

Exemple complet :

<?php

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

// On ajoute une entrée dans la table jeux\_video

$bdd->exec('INSERT INTO jeux\_video(nom, possesseur, console, prix, nbre\_joueurs\_max, commentaires) VALUES(\'Battlefield 1942\', \'Patrick\', \'PC\', 45, 50, \'2nde guerre mondiale\')');

echo 'Le jeu a bien été ajouté !';

?>

### Insertion avec des requêtes séparées :

<?php

$req = $bdd->prepare('INSERT INTO jeux\_video(nom, possesseur, console, prix, nbre\_joueurs\_max, commentaires) VALUES(:nom, :possesseur, :console, :prix, :nbre\_joueurs\_max, :commentaires)');

$req->execute(array(

'nom' => $nom,

'possesseur' => $possesseur,

'console' => $console,

'prix' => $prix,

'nbre\_joueurs\_max' => $nbre\_joueurs\_max,

'commentaires' => $commentaires

));

echo 'Le jeu a bien été ajouté !';

?>

## Update pour modifier des valeurs :

UPDATE jeux\_video SET prix = 10, nbre\_joueurs\_max = 32 WHERE ID = 51

Ou alors avec le nom : WHERE nom =’Battlefield 1942’.

On peut ainsi mettre à jour une grande partie de la table :

UPDATE jeux\_video SET possesseur = 'Florent' WHERE possesseur = 'Michel'

Ici on remplace toutes les occurrences de Florent par Michel.

Exec renvoie le nombre de modification, on peut donc la récupérer dans une variable :

<?php

$nb\_modifs = $bdd->exec('UPDATE jeux\_video SET possesseur = \'Florent\' WHERE possesseur = \'Michel\'');

echo $nb\_modifs . ' entrées ont été modifiées !';

?>

### En requête séparée :

<?php

$req = $bdd->prepare('UPDATE jeux\_video SET prix = :nvprix, nbre\_joueurs\_max = :nv\_nb\_joueurs WHERE nom = :nom\_jeu');

$req->execute(array(

'nvprix' => $nvprix,

'nv\_nb\_joueurs' => $nv\_nb\_joueurs,

'nom\_jeu' => $nom\_jeu

));

?>

## DELETE : supprimer des données

DELETE FROM jeux\_video WHERE nom='Battlefield 1942'

Si vous oubliez le WHERE, toutes les entrées seront supprimées. Cela équivaut à vider la table.

# Les fonction SQL

Deux types de fonction :

* **Les fonctions scalaires** agissent sur l’ensemble des entrées.
* **Les fonctions d’agrégats** qui traitent l’ensemble de la table pour ne retourner qu’une valeur.

## La fonction scalaire :

Découverte avec la fonction UPPER. Voici la syntaxe :

SELECT UPPER(nom) FROM jeux\_video

On l’écrit en majuscule par convention, pas par obligation.

On peut donner un nom à la table que l’on créé le temps de la requête.

SELECT UPPER(nom) AS nom\_maj FROM jeux\_video

C’est un champ virtuel que l’on appelle un alias.

Exemple :

<?php

$reponse = $bdd->query('SELECT UPPER(nom) AS nom\_maj FROM jeux\_video');

while ($donnees = $reponse->fetch())

{

echo $donnees['nom\_maj'] . '<br />';

}

$reponse->closeCursor();

?>

On peut tout sélectionner en ne modifiant qu’un seul champ :

SELECT UPPER(nom) AS nom\_maj, possesseur, console, prix FROM jeux\_video

## Quelques fonctions utiles :

* UPPER
* LOWER
* LENGTH
* ROUND : arrondir un nombre décimal

Exemple :

SELECT ROUND(prix, 2) AS prix\_arrondi FROM jeux\_video

Où 2 est le nombre de chiffre après la virgule.

## Les fonctions d’agrégats :

Explication avec AVG qui renvoie la valeur moyenne des entrées :

SELECT AVG(prix) AS prix\_moyen FROM jeux\_video

Exemple :

<?php

$reponse = $bdd->query('SELECT AVG(prix) AS prix\_moyen FROM jeux\_video');

$donnees = $reponse->fetch();

echo $donnees['prix\_moyen'];

$reponse->closeCursor();

?>

Ici on ne boucle pas parce qu’on sait qu’on n’aura qu’une seule entrée.

Attention : On ne peut pas faire un agrégat et sélectionner tout un champ en même temps. On aurait une entrée d’un côté et plusieurs entrées de l’autre.

### Quelques fonctions utiles :

* AVG
* SUM
* MAX
* MIN
* COUNT

Exemple :

SELECT COUNT(\*) AS nbjeux FROM jeux\_video WHERE possesseur='Florent'

### Compter uniquement le nombre de champ non nul :

SELECT COUNT(nbre\_joueurs\_max) AS nbjeux FROM jeux\_video

On indique le champ que l’on veut compter pour ne comptabiliser que ceux que l’on possède.

### Distinct :

On utilise distinct pour ne pas compter les doublons :

SELECT COUNT(DISTINCT possesseur) AS nbpossesseurs FROM jeux\_video

## Group by et having :

### Group by :

Sert à grouper des champs. On peut donc demander un prix moyen mais ce coup-ci par console et non un même prix moyen pour toutes les consoles :

SELECT AVG(prix) AS prix\_moyen, console FROM jeux\_video GROUP BY console

On aura le prix moyen de chaque console. Il faut utiliser GROUP BY en même temps qu’une fonction d’agrégat sinon elle ne sert à rien.

### HAVING :

HAVING agit sur les données une fois regroupées comparé à WHERE.

SELECT AVG(prix) AS prix\_moyen, console FROM jeux\_video GROUP BY console HAVING prix\_moyen <= 10

Ici on récupère le prix moyen des consoles uniquement s’il ne dépasse pas 10€.

# Les dates en SQL

## Les champs de type date :

Les types de date :

* DATE : stocke une date au format AAAA-MM-JJ (Année-Mois-Jour) ;
* TIME : stocke un moment au format HH:MM:SS (Heures:Minutes:Secondes) ;
* DATETIME : stocke la combinaison d'une date et d'un moment de la journée au format AAAA-MM-JJ HH:MM:SS. Ce type de champ est donc plus précis ;
* TIMESTAMP : stocke une date et un moment sous le format AAAAMMJJHHMMSS ;
* YEAR : stocke une année, soit au format AA, soit au format AAAA.

## Utilisation :

Pour un champ date :

SELECT pseudo, message, date FROM minichat WHERE date = '2010-04-02'

Pour un champ DATETIME :

SELECT pseudo, message, date FROM minichat WHERE date = '2010-04-02 15:28:22'

C’est la sélection qui est intéressante, on peut filtrer de manière précise :

SELECT pseudo, message, date FROM minichat WHERE date >= '2010-04-02 00:00:00' AND date <= '2010-04-18 00:00:00'

On peut aussi écrire cette requête comme ça :

Avec l’utilisation de **BETWEEN**

SELECT pseudo, message, date FROM minichat WHERE date BETWEEN '2010-04-02 00:00:00' AND '2010-04-18 00:00:00'

**Insérer une date :**

INSERT INTO minichat(pseudo, message, date) VALUES('Mateo', 'Message !', '2010-04-02 16:32:22')

## Fonctions de gestion de date :

La liste complète des fonction [sur ce site.](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html)

### La date actuelle :

**NOW()**, fonction que l'on utilise le plus souvent

INSERT INTO minichat(pseudo, message, date) VALUES('Mateo', 'Message !', NOW())

Elle sera insérée en fonction de son type. Il existe aussi **CURDATE()** et **CURTIME()** qui ne renvoient que la date ou que l’heure complète.

**DATE\_FORMAT() :**

La manière compliquée :

SELECT pseudo, message, DAY(date) AS jour, MONTH(date) AS mois, YEAR(date) AS annee, HOUR(date) AS heure, MINUTE(date) AS minute, SECOND(date) AS seconde FROM minichat

Pour formater une date et pouvoir changer l’ordre. Ainsi on peut faire :

<?php

echo $donnees['jour'] . '/' . $donnees['mois'] . '/' . $donnees['annee'] . '...';

?>

La manière simple :

SELECT pseudo, message, DATE\_FORMAT(date, '%d/%m/%Y %Hh%imin%ss') AS date FROM minichat

Même lien que plus haut pour plus de détail sur les DATE\_FORMAT.

### DATE\_ADD et DATE\_SUBB :

Pour ajouter ou soustraire des dates :

SELECT pseudo, message, DATE\_ADD(date, INTERVAL 15 DAY) AS date\_expiration FROM minichat

La date d’expiration correspond à la date + 15 jours.

On peut aussi utiliser : MONTH, YEAR, HOUR, MINUTE et SECOND

# TP blog

On utilise **nl2br()** qui convertie les retours à la ligne en balise HTML.

**Index.php**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Mon blog</title>

<link href="style.css" rel="stylesheet" />

</head>

<body>

<h1>Mon super blog !</h1>

<p>Derniers billets du blog :</p>

<?php

// Connexion à la base de données

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

// On récupère les 5 derniers billets

$req = $bdd->query('SELECT id, titre, contenu, DATE\_FORMAT(date\_creation, \'%d/%m/%Y à %Hh%imin%ss\') AS date\_creation\_fr FROM billets ORDER BY date\_creation DESC LIMIT 0, 5');

while ($donnees = $req->fetch())

{

?>

<div class="news">

<h3>

<?php echo htmlspecialchars($donnees['titre']); ?>

<em>le <?php echo $donnees['date\_creation\_fr']; ?></em>

</h3>

<p>

<?php

// On affiche le contenu du billet

echo nl2br(htmlspecialchars($donnees['contenu']));

?>

<br />

<em><a href="commentaires.php?billet=<?php echo $donnees['id']; ?>">Commentaires</a></em>

</p>

</div>

<?php

} // Fin de la boucle des billets

$req->closeCursor();

?>

</body>

</html>

**Commentaires.php**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Mon blog</title>

<link href="style.css" rel="stylesheet" />

</head>

<body>

<h1>Mon super blog !</h1>

<p><a href="index.php">Retour à la liste des billets</a></p>

<?php

// Connexion à la base de données

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

// Récupération du billet

$req = $bdd->prepare('SELECT id, titre, contenu, DATE\_FORMAT(date\_creation, \'%d/%m/%Y à %Hh%imin%ss\') AS date\_creation\_fr FROM billets WHERE id = ?');

$req->execute(array($\_GET['billet']));

$donnees = $req->fetch();

?>

<div class="news">

<h3>

<?php echo htmlspecialchars($donnees['titre']); ?>

<em>le <?php echo $donnees['date\_creation\_fr']; ?></em>

</h3>

<p>

<?php

echo nl2br(htmlspecialchars($donnees['contenu']));

?>

</p>

</div>

<h2>Commentaires</h2>

<?php

$req->closeCursor(); // Important : on libère le curseur pour la prochaine requête

// Récupération des commentaires

$req = $bdd->prepare('SELECT auteur, commentaire, DATE\_FORMAT(date\_commentaire, \'%d/%m/%Y à %Hh%imin%ss\') AS date\_commentaire\_fr FROM commentaires WHERE id\_billet = ? ORDER BY date\_commentaire');

$req->execute(array($\_GET['billet']));

while ($donnees = $req->fetch())

{

?>

<p><strong><?php echo htmlspecialchars($donnees['auteur']); ?></strong> le <?php echo $donnees['date\_commentaire\_fr']; ?></p>

<p><?php echo nl2br(htmlspecialchars($donnees['commentaire'])); ?></p>

<?php

} // Fin de la boucle des commentaires

$req->closeCursor();

?>

</body>

</html>

# Les jointures entre table

## Modéliser une relation :

Pour l’exemple de jeux\_video.sql : si on devait définir le numro de téléphone, adresse … de chaque personne, ce serait répétitif. On créé plutôt id\_propriétaire dans la table des jeux et on lie avec l’id d’une autre table dans laquelle on rentrera toutes les informations/

## Qu’est-ce qu’une jointure ?

Deux types de jointure :

* Jointures internes : ne sélectionnent que les données qui ont une correspondance entre les deux tables.
* Jointures externes : sélectionnent toutes les données même si elles ne trouvent pas de correspondance dans l’autre table.

Exemple :

Si on rajoute Romain dans la table propriétaire alors qu’il ne possède pas de jeu on aura :

**En jointure interne :**

| **nom\_jeu** | **prenom** |
| --- | --- |
| Super Mario Bros | Florent |
| Sonic | Patrick |
| … | … |

**Et en jointure externe :**

| **nom\_jeu** | **prenom** |
| --- | --- |
| Super Mario Bros | Florent |
| Sonic | Patrick |
| … | … |
| NULL | Romain |

### Les jointures internes :

Elles peuvent être effectuées de deux façons :

* WHERE
* JOIN

**WHERE :**

SELECT nom, prenom FROM proprietaires, jeux\_video

Ici on a une **colonne ambiguë**, car SQL ne sait pas s’il doit prendre nom de jeux\_video ou nom de propriétaire.

On doit faire la jointure avec WHERE de la sorte :

SELECT jeux\_video.nom, proprietaires.prenom

FROM proprietaires, jeux\_video

WHERE jeux\_video.ID\_proprietaire = proprietaires.ID

En utilisant des alias (recommandé) :

SELECT jeux\_video.nom AS nom\_jeu, proprietaires.prenom AS prenom\_proprietaire

FROM proprietaires, jeux\_video

WHERE jeux\_video.ID\_proprietaire = proprietaires.ID

Si on donne des alias aux noms des tables on obtient :

SELECT j.nom nom\_jeu, p.prenom prenom\_proprietaire

FROM proprietaires p, jeux\_video j

WHERE j.ID\_proprietaire = p.ID

On remarque que le AS est facultatif dans la première ligne.

**JOIN :**

On l’utilise pour éviter de confondre le WHERE de filtre et le WHERE de jointure.

Avec un JOIN on écrit la précédente requête de cette manière :

SELECT j.nom nom\_jeu, p.prenom prenom\_proprietaire

FROM proprietaires p

INNER JOIN jeux\_video j

ON j.ID\_proprietaire = p.ID

Les autres paramètres ne sont rajoutés qu’après cette syntaxe :

SELECT j.nom nom\_jeu, p.prenom prenom\_proprietaire

FROM proprietaires p

INNER JOIN jeux\_video j

ON j.ID\_proprietaire = p.ID

WHERE j.console = 'PC'

ORDER BY prix DESC

LIMIT 0, 10

### Jointures externes :

On utilise aussi le **JOIN** mais avec deux écritures possibles :

**LEFT JOIN :** récupérer toute la table de gauche.

SELECT j.nom nom\_jeu, p.prenom prenom\_proprietaire

FROM proprietaires p

LEFT JOIN jeux\_video j

ON j.ID\_proprietaire = p.ID

Ici INNER est remplacé par LEFT, on a donc propriétair en table de gauche et jeux\_video en table de droite. On demande à récupérer tout le contenu de celle de gauche (les propriétaires) même s’il n’y a pas de correspondance avec celle de droite.

**RIGHT JOIN :** récupérer toute la table de droite.

SELECT j.nom nom\_jeu, p.prenom prenom\_proprietaire

FROM proprietaires p

RIGHT JOIN jeux\_video j

ON j.ID\_proprietaire = p.ID

A l’inverse, ici on récupère tous les jeux même s’il n’y a pas de propriétaire associé.

# Créer des images en php

## Activer la bibliothèque GD :

La bibliothèque est d’office avec PHP mais n’est pas activée. Pour l’activer : clique sur WAMP et activé GD2.

## Bases de la création d’image :

**Header :**

Deux façons de générer une image en PHP :

* Soit on fait en sorte que notre script renvoie une image au lieu d’une page web : http://www.monsite.com/testgd.php , une image sera affiché et non une page.
* Soit on demande à PHP d’enregistrer l’image dans un fichier.

Première façon :

NG :

<?php

header ("Content-type: image/png");

?>

ATTENTION : on doit le signifier avant d’écrire la moindre code HTML.

Pour créer une image, soit on créer une image vide, soit on charge une image qui existe déjà.

**A partir d’une image vide :**

<?php

header ("Content-type: image/png");

$image = imagecreate(200,50);

?>

Dans l’ordre, les deux paramètres sont largeur et hauteur. Ici $image n’est pas une variable contenant un nombre ou autre, c’est une **RESSOURCE** qui contient un objet, ici image. Ca pourrait être un PDF que l’on ouvre avec fopen etc …

**A partir d’une image existante :**

Deux fonctions à connaitre :

* Imagecreatefromjpg
* Imagecreatefrompng

Exemple :

<?php

header ("Content-type: image/jpeg");

$image = imagecreatefromjpeg("couchersoleil.jpg");

?>

Quand on a fini, on affiche l’image avec deux fonctions au choix en fonction du format de l’image :

* Imagejpeg
* Imagepng

<?php

header ("Content-type: image/png"); // 1 : on indique qu'on va envoyer une image PNG

$image = imagecreate(200,50); // 2 : on crée une nouvelle image de taille 200 x 50

// 3 : on s'amuse avec notre image (on va apprendre à le faire)

imagepng($image); // 4 : on a fini de faire joujou, on demande à afficher l'image

?>

**Afficher la page PHP comme une image :**

<img src="image.php" />

**Enregistrer l’image sur le disque :**

<?php

$image = imagecreate(200,50);

// on fait joujou avec notre image

imagepng($image, "images/monimage.png"); // on enregistre l'image dans le dossier "images"

?>

Ici le header ne sert plus à rien, on rajoute un paramètre à imagepng() : sa destination.

Ainsi avec image, on ferait :

<img src="images/monimage.png" />

## Texte et couleurs :

<?php

header ("Content-type: image/png");

$image = imagecreate(200,50);

$orange = imagecolorallocate($image, 255, 128, 0);

$bleu = imagecolorallocate($image, 0, 0, 255);

$bleuclair = imagecolorallocate($image, 156, 227, 254);

$noir = imagecolorallocate($image, 0, 0, 0);

$blanc = imagecolorallocate($image, 255, 255, 255);

imagepng($image);

?>

En PHP, pour mettre de la couleur, on utilise imagecolorallocate() avec 4 paramètres :

* L’image en question
* Le rouge
* Le vert
* Le bleu

Celui écrit en premier sera la valeur effective, ici l’image sera donc orange.

### Ecrire du texte :

<?php

imagestring($image, $police, $x, $y, $texte\_a\_ecrire, $couleur);

?>

* $image : c'est notre fameuse variable qui contient l'image.
* $police : c'est la police de caractères que vous voulez utiliser. Vous devez mettre un nombre de 1 à 5 ; 1 = petit, 5 = grand. Il est aussi possible d'utiliser une police de caractères personnalisée, mais il faut avoir des polices dans un format spécial qu'il serait trop long de détailler ici. On va donc se contenter des polices par défaut. ;-)
* $x et $y : ce sont les coordonnées auxquelles vous voulez placer votre texte sur l'image. Et là vous vous dites : « Aïe, ça sent les maths » (comme quoi les maths, ça sert). Vous devez savoir que l'origine se trouve en haut à gauche de votre image. Le point de coordonnées (0, 0) représente donc le point tout en haut à gauche de l'image.



## Dessiner une forme :

### ImageSetPixel :

Dessine un pixel aux cordonnées x,y. (i majuscule).

ImageSetPixel ($image, $x, $y, $couleur);

### ImageLine :

Dessine une ligne entre deux points.

ImageLine ($image, $x1, $y1, $x2, $y2, $couleur);

### ImageEllipse :

Dessine une ellipse de centre x,y :

ImageEllipse ($image, $x, $y, $largeur, $hauteur, $couleur);

### ImageRectange :

Dessine un rectangle avec le coin en haut à gauche : x1, y1 et en bas à droite x2, y2.

ImageRectangle ($image, $x1, $y1, $x2, $y2, $couleur);

### ImgePolygon

Ici, array\_point dépend de $nombre, si on a 3, c’est un triangle, on met les coordonnées x puis y dans l’ordre dans l’array.

ImagePolygon ($image, $array\_points, $nombre\_de\_points, $couleur);

$points = array(10, 40, 120, 50, 160, 160); ImagePolygon ($image, $points, 3, $noir);

## Fonctions plus puissantes :

### Rendre une image transparente :

Ne fonctionne que sur les PNG.

<?php

imagecolortransparent($image, $couleur);

?>

Où ici on indique la couleur que l’on veut rendre transparent.

Exemple :

<?php

header ("Content-type: image/png");

$image = imagecreate(200,50);

$orange = imagecolorallocate($image, 255, 128, 0); // Le fond est orange (car c'est la première couleur)

$bleu = imagecolorallocate($image, 0, 0, 255);

$bleuclair = imagecolorallocate($image, 156, 227, 254);

$noir = imagecolorallocate($image, 0, 0, 0);

$blanc = imagecolorallocate($image, 255, 255, 255);

imagestring($image, 4, 35, 15, "Salut les Zéros !", $noir);

imagecolortransparent($image, $orange); // On rend le fond orange transparent

imagepng($image);

?>

### Mélanger deux images :

<?php

header ("Content-type: image/jpeg"); // L'image que l'on va créer est un jpeg

// On charge d'abord les images

$source = imagecreatefrompng("logo.png"); // Le logo est la source

$destination = imagecreatefromjpeg("couchersoleil.jpg"); // La photo est la destination

// Les fonctions imagesx et imagesy renvoient la largeur et la hauteur d'une image

$largeur\_source = imagesx($source);

$hauteur\_source = imagesy($source);

$largeur\_destination = imagesx($destination);

$hauteur\_destination = imagesy($destination);

// On veut placer le logo en bas à droite, on calcule les coordonnées où on doit placer le logo sur la photo

$destination\_x = $largeur\_destination - $largeur\_source;

$destination\_y = $hauteur\_destination - $hauteur\_source;

// On met le logo (source) dans l'image de destination (la photo)

imagecopymerge($destination, $source, $destination\_x, $destination\_y, 0, 0, $largeur\_source, $hauteur\_source, 60);

// On affiche l'image de destination qui a été fusionnée avec le logo

imagejpeg($destination);

?>

Imagecopymerge a plusieurs paramètres :

1. **L'image de destination** : ici $destination, la photo. C'est l'image qui va être modifiée et dans laquelle on va mettre notre logo.
2. **L'image source** : ici $source, c'est notre logo. Cette image n'est pas modifiée.
3. **L'abscisse à laquelle vous désirez placer le logo sur la photo** : il s'agit ici de l'abscisse du point située à la position $largeur\_de\_la\_photo - $largeur\_du\_logo.
4. **L'ordonnée à laquelle vous désirez placer le logo sur la photo** : de même, il s'agit de l'ordonnée du point sur la photo (ici, $hauteur\_de\_la\_photo - $hauteur\_du\_logo).
5. **L'abscisse de la source** : en fait, la fonction imagecopymerge permet aussi de ne prendre qu'une partie de l'image source. Ça peut devenir un peu compliqué, alors nous, on va dire qu'on prend tout le logo. On part donc du point situé aux coordonnées (0, 0) de la source. Mettez donc 0 pour l'abscisse.
6. **L'ordonnée de la source** : de même pour l'ordonnée. Mettez 0.
7. **La largeur de la source** : c'est la largeur qui détermine quelle partie de l'image source vous allez prendre. Nous on prend toute l'image source, ne vous prenez donc pas la tête non plus et mettez $largeur\_source.
8. **La hauteur de la source** : de même, mettez $hauteur\_source.
9. **Le pourcentage de transparence** : c'est un nombre entre 0 et 100 qui indique la transparence de votre logo sur la photo. Si vous mettez 0, le logo sera invisible (totalement transparent), et si vous mettez 100, il sera totalement opaque (il n'y aura pas d'effet de « fusion »). Mettez un nombre autour de 60-70, en général c'est bien. ;-)

<img src="copyrighter.php?image=tropiques.jpg" />

On peut donc récupérer l’image tropique dans une page copyright et récupérer l’image avec un $\_GET.

### Redimensionner une image :

On peut ici créer des images miniatures par exemple.

<?php

$source = imagecreatefromjpeg("couchersoleil.jpg"); // La photo est la source

$destination = imagecreatetruecolor(200, 150); // On crée la miniature vide

// Les fonctions imagesx et imagesy renvoient la largeur et la hauteur d'une image

$largeur\_source = imagesx($source);

$hauteur\_source = imagesy($source);

$largeur\_destination = imagesx($destination);

$hauteur\_destination = imagesy($destination);

// On crée la miniature

imagecopyresampled($destination, $source, 0, 0, 0, 0, $largeur\_destination, $hauteur\_destination, $largeur\_source, $hauteur\_source);

// On enregistre la miniature sous le nom "mini\_couchersoleil.jpg"

imagejpeg($destination, "mini\_couchersoleil.jpg");

?>

1. **L'image de destination** : c'est $destination, l'image qu'on a créée avec imagecreatetruecolor.
2. **L'image source** : l'image dont on veut créer la miniature ; ici, c'est notre couchersoleil.jpg qu'on a chargée avec imagecreatefromjpeg.
3. **L'abscisse du point à laquelle vous placez la miniature sur l'image de destination** : pour faire simple, on va dire que notre image de destination contiendra uniquement la miniature. Donc on placera la miniature aux coordonnées (0, 0), ce qui fait qu'il faut mettre 0 à cette valeur.
4. **L'ordonnée du point à laquelle vous placez la miniature sur l'image de destination** : pour les mêmes raisons, mettez 0.
5. **L'abscisse du point de la source** : ici, on prend toute l'image source et on en fait une miniature. Pour tout prendre, il faut partir du point (0, 0), ce qui fait que là encore on met 0 à cette valeur.
6. **L'ordonnée du point de la source** : encore 0.
7. **La largeur de la miniature** : un des paramètres les plus importants, qui détermine la taille de la miniature à créer. Dans notre cas notre miniature fait 200 pixels de large. On a stocké ce nombre dans la variable $largeur\_destination.
8. **La hauteur de la miniature** : de même pour la hauteur de la miniature à créer.
9. **La largeur de la source** : il suffit d'indiquer la taille de notre image source. On a stocké cette valeur dans $largeur\_source, donc on la réutilise ici.
10. **La hauteur de la source** : de même pour la hauteur.

Donc on supprime le header si on enregistre le fichier.

# Les expressions régulières

## Où utiliser une REGEX ?

* **PISIX** plus lent que le suivant, créé par PHP.
* **PCRE** créé par Perl, plus rapide et plus performant.

On va utiliser PCRE.

### Grep\_match :

Renvoie true or false.

<?php

if (preg\_match("\*\* Votre REGEX \*\*", "Ce dans quoi vous faites la recherche"))

{

echo 'Le mot que vous cherchez se trouve dans la chaîne';

}

else

{

echo 'Le mot que vous cherchez ne se trouve pas dans la chaîne';

}

?>

## Des recherches simples :

Un REGEX est toujours entouré de #.

<?php

if (preg\_match("#guitare#", "J'aime jouer de la guitare."))

{

echo 'VRAI';

}

else

{

echo 'FAUX';

}

?>

## La casse :

Le REGEX est sensible à la casse. Pour qu’il ne le soit plus, on ajoute l’option i.

| **Chaîne** | **Regex** | **Résultat** |
| --- | --- | --- |
| J'aime jouer de la guitare | #Guitare#i | VRAI |
| Vive la GUITARE ! | #guitare#i | VRAI |
| Vive la GUITARE ! | #guitare# | FAUX |

## Le OU :

| **Chaîne** | **Regex** | **Résultat** |
| --- | --- | --- |
| J'aime jouer de la guitare. | #guitare|piano# | VRAI |
| J'aime jouer du piano. | #guitare|piano# | VRAI |
| J'aime jouer du banjo. | #guitare|piano# | FAUX |
| J'aime jouer du banjo. | #guitare|piano|banjo# | VRAI |

## Début et fin de chaîne :

^ indique le début et $ indique la fin de chaîne.

| **Chaîne** | **Regex** | **Résultat** |
| --- | --- | --- |
| Bonjour petit zéro | #^Bonjour# | VRAI |
| Bonjour petit zéro | #zéro$# | VRAI |
| Bonjour petit zéro | #^zéro# | FAUX |
| Bonjour petit zéro !!! | #zéro$# | FAUX |

## Les classes de caractères :

### Des classes simples :

Les classes de caractères sont entre crochets, cela signifie que l’on peut avoir un des caractères entre crochet.

| **Chaîne** | **Regex** | **Résultat** |
| --- | --- | --- |
| La nuit, tous les chats sont gris | #gr[aoi]s# | VRAI |
| Berk, c'est trop gras comme nourriture | #gr[aoi]s# | VRAI |
| Berk, c'est trop gras comme nourriture | #gr[aoi]s$# | FAUX |
| Je suis un vrai zéro | #[aeiouy]$# | VRAI |
| Je suis un vrai zéro | #^[aeiouy]# | FAUX |

### Les intervalles de classes :

Avec « - » on autorise les plages de caractères :

| **Chaîne** | **Regex** | **Résultat** |
| --- | --- | --- |
| Cette phrase contient une lettre | #[a-z]# | VRAI |
| cette phrase ne comporte ni majuscule ni chiffre | #[A-Z0-9]# | FAUX |
| Je vis au 21e siècle | #^[0-9]# | FAUX |
| <h1>Une balise de titre HTML</h1> | #<h[1-6]># | VRAI |

Pour la négation on utilise ^ dans le crochet.

|  |  |  |
| --- | --- | --- |
| **Chaîne** | **Regex** | **Résultat** |
| Cette phrase contient autre chose que des chiffres | #[^0-9]# | VRAI |
| cette phrase contient autre chose que des majuscules et des chiffres | #[^A-Z0-9]# | VRAI |
| Cette phrase ne commence pas par une minuscule | #^[^a-z]# | VRAI |
| Cette phrase ne se termine pas par une voyelle | #[^aeiouy]$# | FAUX |
| ScrrmmmblllGnngngnngnMmmmmffff | #[^aeiouy]# | VRAI |

## Les quantificateurs :

### Les plus courants :

* ? : pour 0 ou 1 fois.
* + : 1 ou plusieurs fois.
* \* : 0, 1 ou plusieurs fois.

| **Chaîne** | **Regex** | **Résultat** |
| --- | --- | --- |
| eeeee | #e+# | VRAI |
| ooo | #u?# | VRAI |
| magnifique | #[0-9]+# | FAUX |
| Yahoooooo | #^Yaho+$# | VRAI |
| Yahoooooo c'est génial ! | #^Yaho+$# | FAUX |
| Blablablablabla | #^Bla(bla)\*$# | VRAI |

### Plus de précision : les accolades :

Trois façons de les utiliser :

* {3} : si on met juste un nombre, cela veut dire que la lettre (ou le groupe de lettres s'il est entre parenthèses) doit être répétée **3 fois exactement**.  
  #a{3}# fonctionne donc pour la chaîne « aaa ».
* {3,5} : ici, on a plusieurs possibilités. On peut avoir la lettre de **3 à 5 fois**.  
  #a{3,5}# fonctionne pour « aaa », « aaaa », « aaaaa ».
* {3,} : si vous mettez une virgule, mais pas de 2e nombre, ça veut dire qu'il peut y en avoir jusqu'à l'infini. Ici, cela signifie « **3 fois ou plus** ».  
  #a{3,}# fonctionne pour « aaa », « aaaa », « aaaaa », « aaaaaa », etc. Je ne vais pas tous les écrire, ça serait un peu long.

Avec des exemples :

| **Chaîne** | **Regex** | **Résultat** |
| --- | --- | --- |
| eeeee | #e{2,}# | VRAI |
| Blablablabla | #^Bla(bla){4}$# | FAUX |
| 546781 | #^[0-9]{6}$# | VRAI |

## Les métacaractères :

Ce sont les caractères speciaux :

# ! ^ $ ( ) [ ] { } ? + \* . \ |

Donc si on veut rechercher « Quoi ? », il faut **ECHAPPER** le ? avec un antislash :

| **Chaîne** | **regex** | **Résultat** |
| --- | --- | --- |
| Je suis impatient ! | #impatient \!# | VRAI |
| Je suis (très) fatigué | #\(très\) fatigué# | VRAI |
| J'ai sommeil… | #sommeil\.\.\.# | VRAI |
| Le smiley :-\ | #:-\\# | VRAI |

Pas besoin d’échapper les métacaractères dans les crochets, ils ne comptent plus.

Trois cas particuliers :

* « # » (dièse) : il sert toujours à indiquer la fin de la regex. Pour l'utiliser, vous DEVEZ mettre un antislash devant, même dans une classe de caractères.
* « ] » (crochet fermant) : normalement, le crochet fermant indique la fin de la classe. Si vous voulez vous en servir comme d'un caractère que vous recherchez, il faut là aussi mettre un antislash devant.
* « - » (tiret) : encore un cas un peu particulier. Le tiret – vous le savez – sert à définir un **intervalle de classe** (comme [a-z]). Et si vous voulez ajouter le tiret dans la liste des caractères possibles ? Eh bien il suffit de le mettre soit au début de la classe, soit à la fin. Par exemple : [a-z0-9-] permet de chercher une lettre, un chiffre ou un tiret.

## Les classes abrégées :

Ce sont des « raccourcis » :

| **Raccourci** | **Signification** |
| --- | --- |
| **\d** | Indique un chiffre. Ça revient exactement à taper [0-9] |
| **\D** | Indique ce qui n'est PAS un chiffre. Ça revient à taper [^0-9] |
| **\w** | Indique un caractère alphanumérique ou un tiret de soulignement. Cela correspond à [a-zA-Z0-9\_] |
| **\W** | Indique ce qui n'est PAS un mot. Si vous avez suivi, ça revient à taper [^a-zA-Z0-9\_] |
| **\t** | Indique une tabulation |
| **\n** | Indique une nouvelle ligne |
| **\r** | Indique un retour chariot |
| **\s** | Indique un espace blanc |
| **\S** | Indique ce qui n'est PAS un espace blanc (\t \n \r) |
| **.** | Indique n'importe quel caractère. Il autorise donc tout ! |

## Construire une REGEX complète :

**Le numéro de téléphone :**

Cd cours pour avoir étape par étape.

<p>

<?php

if (isset($\_POST['telephone']))

{

$\_POST['telephone'] = htmlspecialchars($\_POST['telephone']); // On rend inoffensives les balises HTML que le visiteur a pu entrer

if (preg\_match("#^0[1-68]([-. ]?[0-9]{2}){4}$#", $\_POST['telephone']))

{

echo 'Le ' . $\_POST['telephone'] . ' est un numéro <strong>valide</strong> !';

}

else

{

echo 'Le ' . $\_POST['telephone'] . ' n\'est pas valide, recommencez !';

}

}

?>

</p>

<form method="post">

<p>

<label for="telephone">Votre téléphone ?</label> <input id="telephone" name="telephone" /><br />

<input type="submit" value="Vérifier le numéro" />

</p>

</form>

**L’adresse e-mail :**

<p>

<?php

if (isset($\_POST['mail']))

{

$\_POST['mail'] = htmlspecialchars($\_POST['mail']); // On rend inoffensives les balises HTML que le visiteur a pu rentrer

if (preg\_match("#^[a-z0-9.\_-]+@[a-z0-9.\_-]{2,}\.[a-z]{2,4}$#", $\_POST['mail']))

{

echo 'L\'adresse ' . $\_POST['mail'] . ' est <strong>valide</strong> !';

}

else

{

echo 'L\'adresse ' . $\_POST['mail'] . ' n\'est pas valide, recommencez !';

}

}

?>

</p>

<form method="post">

<p>

<label for="mail">Votre mail ?</label> <input id="mail" name="mail" /><br />

<input type="submit" value="Vérifier le mail" />

</p>

</form>

**MySQL** comprend le REGEX mais POSIX dans lequel

* Il n’y a pas de délimiteur ni d’option, donc pas de parenthèse.
* Il n’y a pas de classe abrégée sauf le point.

On peut donc stocker les adresse IP des visiteurs et ensuite faire :

SELECT nom FROM visiteurs WHERE ip REGEXP '^84\.254(\.[0-9]{1,3}){2}$'

## Capture et remplacement :

Grace à ça, on pourra remplacer les adresses e-mail dans les messages des utilisateurs par une balise HTML mailto cliquable, on rendre les lien http cliquable aussi.

### Les parenthèses captures :

On travaille preg\_replace.

Quand on met des parenthèses, on créer une variable.

Exemple :

#\[b\](.+)\[/b\]#

Ici on cherche une chaîne mise entre balise [b] [/b]. Une variable $1 est créée. Si on avait plusieurs parenthèses, on aurait $2 …

Fonctionne ainsi :

<?php

$texte = preg\_replace('#\[b\](.+)\[/b\]#i', '<strong>$1</strong>', $texte);

?>

1. On lui donne en premier paramètre la regex. Rien de particulier, comme vous pouvez le constater, à part qu'il faut bien garder en tête que chaque parenthèse va créer une variable ($1, $2, etc.).     
      
   Ici, j'ai rajouté l'option « i » pour que le code fonctionne aussi avec des majuscules ([B][/B]).
2. Ensuite, et c'est là qu'est la nouveauté, on indique le texte de remplacement : « <strong>$1</strong> » (je vous rappelle que <strong> permet de mettre en gras en HTML).   
   Entre les balises HTML, j'ai mis $1. Cela signifie que ce qui se trouve dans la parenthèse capturante (entre [b] et [/b]) sera en fait entouré des balises <strong> !
3. Enfin, dernier paramètre, c'est le texte dans lequel on fait notre recherche / remplacement (ça, vous connaissez déjà).

Note : une variable $0 est **TOUJOURS** créée, elle contient tout le REGEX.

Pour qu’une parenthèse ne soit pas capturante, on indique ?: après l’ouverture de parenthèse.

#### Créer le bbCode :

**Création d’un parser :**

Avec le cas des [b], on peut utiiser trois option :

* i : pour accepter les majuscules comme les minuscules ([B] et [b]) ;
* s : pour que le « point » fonctionne aussi pour les retours à la ligne (pour que le texte puisse être en gras sur plusieurs lignes) ;
* U : le U majuscule est une option que vous ne connaissez pas et qui signifie « Ungreedy » (« pas gourmand »). Je vous passe les explications un peu complexes sur son fonctionnement, mais sachez que, grosso modo, ça ne marcherait pas correctement s'il y avait plusieurs [b] dans votre texte. Exemple :

« Ce texte est [b]important[/b], il faut me [b]comprendre[/b] ! » … sans activer l'option Ungreedy, la regex aurait voulu mettre en gras tout ce qu'il y a entre le premier [b] et le dernier [/b] (c'est-à-dire « important[/b], il faut me [b]comprendre »). En utilisant l'option « U », la regex s'arrêtera au premier [/b], et c'est ce qu'on veut.

Permet de mettre en gras et italique :

<?php

$texte = preg\_replace('#\[b\](.+)\[/b\]#isU', '<strong>$1</strong>', $texte);

$texte = preg\_replace('#\[i\](.+)\[/i\]#isU', '<em>$1</em>', $texte);

Pour insérer de la couleur au texte avec le bbCode :

<?php

$texte = preg\_replace('#\[color=(red|green|blue|yellow|purple|olive)\](.+)\[/color\]#isU', '<span style="color:$1">$2</span>', $texte);

?>

Ainsi, si on tape [color=blue]texte[/color], ça écrira texte en bleu. Vous pouvez essayer avec les autres couleurs aussi !

Transformer les liens en lien cliquable :

<?php

$texte = preg\_replace('#http://[a-z0-9.\_/-]+#i', '<a href="$0">$0</a>', $texte);

**Le bbCode au complet :**

<?php

if (isset($\_POST['texte']))

{

$texte = stripslashes($\_POST['texte']); // On enlève les slashs qui se seraient ajoutés automatiquement

$texte = htmlspecialchars($texte); // On rend inoffensives les balises HTML que le visiteur a pu rentrer

$texte = nl2br($texte); // On crée des <br /> pour conserver les retours à la ligne

// On fait passer notre texte à la moulinette des regex

$texte = preg\_replace('#\[b\](.+)\[/b\]#isU', '<strong>$1</strong>', $texte);

$texte = preg\_replace('#\[i\](.+)\[/i\]#isU', '<em>$1</em>', $texte);

$texte = preg\_replace('#\[color=(red|green|blue|yellow|purple|olive)\](.+)\[/color\]#isU', '<span style="color:$1">$2</span>', $texte);

$texte = preg\_replace('#http://[a-z0-9.\_/-]+#i', '<a href="$0">$0</a>', $texte);

// Et on affiche le résultat. Admirez !

echo $texte . '<br /><hr />';

}

?>

<p>

Bienvenue dans le parser du Site du Zéro !<br />

Nous avons écrit ce parser ensemble, j'espère que vous saurez apprécier de voir que tout ce que vous avez appris va vous être très utile !

</p>

<p>Amusez-vous à utiliser du bbCode. Tapez par exemple :</p>

<blockquote style="font-size:0.8em">

<p>

Je suis un gros [b]Zéro[/b], et pourtant j'ai [i]tout appris[/i] sur http://www.siteduzero.com<br />

Je vous [b][color=green]recommande[/color][/b] d'aller sur ce site, vous pourrez apprendre à faire ça [i][color=purple]vous aussi[/color][/i] !

</p>

</blockquote>

<form method="post">

<p>

<label for="texte">Votre message ?</label><br />

<textarea id="texte" name="texte" cols="50" rows="8"></textarea><br />

<input type="submit" value="Montre-moi toute la puissance des regex" />

</p>

</form>

# La POO

**L’objet est l’instance de la classe.**

Les flèches sont propres aux objets.

Exemple :

<?php

$maison1 = new *Maison*();

$maison2 = new *Maison*();

$maison1->nettoyer();

?>

On a la classe maison. On créé deux instances de classe : maison 1 et 2. ON applique à maison 1 la méthode de la classe.

## Créer une classe :

Il est recommandé de créer chaque classe dans un fichier séparé. Exemple maClasse.class.php.

<?php

class Membre

{

private $pseudo;

private $email;

private $signature;

private $actif;

public function getPseudo()

{

return $this->pseudo;

}

public function setPseudo($nouveauPseudo)

{

$this->pseudo = $nouveauPseudo;

}

}

## Getter et Setters :

<?php

public function setPseudo($nouveauPseudo)

{

// Vérifier si le nouveau pseudo n'est ni vide ni trop long

if (!empty($nouveauPseudo) AND strlen($nouveauPseudo) < 15)

{

// Ok, on change son pseudo

$this->pseudo = $nouveauPseudo;

}

}

?>

## Autres fonctions :

<?php

class Membre

{

public function envoyerEMail($titre, $message)

{

mail($this->email, $titre, $message);

}

public function bannir()

{

$this->actif = false;

$this->envoyerEMail('Vous avez été banni', 'Ne revenez plus !');

}

...

}

?>

On aura donc :

<?php

include\_once('Membre.class.php');

$membre = new *Membre*();

$membre->setPseudo('M@teo21');

echo $membre->getPseudo() . ', je vais te bannir !';

$membre->bannir();

?>

On utilise include\_once pour s’assurer que la page n’a pas déjà été inclue.

## Constructeurs et destructeurs :

### Les fonctions magiques :

Elles commencent toujours par un underscore.

**\_construct :**

Quand on fait new(), PHP recherche un \_construct, s’il existe, il l’applique, sinon il créé un objet vide.

Par exemple on aimerait appliquer le resultat de notre BDD aux objets :

<?php

class Membre

{

public function \_\_construct($idMembre)

{

// Récupérer en base de données les infos du membre

// SELECT pseudo, email, signature, actif FROM membres WHERE id = ...

// Définir les variables avec les résultats de la base

$this->pseudo = $donnees['pseudo'];

$this->email = $donnees['email'];

// etc.

}

...

?>

**\_destruct :**

Pour détruire un objet on utilise unset(). Si on ne le fait pas, l’objet sera détruit à la fin du développement.

<?php

unset($membre);

?>

S’il est créé dans index.php, il sera détruit à la fin de l’exécution de la page.

Dans notre exemple on n’a pas vraiment besoin de destructeur, on peut le mettre pour tester sa présence :

<?php

public function \_\_destruct()

{

echo 'Cet objet va être détruit !';

}

?>

### Les autres fonctions magiques :

[Documentation officielle](http://fr.php.net/manual/fr/language.oop5.magic.php)

## L’héritage :

Prenez un exemple très simple. On peut dire « Un administrateur est un membre », ou encore « Un modérateur est un membre ». Donc on peut faire un héritage : « La classe Admin hérite de Membre », « La classe Moderateur hérite de Membre ».

Exemple :

<?php

include\_once('Membre.class.php');

class Admin extends Membre

{

private $couleur;

public function setCouleur()

{

// ...

}

public function getCouleur()

{

// ...

}

}

?>

<?php

$membre = new *Membre*(31); // Contient un pseudo, une adresse e-mail...

$maitreDesLieux = new *Admin*(2); // Contient les mêmes données qu'un membre + la couleur

$membre->setPseudo('Arckintox'); // OK

$maitreDesLieux->setPseudo('M@teo21'); // OK

$membre->setCouleur('Rouge'); // Impossible (un membre n'a pas de couleur)

$maitreDesLieux->setCouleur('Rouge'); // OK

?>

## Droits d’accès et encapsulation :

### Le droit d’accès :

Il y a trois droits d'accès à connaître :

* public : tout le monde peut accéder à l'élément ;
* private : personne (à part la classe elle-même) n'a le droit d'accéder à l'élément ;
* protected : identique à private, sauf qu'un élément ayant ce droit d'accès dans une classe mère sera accessible aussi dans les classes filles.

<?php

class Membre

{

private $pseudo;

private $email;

private $signature;

private $actif;

public function getPseudo()

{

}

public function setPseudo($nouveauPseudo)

{

}

}

?>

<?php

$membre = new *Membre*(4);

$membre->setPseudo('M@teo21'); // OK car setPseudo est public

$membre->pseudo = 'M@teo21'; // Interdit car $pseudo est private

?>

**Toutes les variables d’une classe doivent être private ou protected.**

# Organiser son code selon l’architecture MVC

## MVC :

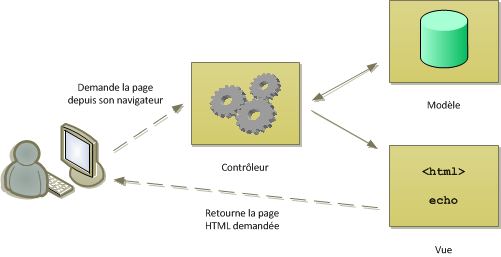
Certains problèmes reviennent tellement souvent que l’on a créé une bonne série de pratique appelé *design pattern*. **MVC** en est un : Modèle-Vue-Contrôleur.

On sépare donc le code en 3 parties :

* **Modèle** gère les données du site. On trouve les requêtes SQL.
* **Vue** gère l’affichage, beaucoup de HTML, quelques PHP simple.
* **Contrôleur :** presque uniquement du PHP.

Le contrôleur est donc le chef d’orchestre, il reçoit les requêtes et questionne les autres.

Il reçoit les requêtes, questionne le modèle, qui renvoie les réponses au contrôleur, qui transmet à l’affichage.



## Code du TP blog et ses défauts :

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Mon blog</title>

<link href="style.css" rel="stylesheet" />

</head>

<body>

<h1>Mon super blog !</h1>

<p>Derniers billets du blog :</p>

<?php

// Connexion à la base de données

try

{

$bdd = new *PDO*('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');

}

catch(Exception $e)

{

die('Erreur : '.$e->getMessage());

}

// On récupère les 5 derniers billets

$req = $bdd->query('SELECT id, titre, contenu, DATE\_FORMAT(date\_creation, \'%d/%m/%Y à %Hh%imin%ss\') AS date\_creation\_fr FROM billets ORDER BY date\_creation DESC LIMIT 0, 5');

while ($donnees = $req->fetch())

{

?>

<div class="news">

<h3>

<?php echo htmlspecialchars($donnees['titre']); ?>

<em>le <?php echo $donnees['date\_creation\_fr']; ?></em>

</h3>

<p>

<?php

// On affiche le contenu du billet

echo nl2br(htmlspecialchars($donnees['contenu']));

?>

<br />

<em><a href="commentaires.php?billet=<?php echo $donnees['id']; ?>">Commentaires</a></em>

</p>

</div>

<?php

} // Fin de la boucle des billets

$req->closeCursor();

?>

</body>

</html>

Pour l’améliorer, il faut le séparer en 3 parties : MVC.

## Amélioration :

A la racine on créé donc 3 répertoires :

* Modèle
* Vue
* Contrôleur

Ensuite dans chacun d’eux, on créé forums, blog, minichat.

### Le modèle :

On créé get\_billet.php dans modèle/blog :

<?php

function get\_billets($offset, $limit)

{

global $bdd;

$offset = (int) $offset;

$limit = (int) $limit;

$req = $bdd->prepare('SELECT id, titre, contenu, DATE\_FORMAT(date\_creation, \'%d/%m/%Y à %Hh%imin%ss\') AS date\_creation\_fr FROM billets ORDER BY date\_creation DESC LIMIT :offset, :limit');

$req->bindParam(':offset', $offset, *PDO*::PARAM\_INT);

$req->bindParam(':limit', $limit, *PDO*::PARAM\_INT);

$req->execute();

$billets = $req->fetchAll();

return $billets;

}

Je n'ai pas utilisé la méthode traditionnelle à laquelle vous avez été habitués pour transmettre les paramètres à la requête SQL. En effet, j'ai utilisé ici la fonction bindParam qui me permet de préciser que le paramètre est un entier (PDO$\colon\colon$PARAM\_INT). Cette méthode alternative est obligatoire dans le cas où les paramètres sont situés dans la clause LIMIT car il faut préciser qu'il s'agit d'entiers.

Ici on utilise fetchall qui met toutes les données dans un array.

On n’écrit pas ?> pour le modèle et le contrôleur.

### Le contrôleur :

C’est le chef d’orchestre. Dans controleur/blog on créé index.php qui aura la page d’accueil du site.

<?php

// On demande les 5 derniers billets (modèle)

include\_once('modele/blog/get\_billets.php');

$billets = get\_billets(0, 5);

// On effectue du traitement sur les données (contrôleur)

// Ici, on doit surtout sécuriser l'affichage

foreach($billets as $cle => $billet)

{

$billets[$cle]['titre'] = htmlspecialchars($billet['titre']);

$billets[$cle]['contenu'] = nl2br(htmlspecialchars($billet['contenu']));

}

// On affiche la page (vue)

include\_once('vue/blog/index.php');

Vous noterez qu'on opère sur les clés du tableau plutôt que sur $billet (sans s) directement. En effet, $billet est une copie du tableau $billets créée par le foreach. $billet n'existe qu'à l'intérieur du foreach, il est ensuite supprimé. Pour éviter les failles XSS, il faut agir sur le tableau utilisé à l'affichage, c'est-à-dire $billets.

### La vue :

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Mon blog</title>

<link href="vue/blog/style.css" rel="stylesheet" />

</head>

<body>

<h1>Mon super blog !</h1>

<p>Derniers billets du blog :</p>

<?php

foreach($billets as $billet)

{

?>

<div class="news">

<h3>

<?php echo $billet['titre']; ?>

<em>le <?php echo $billet['date\_creation\_fr']; ?></em>

</h3>

<p>

<?php echo $billet['contenu']; ?>

<br />

<em><a href="commentaires.php?billet=<?php echo $billet['id']; ?>">Commentaires</a></em>

</p>

</div>

<?php

}

?>

</body>

</html>

### Le contrôleur global du site

<?php

include\_once('modele/connexion\_sql.php');

if (!isset($\_GET['section']) OR $\_GET['section'] == 'index')

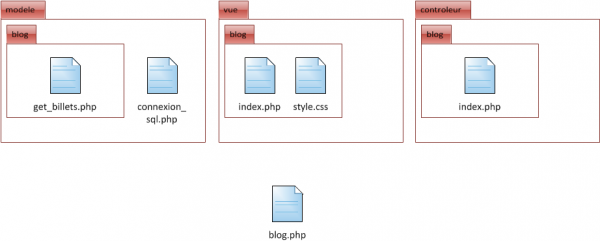
{

include\_once('controleur/blog/index.php');

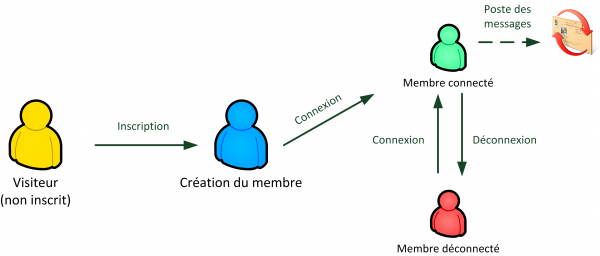
}

Il est conseillé de créer un contrôleur global qui se connecte à la BDD et le renvoie dans les autres fichiers.

### Résumé des fichiers :



# TP créer espace membre



Pour le MDP, il est recommandé d’utiliser un hachage, qui ne renvoie pas un mdp mais un code en hexadécimal. On utilisera donc **SHA1** qui est plus sûr que le md5.

## Les pages principales :

Pour une inscription rapide on se limite à 4 champs :

* Pseudo
* Mdp
* Mdp confirmation
* E-mail (expression régulière)

Dans les grandes lignes, le code sera donc ainsi :

<?php

// Vérification de la validité des informations

// Hachage du mot de passe

$pass\_hache = sha1($\_POST['pass']);

// Insertion

$req = $bdd->prepare('INSERT INTO membres(pseudo, pass, email, date\_inscription) VALUES(:pseudo, :pass, :email, CURDATE())');

$req->execute(array(

'pseudo' => $pseudo,

'pass' => $pass\_hache,

'email' => $email));

On peut aussi rajouter un système de captcha.

On peut aussi rajouter une checkbox de connexion auto pour générer deux coockies : le pseudo et le mdp.

<?php

// Hachage du mot de passe

$pass\_hache = sha1($\_POST['pass']);

// Vérification des identifiants

$req = $bdd->prepare('SELECT id FROM membres WHERE pseudo = :pseudo AND pass = :pass');

$req->execute(array(

'pseudo' => $pseudo,

'pass' => $pass\_hache));

$resultat = $req->fetch();

if (!$resultat)

{

echo 'Mauvais identifiant ou mot de passe !';

}

else

{

session\_start();

$\_SESSION['id'] = $resultat['id'];

$\_SESSION['pseudo'] = $pseudo;

echo 'Vous êtes connecté !';

}

On peut centraliser les connexions avec FB, Google .. il faut renseigner un script. Pour plus d’infos : [regarder cette page.](decouvrez-les-solutions-cms.pdf)

Pour se déconnecter :

<?php

session\_start();

// Suppression des variables de session et de la session

$\_SESSION = array();

session\_destroy();

// Suppression des cookies de connexion automatique

setcookie('login', '');

setcookie('pass\_hache', '');

# Mettre son site en ligne

Veuillez noter qu'il faut que votre page d'accueil s'appelleindex.html. C'est la page qui sera chargée lorsqu'un nouveau visiteur arrivera sur votre site.

Accéder à la BDD. :

<?php

$bdd = new *PDO*('mysql:host=sql.hebergeur.com;dbname=mabase', 'pierre.durand', 's3cr3t');

?>

# Coder proprement

* Noms de variables clairs
* Indenter le code
* Ajouter des commentaires d’étapes

# Créer un dossier d’admin du site

On peut mettre les codes sources dans un dossier admin et limité l’accès avec pseudo et mdp.

## Créer le dossier htaccess :

AuthName "Page d'administration protégée"

AuthType Basic

AuthUserFile "/home/site/www/admin/.htpasswd"

Require valid-user

Ce n’est ni du HTML ni du PHP, ce sont des instructions serveurs.

En effet, la plupart du temps, cela s'avère délicat à trouver car cela dépend du serveur. Heureusement, il existe une fonction PHP qui va beaucoup nous aider : realpath. Cette fonction donne le chemin absolu vers le fichier que vous indiquez. Vous allez donc faire comme suit pour trouver le chemin absolu.

Créez un fichier appelé chemin.php.

Inscrivez juste cette ligne de code à l'intérieur :

<?php echo realpath('chemin.php'); ?>

Envoyez ce fichier sur votre serveur avec votre logiciel FTP, et placez-le dans le dossier que vous voulez protéger.

Ouvrez votre navigateur et allez voir ce fichier PHP. Il vous donne le chemin absolu, par exemple dans mon cas :

/home/site/www/admin/chemin.php

Copiez ce chemin dans votre .htaccess, et remplacez le chemin.php par .htpasswd, ce qui nous donne au final par exemple :

/home/site/www/admin/.htpasswd

Supprimez le fichier chemin.php de votre serveur, il ne nous sert plus à rien maintenant qu'il nous a donné le chemin absolu.

## Créer le .htpasswd :

Il est de la forme : login :mdpCrypté. Exemple :

mateo21:$1$MEqT//cb$hAVid.qmmSGFW/wDlIfQ81

ptipilou:$1$/lgP8dYa$sQNXcCP47KhP1sneRIZoO0

djfox:$1$lT7nqnsg$cVtoPfe0IgrjES7Ushmoy.

vincent:$1$h4oVHp3O$X7Ejpn.uuOhJRkT3qnw3i0

Pour crypter son mdp :

<?php

if (isset($\_POST['login']) AND isset($\_POST['pass']))

{

$login = $\_POST['login'];

$pass\_crypte = crypt($\_POST['pass']); // On crypte le mot de passe

echo '<p>Ligne à copier dans le .htpasswd :<br />' . $login . ':' . $pass\_crypte . '</p>';

}

else // On n'a pas encore rempli le formulaire

{

?>

<p>Entrez votre login et votre mot de passe pour le crypter.</p>

<form method="post">

<p>

Login : <input type="text" name="login"><br />

Mot de passe : <input type="text" name="pass"><br /><br />

<input type="submit" value="Crypter !">

</p>

</form>

<?php

}

?>

Au final, on doit placer nos deux fichiers au même endroit que pour admin.php