PHP

**Table des matières**

[Premiers pas avec PHP 5](#_Toc467680256)

[Forme d’une balise PHP : 5](#_Toc467680257)

[Echo : 5](#_Toc467680258)

[Les commentaires : 6](#_Toc467680259)

[Inclure des portions de page 6](#_Toc467680260)

[Les variables 7](#_Toc467680261)

[Les types de variables : 7](#_Toc467680262)

[Déclaration de variable : 7](#_Toc467680263)

[Afficher et concaténer des variables : 7](#_Toc467680264)

[Les opérations : 8](#_Toc467680265)

[Les conditions 9](#_Toc467680266)

[If … else : 9](#_Toc467680267)

[Conditions multiples : 10](#_Toc467680268)

[Le switch-case : 11](#_Toc467680269)

[Les ternaires : 12](#_Toc467680270)

[Les boucles 12](#_Toc467680271)

[Le while : 12](#_Toc467680272)

[Le for : 12](#_Toc467680273)

[Les fonctions 13](#_Toc467680274)

[Quelques fonctions utiles : 13](#_Toc467680275)

[Strlen() : 13](#_Toc467680276)

[Str\_replace() : 13](#_Toc467680277)

[Str\_shuffle() : 13](#_Toc467680278)

[Strlower() : 14](#_Toc467680279)

[Récupérer les dates : 14](#_Toc467680280)

[Créer ses propres fonctions : 15](#_Toc467680281)

[Les tableaux 15](#_Toc467680282)

[Les tableaux numérotés : 15](#_Toc467680283)

[Récupérer une valeur : 16](#_Toc467680284)

[Le tableau associatif : 16](#_Toc467680285)

[Récupérer une valeur : 17](#_Toc467680286)

[Parcourir tout le tableau : for, foreach, print\_r : 17](#_Toc467680287)

[La boucle for : 17](#_Toc467680288)

[Foreach : 17](#_Toc467680289)

[Print\_r : 18](#_Toc467680290)

[Recherche dans un tableau : 18](#_Toc467680291)

[Vérifier que la clé existe dans l’array () : 18](#_Toc467680292)

[Vérifier qu’une valeur existe dans array() : 19](#_Toc467680293)

[Récupérer la clé d’une valeur : 19](#_Toc467680294)

[Transmettre des données dans l’URL 20](#_Toc467680295)

[Envoyer des données : 20](#_Toc467680296)

[Récupérer les données : 20](#_Toc467680297)

[Contrôler les valeurs des paramètres : 21](#_Toc467680298)

[Transmettre des données avec les formulaires 22](#_Toc467680299)

[Créer la base du formulaire : 22](#_Toc467680300)

[Petites zones de texte : 22](#_Toc467680301)

[Les grandes zones de texte : 23](#_Toc467680302)

[Les listes déroulantes : 23](#_Toc467680303)

[Les checkbox : 23](#_Toc467680304)

[Les champs cachés : 24](#_Toc467680305)

[La faille XSS : 24](#_Toc467680306)

[L’envoi des fichiers : 24](#_Toc467680307)

[Variable superglobale, session et cookie 26](#_Toc467680308)

[Les variables superglobales : 26](#_Toc467680309)

[Les sessions : 27](#_Toc467680310)

[Les cookies : 27](#_Toc467680311)

[Afficher un cookie : 28](#_Toc467680312)

[Modifier un cookie : 28](#_Toc467680313)

[Lire et écrire dans un fichier 29](#_Toc467680314)

[Autoriser l’écriture de fichier : 29](#_Toc467680315)

[Ouvrir et fermer un fichier : 29](#_Toc467680316)

[Lire et écrire dans un fichier : 30](#_Toc467680317)

[Lire un fichier : 30](#_Toc467680318)

[Ecrire dans un fichier : 30](#_Toc467680319)

[Les bases de données 31](#_Toc467680320)

[PhpMyAdmin 31](#_Toc467680321)

[Créer une table : 31](#_Toc467680322)

[Modifier une table : 32](#_Toc467680323)

[Lire des données SQL 32](#_Toc467680324)

[Se connecter à la base de données en PHP : 32](#_Toc467680325)

[Se connecter à mysql avec PDO : 32](#_Toc467680326)

[Tester la présence d’erreur : 33](#_Toc467680327)

[Récupérer les données : 33](#_Toc467680328)

[Filtrer les champs : 35](#_Toc467680329)

[Where : 36](#_Toc467680330)

[ORDER BY : 36](#_Toc467680331)

[LIMIT : 37](#_Toc467680332)

[Construire des requêtes en fonction de variables : 37](#_Toc467680333)

[Avec des marqueurs « ? » : 38](#_Toc467680334)

[Traquer les erreurs : 39](#_Toc467680335)

[Ecrire des données 39](#_Toc467680336)

[INSERT : ajoute des données : 39](#_Toc467680337)

[Insertion avec des requêtes séparées : 40](#_Toc467680338)

[Update pour modifier des valeurs : 40](#_Toc467680339)

[En requête séparée : 40](#_Toc467680340)

[DELETE : supprimer des données 41](#_Toc467680341)

# 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 :

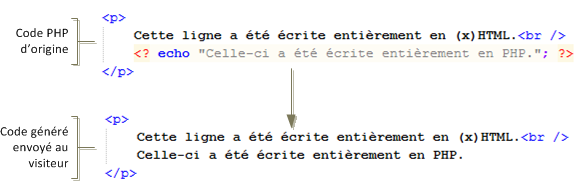
<?php echo "Ceci est du <strong>texte</strong>"; ?>

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');

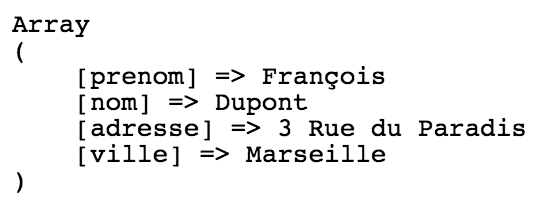
echo '<pre>';

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;

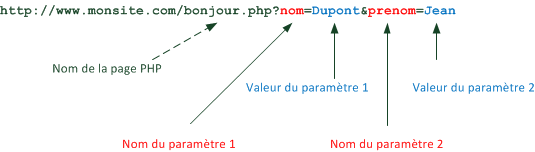
?>

# 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 eput 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 $çSESSION[‘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

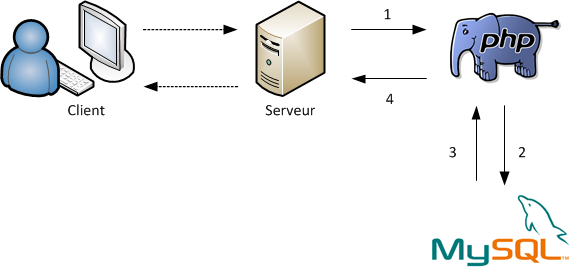
fclose($monfichier);

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>

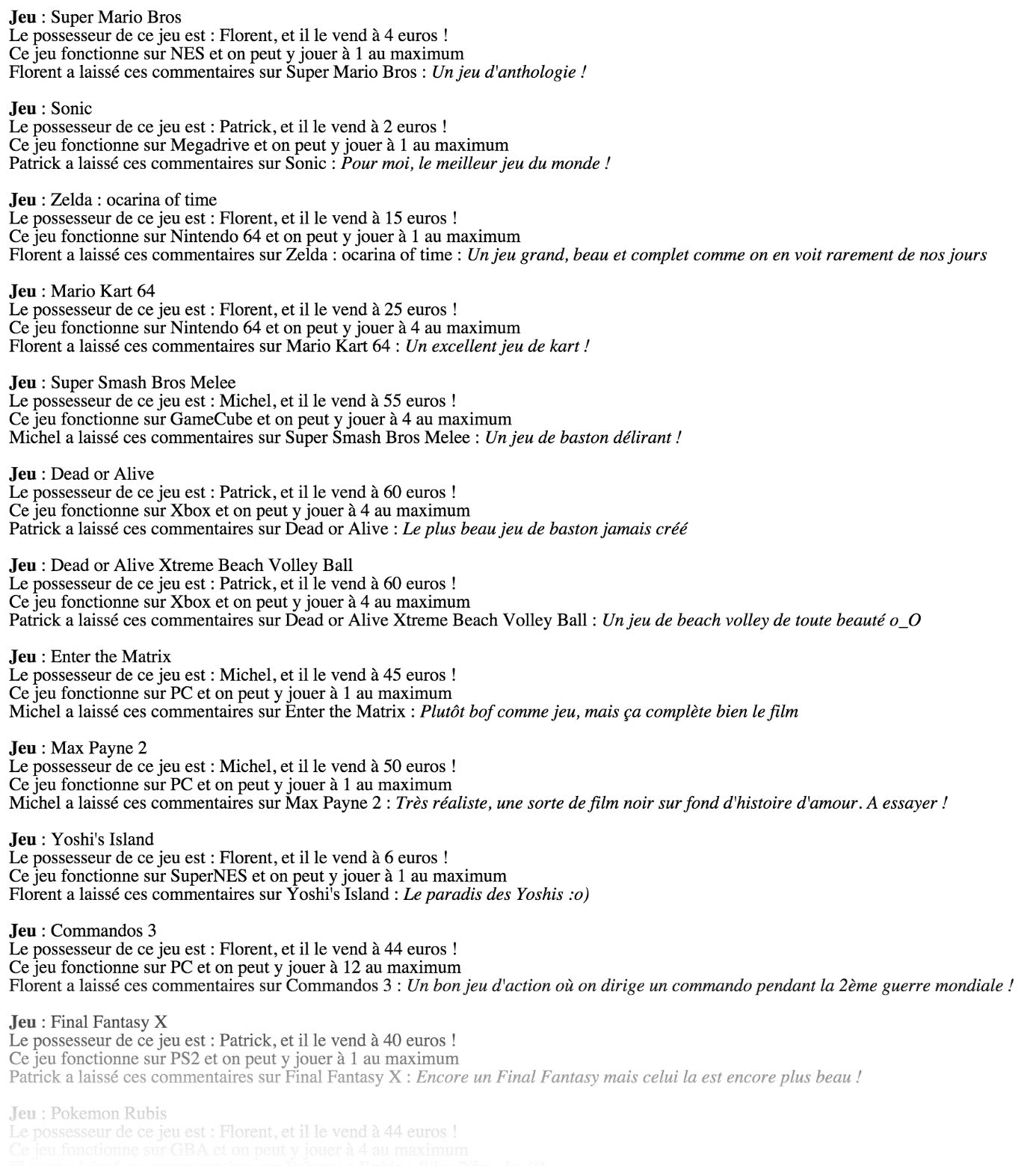
<?php

}

$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€.