



SOMMAIRE

LES SITES DYNAMIQUES	5
Comment fonctionne un site web ?	6
Les Concurrents de PHP	7
Les Concurrents de MYSQL	7
Outils de Travail.....	8
POINT VIRGULE.....	8
Commentaires :	8
Résumé :	9
INCLUDE DES PORTIONS DE PAGE	9
Résumé :	10
Type de donné	10
Déclarer une constante	10
Concaténation	10
Résumé	11
IF :	11
Switch :.....	11
Condition ternaire :.....	12
Résumé :	12
BOUCLE :	12
While :	12
For :	13
Résumé :	13
FONCTIONS PHP	13
Fichiers	13
Chaine de caractères :	13
Date et heures :.....	14

Fonction de codage	14
Fonction personnalisé	15
Résumé :	15
LES TABLEAUX.....	15
Les tableaux numérotés :	15
Les tableaux associatifs :	16
Parcourir avec FOR.....	16
Parcourir avec Foreach.....	16
Afficher rapidement un array avec print_r	17
Rechercher dans un tableau.....	17
Résumé :	18
Transmettre des données avec l'URL.....	18
Envoyer le message	18
Extraire	18
Tester les paramètres :	18
Résumé :	19
Révision : les formulaires.....	19
La faille XSS : cross-site Scripting	22
Pour l'échaper	22
Upload des fichiers :	22
La variable \$_FILES['monfichier'].....	22
Résumé :	24
Variables super globales, sessions et cookies.....	24
Les variables super globales	24
Fonctionnement des sessions	25
Les cookies.....	26
Qu'est-ce qu'un cookie?	26
Créer un cookie qui va expirer après un an :	27
Modifier un cookie : écraser l'ancienne cookie	27
Résumé :	28
Lire et écrire dans un fichier :.....	28
Ouvrir et fermer un fichier :	28
Mode :	28
Procédure :	29
Résumé :	29
Stocker des informations dans une base de données :	30
PHP et MYSQL.....	30
Type de donné MYSQL :	30
Résumé :	30
Lire des données avec PHP	31

Comment se connecte-t-on à la base de données en PHP?.....	31
Afficher les résultat d'un requête :	31
Mot clé LIMIT dans MYSQL.....	32
Les requêtes préparées :.....	32
Traquer les erreurs :	33
Case d'une requête préparée :	33
Résumé :	33
Ecrire des données.....	33
Insertion.....	33
Insertion de données variables grâce à une requête préparée	34
Mise à jour	34
Mise à jour avec une requête préparée	34
Suppression	34
Résumé :	35
Fonctions SQL.....	35
Utiliser les alias	35
Résumé :	35
TYPE DE DONNEE MYSQL.....	36
Résumé :	37
CREER DES IMAGES EN PHP	37
Écrire du texte.....	37
ImageSetPixel.....	39
ImageLine	39
ImageEllipse	39
ImageRectangle.....	40
Rendre une image transparente.....	40
Mélanger deux images :.....	40
Redimensionner une image :	41
Résumé :	42
LES EXPRESSIONS RÉGULIÈRES :.....	42
Les types de REGEX	43
Les fonctions PCRE :	43
Le symbole OU	43
Début et fin de chaîne	43
Des classes simples	43
Les intervalles de classe.....	44
La négation.....	44
Les quantificateurs	44
Les accolades :.....	44
Résumé :	45

Les métacaractères	45
Echapper un métacaractères	45
Le cas des classes :	46
Les classes abrégées(les raccourcis)	46
Des regex avec MySQL.....	47
Capture et remplacement	47
Échapper une parenthèse :.....	48
Créer votre bbCode :	48
Résumé :	49
Mémento Regex.....	50
Base :.....	50
Classes de caractères :.....	50
Quantificateurs :.....	50
Métacarctéres :	51
Classes abrégées	51
Capture et remplacement	51
Options.....	51
PHP ET LA POO	51
Créer une classe	52
Getters et Setters	52
Fonctions :	53
Instanciation.....	53
Constructeur : __construct	53
Détruire un objet :	53
L'héritage :.....	54
L'encapsulation :	54
Résumé :	54
ORGANISER SON CODE SELON L'ARCHITECTURE MVC	54
Le modèle.....	55
Le Contrôleur	56
La Vue	56
Utiliser Framework.....	56
Résumé :	57
FONCTION UTILE :	57

LES SITES DYNAMIQUES

- **Les sites dynamiques** : plus complexes, ils utilisent d’autres langages en plus de HTML et CSS, tels que PHP et MySQL. Le contenu de ces sites web est dit « dynamique » parce qu’il peut changer sans l’intervention du webmaster ! La plupart des sites web que vous visitez aujourd’hui, y compris le Site du Zéro, sont des sites dynamiques. Le seul prérequis pour apprendre à créer ce type de sites est de déjà savoir réaliser des sites statiques en HTML et CSS¹.

Comment fonctionne un site web ?

Lorsque vous voulez visiter un site web, vous tapez son adresse dans votre navigateur web, que ce soit Mozilla Firefox, Internet Explorer, Opera, Safari ou un autre. Mais ne vous êtes-vous jamais demandé comment faisait la page web pour arriver jusqu'à vous ?

Il faut savoir qu'Internet est un réseau composé d'ordinateurs. Ceux-ci peuvent être classés en deux catégories.

- **Les clients** : ce sont les ordinateurs des internautes comme vous. Votre ordinateur fait donc partie de la catégorie des clients. Chaque client représente un visiteur d'un site web. Dans les schémas qui vont suivre, l'ordinateur d'un client sera représenté par l'image 1.2.
- **Les serveurs** : ce sont des ordinateurs puissants qui stockent et délivrent des sites web aux internautes, c'est-à-dire aux clients. La plupart des internautes n'ont jamais vu un serveur de leur vie. Pourtant, les serveurs sont indispensables au bon fonctionnement du Web. Sur les prochains schémas, un serveur sera représenté par l'image de la figure 1.3.

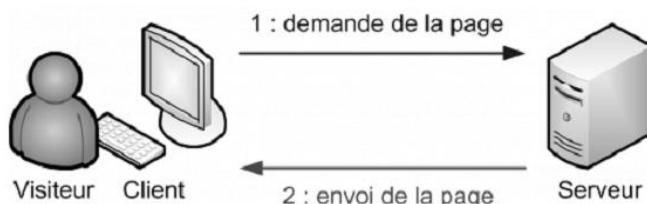


FIGURE 1.4 – Transferts avec un site statique



FIGURE 1.5 – Transfert avec un site dynamique



FIGURE 1.7 – PHP décide ce qui va être affiché sur la page web envoyée au visiteur

Les concurrents de PHP

Parmi les concurrents de PHP, on peut citer les suivants :

- **ASP .NET** : conçu par Microsoft, il exploite le framework³ .NET bien connu des développeurs C#. Ce langage peut être intéressant si vous avez l'habitude de développer en C# .NET et que vous ne voulez pas être dépayrés.
- **Ruby on Rails** : très actif, ce framework s'utilise avec le langage Ruby et permet de réaliser des sites dynamiques rapidement en suivant certaines conventions.
- **Django** : il est similaire à Ruby on Rails, mais il s'utilise en langage Python.
- **Java et les JSP (Java Server Pages)** : plus couramment appelé « **JEE** », il est particulièrement utilisé dans le monde professionnel. Il demande une certaine rigueur. La mise en place d'un projet JEE est traditionnellement un peu plus longue et plus lourde mais le système est apprécié des professionnels et des institutions⁴.

Les Concurrents de MYSQL

Les concurrents de MySQL

En ce qui concerne les bases de données, le choix est là encore très vaste. Cependant, alors que PHP et ses concurrents sont la plupart du temps libres et gratuits, ce n'est pas le cas de la plupart des SGBD.

Parmi les concurrents de MySQL, je vous conseille de connaître (au moins de nom) les suivants :

- **Oracle** : c'est le SGBD le plus célèbre, le plus complet et le plus puissant. Il est malheureusement payant (et cher), ce qui le réserve plutôt aux entreprises qui l'utilisent déjà massivement. Il existe cependant des versions gratuites d'Oracle, notamment pour ceux qui veulent apprendre à s'en servir.
- **Microsoft SQL Server** : édité par Microsoft, on l'utilise souvent en combinaison avec ASP .NET, bien qu'on puisse l'utiliser avec n'importe quel autre langage. Il est payant, mais il existe des versions gratuites limitées.
- **PostgreSQL** : il s'agit d'un SGBD libre et gratuit comme MySQL, qui propose des fonctionnalités plus avancées. Parfois comparé à Oracle, il lui reste cependant du chemin à parcourir. Il dispose d'une communauté un peu moins importante que MySQL et Oracle. Le Site du Zéro utilise PostgreSQL.

En résumé

- Pour créer des sites web dynamiques, nous devons installer des outils qui transformeront notre ordinateur en serveur afin de pouvoir tester notre site.
- Les principaux outils dont nous avons besoin sont :
 - Apache : le serveur web ;
 - PHP : le programme qui permet au serveur web d'exécuter des pages PHP ;
 - MySQL : le logiciel de gestion de bases de données.
- Bien qu'il soit possible d'installer ces outils séparément, il est plus simple pour nous d'installer un paquetage tout prêt : WAMP sous Windows, MAMP sous Mac OS X ou XAMPP sous Linux.
- Il est conseillé d'utiliser un éditeur de texte qui colore le code source comme Notepad++ pour programmer convenablement en PHP.

La forme d'une balise PHP :

```
<?php  
/* Le code PHP se met ici  
Et ici  
Et encore ici */  
?>
```

POINT VIRGULE



Il ne faut jamais oublier le point-virgule à la fin d'une instruction. Si jamais ça arrive, vous aurez le message d'erreur : « Parse Error ». Notez que ça plante uniquement si votre code PHP fait plus d'une ligne (ça sera tout le temps le cas). Prenez donc l'habitude de toujours mettre un « ; » à la fin des instructions.

Commentaires :

Les commentaires monolignes

Pour indiquer que vous écrivez un commentaire sur une seule ligne, vous devez taper deux slashes : « // ». Tapez ensuite votre commentaire. Un exemple ?

```
<?php  
echo "J'habite en Chine."; // Cette ligne indique où j'habite  
  
// La ligne suivante indique mon âge  
echo "J'ai 92 ans.";  
?>
```

Les commentaires multilignes

Ce sont les plus pratiques si vous pensez écrire un commentaire sur plusieurs lignes, mais on peut aussi s'en servir pour écrire des commentaires d'une seule ligne. Il faut commencer par écrire /* puis refermer par */ :

```
<?php
/* La ligne suivante indique mon âge
Si vous ne me croyez pas...
... vous avez raison ;o) */
echo "J'ai 92 ans.";
?>
```

Résumé :

En résumé

- Les pages web contenant du PHP ont l'extension .php.
- Une page PHP est en fait une simple page HTML qui contient des instructions en langage PHP.
- Les instructions PHP sont placées dans une balise <?php ?>.
- Pour afficher du texte en PHP, on utilise l'instruction echo.
- Il est possible d'ajouter des commentaires en PHP pour décrire le fonctionnement du code. On utilise pour cela les symboles // ou /* */.

INCLUDE DES PORTIONS DE PAGE

The diagram illustrates the inclusion of external PHP files into a main page. The main page (index.php) includes four files: entete.php, menus.php, corps.php, and pied_de_page.php. Arrows point from the include statements in the main page to their respective files.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/1999/xhtml" xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
<head>
    <title>Mon super site</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>
    <?php include("entete.php"); ?>
    <?php include("menus.php"); ?>
    <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
        </p>
    </div>
    <?php include("pied_de_page.php"); ?>
</body>
</html>
```

entete.php

```
<div id="en_tete">
</div>
```

menus.php

```
<div id="menu">
    <div class="element_menu">
        <h3>Titre menu</h3>
        <ul>
            <li><a href="page1.html">Lien</a></li>
            <li><a href="page2.html">Lien</a></li>
            <li><a href="page3.html">Lien</a></li>
        </ul>
    </div>
</div>
```

pied_de_page.php

```
<div id="pied_de_page">
    <p>Copyright moi, tous droits réservés</p>
</div>
```

Résumé :

En résumé

- Une page PHP peut inclure une autre page (ou un morceau de page) grâce à l'instruction `include`.
- L'instruction `include` sera remplacée par le contenu de la page demandée.
- Cette technique, très simple à mettre en place, permet par exemple de placer les menus de son site dans un fichier `menus.php` que l'on inclura dans toutes les pages. Cela permet de centraliser le code des menus alors qu'on était auparavant obligé de le copier dans chaque page sur nos sites statiques en HTML et CSS!

Type de donné

Type de données	Exemple de valeur
string	"Du texte"
int	42
float	14.738
bool	true <input checked="" type="checkbox"/> false <input type="checkbox"/>
NULL	X

FIGURE 5.1 – Types de données

Déclarer une constante

`Define('NOM','valeur');`

EXEMPLE :

```
<?php  
$age_du_visiteur = 17;  
?>
```

Concaténation

```
<?php  
$age_du_visiteur = 17;  
echo "Le visiteur a $age_du_visiteur ans";  
?>
```

```
<?php  
$age_du_visiteur = 17;  
echo 'Le visiteur a ' . $age_du_visiteur . ' ans';  
?>
```

En résumé

- Une variable est une petite information qui reste stockée en mémoire le temps de la génération de la page PHP. Elle a un nom et une valeur.
- Il existe plusieurs types de variables qui permettent de stocker différents types d'informations : du texte (**string**), des nombres entiers (**int**), des nombres décimaux (**float**), des booléens pour stocker vrai ou faux (**bool**), etc.
- En PHP, un nom de variable commence par le symbole dollar : **\$age** par exemple.
- La valeur d'une variable peut être affichée avec l'instruction **echo**.
- Il est possible de faire des calculs mathématiques entre plusieurs variables : addition, soustraction, multiplication... .

CONDITIONS

IF :

```
<?php
if ($sexe == "fille" OR $sexe == "garçon")
{
    echo "Salut Terrien !";
}
else
{
    echo "Euh, si t'es ni une fille ni un garçon, t'es quoi alors ?";
}
?>
```

Switch :

```
<?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 Zér0 !!!";
        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 20:
        echo "Excellent travail, c'est parfait !";
        break;

    default:
        echo "Désolé, je n'ai pas de message à afficher pour cette note";
}
?>

```

Condition ternaire :

```

<?php
$age = 24;

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

```

Résumé :

En résumé

- Les conditions permettent à PHP de prendre des décisions en fonction de la valeur des variables.
- La forme de condition la plus courante est `if... elseif... else` qui signifie « si »... « sinon si »... « sinon ».
- On peut combiner des conditions avec les mots-clés `AND` (« et ») et `OR` (« ou »).
- Si une condition comporte de nombreux `elseif`, il peut être plus pratique d'utiliser `switch`, une autre forme de condition.
- Les ternaires sont des conditions condensées qui font un test sur une variable, et en fonction des résultats de ce test donnent une valeur à une autre variable. Elles sont cependant plus rarement utilisées.

BOUCLE :

While :

```

<?php
while ($continuer_boucle == true)
{
    // instructions à exécuter dans la boucle
}
?>

```



Il faut TOUJOURS s'assurer que la condition sera fausse au moins une fois. Si elle ne l'est jamais, alors la boucle s'exécutera à l'infini ! PHP refuse normalement de travailler plus d'une quinzaine de secondes. Il s'arrêtera tout seul s'il voit que son travail dure trop longtemps et affichera un message d'erreur.

For :

```
<?php
for ($nombre_de_lignes = 1; $nombre_de_lignes <= 100; $nombre_de_lignes++)
{
    echo 'Ceci est la ligne n°' . $nombre_de_lignes . '<br />';
}
?>
```

Résumé :

En résumé

- Les boucles demandent à PHP de répéter des instructions plusieurs fois.
- Les deux principaux types de boucles sont :
 - **while** : à utiliser de préférence lorsqu'on ne sait pas par avance combien de fois la boucle doit être répétée ;
 - **for** : à utiliser lorsqu'on veut répéter des instructions un nombre précis de fois.
- L'incrémentation est une technique qui consiste à ajouter 1 à la valeur d'une variable. La décrémentation retire au contraire 1 à cette variable. On trouve souvent des incrémentations au sein de boucles **for**.

FONCTIONS PHP

Fichiers

File_exists(fichier) renvoie null si le fichier n'existe pas

Lire un Fichiers

\$var=File_get_contents(fichier)

Ecriture

File_put_contents(file,data)

Ouverture et affichage

Readfile(file)

Implode('
',\$data); // séparer les ligne d'un array par '
'

Chaine de caractères :

strlen : longueur d'une chaîne

Cette fonction retourne la longueur d'une chaîne de caractères, c'est-à-dire le nombre de lettres et de chiffres dont elle est constituée (espaces compris). Exemple :

```
<?php
$phrase = 'Bonjour les Zéros ! Je suis une phrase !';
$longueur = strlen($phrase);

echo 'La phrase ci-dessous comporte ' . $longueur . ' caractères :<br />' . $phr
→ ase;
?>
```

str_replace : rechercher et remplacer

str_replace remplace une chaîne de caractères par une autre. Exemple :

```
<?php  
$ma_variable = str_replace('b', 'p', 'bim bam boum');  
  
echo $ma_variable;  
?>
```

strtolower : écrire en minuscules

strtolower met tous les caractères d'une chaîne en minuscules.

```
<?php  
$chaine = 'COMMENT CA JE CRIE TROP FORT ???';  
$chaine = strtolower($chaine);  
  
echo $chaine;  
?>
```

Date et heures :

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

```
<?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 e  
st ' . $heure. ' h ' . $minute;  
?>
```

Fonction de codage

Utf8_encode(\$data)

Utf8_decode(\$data) //decode vers iso

Fonction personnalisé

```
<?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;
?>
```

Résumé :

En résumé

- Les fonctions sont des blocs de code qui exécutent des instructions en fonction de certains paramètres.
- Les fonctions ont généralement une entrée et une sortie. Par exemple, si on donne la valeur 4 à la fonction de calcul du cube, celle-ci renvoie 64 en sortie.
- PHP propose des centaines et des centaines de fonctions prêtes à l'emploi pour tous types de tâches : envoyer un e-mail, récupérer l'heure, crypter des mots de passe, etc.
- Si PHP ne propose pas la fonction dont on a besoin, il est possible de la créer avec le mot-clé `function`.

LES TABLEAUX

Les tableaux numérotés :

```
<?php
// La fonction array permet de créer un array
$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique', 'Benoît');
?>
```

```
<?php
$prenoms[0] = 'François';
$prenoms[1] = 'Michel';
$prenoms[2] = 'Nicole';
?>
```

```
<?php
$prenoms[] = 'François'; // Créera $prenoms[0]
$prenoms[] = 'Michel'; // Créera $prenoms[1]
$prenoms[] = 'Nicole'; // Créera $prenoms[2]
?>
```

Les tableaux associatifs :

```
<?php
// On crée notre array $coordonnees
$coordonnees = array (
    'prenom' => 'François',
    'nom' => 'Dupont',
    'adresse' => '3 Rue du Paradis',
    'ville' => 'Marseille');
?>
```

Clé	Valeur
prenom	François
nom	Dupont
adresse	3 Rue du Paradis
ville	Marseille

```
<?php
$coordonnees['prenom'] = 'François';
$coordonnees['nom'] = 'Dupont';
$coordonnees['adresse'] = '3 Rue du Paradis';
$coordonnees['ville'] = 'Marseille';
?>
```

PARCOURIR UN TABLEAU

Lorsqu'un tableau a été créé, on a souvent besoin de le parcourir pour savoir ce qu'il contient. Nous allons voir trois moyens d'explorer un array :

- la boucle **for** ;
- la boucle **foreach** ;
- la fonction **print_r** (utilisée principalement pour le débogage).

Parcourir avec 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.
}
?>
```

Parcourir avec Foreach

```
<?php
$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique', 'Benoît');

foreach($prenoms as $element)
{
    echo $element . '<br />'; // affichera $prenoms[0], $prenoms[1] etc.
}
?>
```

```
| <?php foreach($coordonnees as $cle => $element) ?>
```

À chaque tour de boucle, on disposera non pas d'une, mais de deux variables :

- **\$cle**, qui contiendra la clé de l'élément en cours d'analyse (« prenom », « nom », etc.) ;
- **\$element**, qui contiendra la valeur de l'élément en cours (« François », « Dupont », etc.).

Testons le fonctionnement avec un exemple :

```
<?php
$coordonnees = array (
    'prenom' => 'François',
    'nom' => 'Dupont',
    'adresse' => '3 Rue du Paradis',
    'ville' => 'Marseille');

foreach($coordonnees as $cle => $element)
{
    echo '[' . $cle . '] vaut ' . $element . '<br />';
}
?>
```

Afficher rapidement un array avec print_r

```
<?php
$coordonnees = array (
    'prenom' => 'François',
    'nom' => 'Dupont',
    'adresse' => '3 Rue du Paradis',
    'ville' => 'Marseille');

echo '<pre>';
print_r($coordonnees);
echo '</pre>';
?>
```

<pre> qui nous permet d'avoir un affichage plus correct.

REMARQUE :

Bien entendu, vous n'afficherez jamais des choses comme ça à vos visiteurs. On peut en revanche s'en servir pour le débogage, **pendant** la création du site, afin de voir rapidement ce que contient l'array.

Rechercher dans un tableau

- **array_key_exists** : pour vérifier si une clé existe dans l'array ;
- **in_array** : pour vérifier si une valeur existe dans l'array ;
- **array_search** : pour récupérer la clé d'une valeur dans l'array.

array_search fonctionne comme **in_array** : il travaille sur les valeurs d'un array. Voici ce que renvoie la fonction :

- si elle a trouvé la valeur, **array_search** renvoie la clé correspondante³ ;
- si elle n'a pas trouvé la valeur, **array_search** renvoie **false**.

Résumé :

En résumé

- Les tableaux (ou arrays) sont des variables représentées sous forme de tableau. Elles peuvent donc stocker de grandes quantités d'informations.
- Chaque ligne d'un tableau possède une clé (qui permet de l'identifier) et une valeur.
- Il existe deux types de tableaux :
 - les tableaux **numérotés** : chaque ligne est identifiée par une clé numérotée. La numérotation commence à partir de 0 ;
 - les tableaux **associatifs** : chaque ligne est identifiée par une courte chaîne de texte.
- Pour parcourir un tableau, on peut utiliser la boucle **for** que l'on connaît déjà, mais aussi la boucle **foreach** qui est dédiée aux tableaux.
- Il existe de nombreuses fonctions permettant de travailler sur des tableaux et notamment d'effectuer des recherches.

Transmettre des données avec l'URL

URL : Uniform Resource Locator

Envoyer le message

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

Extraire

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

Tester les paramètres :

```
<?php
if (isset($_GET['prenom']) AND isset($_GET['nom'])) // On a le nom et le prénom
{
    echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !';
}
else // Il manque des paramètres, on avertit le visiteur
{
    echo 'Il faut renseigner un nom et un prénom !';
}
?>
```

En résumé

- Une URL représente l'adresse d'une page web (commençant généralement par `http://`).
- Lorsqu'on fait un lien vers une page, il est possible d'ajouter des paramètres sous la forme `bonjour.php?nom=Dupont&prenom=Jean` qui seront transmis à la page.
- La page `bonjour.php` dans l'exemple précédent recevra ces paramètres dans un array nommé `$_GET` :
 - `$_GET['nom']` aura pour valeur « Dupont » ;
 - `$_GET['prenom']` aura pour valeur « Jean ».
- Cette technique est très pratique pour transmettre des valeurs à une page, mais il faut prendre garde au fait que le visiteur peut les modifier très facilement. Il ne faut donc pas faire aveuglément confiance à ces informations, et tester prudemment leur valeur avant de les utiliser.
- La fonction `isset()` permet de vérifier si une variable est définie ou non.
- Le transtypage est une technique qui permet de convertir une variable dans le type de données souhaité. Cela permet de s'assurer par exemple qu'une variable est bien un `int` (nombre entier).

Révision : les formulaires

La méthode

Il faut savoir qu'il existe plusieurs moyens d'envoyer les données du formulaire (plusieurs « méthodes »). Vous pouvez en employer deux.

- `get` : les données transiteront par l'URL comme on l'a appris précédemment. On pourra les récupérer grâce à l'array `$_GET`. Cette méthode est assez peu utilisée car on ne peut pas envoyer beaucoup d'informations dans l'URL¹.
- `post` : les données ne transiteront pas par l'URL, l'utilisateur ne les verra donc pas passer dans la barre d'adresse. Cette méthode permet d'envoyer autant de données que l'on veut, ce qui fait qu'on la privilégie le plus souvent. Néanmoins, les données ne sont pas plus sécurisées qu'avec la méthode GET et il faudra toujours vérifier si tous les paramètres sont bien présents et valides, comme on l'a fait dans le chapitre précédent. **On ne doit pas plus faire confiance aux formulaires qu'aux URL.**

La cible

L'attribut `action` sert à définir la page appelée par le formulaire. C'est cette page qui recevra les données du formulaire et qui sera chargée de les traiter.

Imaginons le schéma de la figure 11.1.

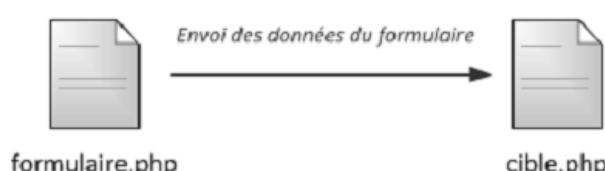


FIGURE 11.1 – Appel de la page cible par le formulaire

Les grandes zones de texte

La grande zone de texte² ressemble à la figure 11.4.

Comment pensez-vous que je pourrais améliorer mon site ?
Difficile d'améliorer la perfection non ?
Enfin, moi ce que j'en dis...
A toi de voir !

FIGURE 11.4 – Une grande zone de texte

```
<textarea name="message" rows="8" cols="45">  
Votre message ici.  
</textarea>
```

Là encore, on a un attribut `name` qui va définir le nom de la variable qui sera créée dans `cible.php`. Dans notre cas, ce sera la variable `$_POST['message']`.

La liste déroulante

La figure 11.5 est une liste déroulante.

Dans quel pays habitez-vous ?

- France
- France
- Espagne
- Italie
- Royaume-Uni
- Canada
- Etats-Unis
- Chine
- Japon

FIGURE 11.5 – Une liste déroulante

Ici, une variable `$_POST['choix']` sera créée, et elle contiendra le choix qu'a fait l'utilisateur. S'il a choisi « Choix 3 », la variable `$_POST['choix']` sera égale au `value` correspondant, c'est-à-dire `choix3`.

Les cases à cocher

La figure 11.6 représente une série de cases à cocher.

- Frites
- Steak haché
- Epinards
- Huitres

FIGURE 11.6 – Cases à cocher

On utilisera le code suivant pour afficher des cases à cocher :

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

Là encore, on donne un nom à la case à cocher via l'attribut `name` (ici : « case »). Ce nom va générer une variable dans la page cible, par exemple `$_POST['case']`.

- Si la case a été cochée, alors `$_POST['case']` aura pour valeur « on ».
- Si elle n'a pas été cochée, alors `$_POST['case']` n'existera pas. Vous pouvez faire un test avec `isset($_POST['case'])` pour vérifier si la case a été cochée ou non.

Les boutons d'option

Les boutons d'option fonctionnent par groupes de deux minimum. Vous trouverez un exemple sur la figure 11.7.

Aimez-vous les frites ? Oui Non

FIGURE 11.7 – Boutons d'option

Le code correspondant à cet exemple est le suivant :

```
Aimez-vous les frites ?  
<input type="radio" name="frites" value="oui" id="oui" checked="checked" /> <lab  
→ el for="oui">Oui</label>  
<input type="radio" name="frites" value="non" id="non" /> <label for="non">Non</  
→ label>
```

Dans la page cible, une variable `$_POST['frites']` sera créée. Elle aura la valeur du bouton d'option choisi par le visiteur, issue de l'attribut `value`. Si on aime les frites, alors on aura `$_POST['frites'] = 'oui'`.

Il faut bien penser à renseigner l'attribut `value` du bouton d'option car c'est lui qui va déterminer la valeur de la variable.

Les champs cachés

Les champs cachés constituent un type de champ à part. En quoi ça consiste ? C'est un code dans votre formulaire qui n'apparaîtra pas aux yeux du visiteur, mais qui va quand même créer une variable avec une valeur. On peut s'en servir pour transmettre des informations fixes.

Je m'explique : supposons que vous ayez besoin de « retenir » que le pseudo du visiteur est « Mateo21 ». Vous allez taper ce code :

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

À l'écran, sur la page web on ne verra rien. Mais dans la page cible, une variable `$_POST['pseudo']` sera créée, et elle aura la valeur « Mateo21 » !

C'est apparemment inutile, mais vous verrez que vous en aurez parfois besoin.

La faille XSS : cross-site Scripting

La faille XSS (pour **cross-site scripting**) est vieille comme le monde⁵ et on la trouve encore sur de nombreux sites web, même professionnels ! C'est une technique qui consiste à injecter du code HTML contenant du JavaScript dans vos pages pour le faire exécuter à vos visiteurs.

Pour l'échaper

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

Le code HTML qui en résultera sera propre et protégé car les balises HTML insérées par le visiteur auront été échappées :

```
|<p>Je sais comment tu t'appelles, hé hé. Tu t'appelles &lt;strong&gt;Badaboum&lt;  
|→ ;/strong&gt; !</p>
```



Si vous préférez retirer les balises HTML que le visiteur a tenté d'envoyer plutôt que de les afficher, utilisez la fonction `strip_tags`.

Upload des fichiers :

```
|<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>
```

La variable `$_FILES['monfichier']`

Variable	Signification
<code>\$_FILES['monfichier']['name']</code>	Contient le nom du fichier envoyé par le visiteur.
<code>\$_FILES['monfichier']['type']</code>	Indique le type du fichier envoyé. Si c'est une image gif par exemple, le type sera <code>image/gif</code> .
<code>\$_FILES['monfichier']['size']</code>	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.
<code>\$_FILES['monfichier']['tmp_name']</code>	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).
<code>\$_FILES['monfichier']['error']</code>	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.

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, $extensions_autorisees))
        {
            // On peut valider le fichier et le stocker
            move_uploaded_file($_FILES['monfichier']['tmp_name'],
            'uploads/' . basename($_FILES['monfichier']['name']));
            echo "L'envoi a bien été effectué !";
        }
    }
}
?>

```

En résumé

- Les formulaires sont le moyen le plus pratique pour le visiteur de transmettre des informations à votre site. PHP est capable de récupérer les données saisies par vos visiteurs et de les traiter.
- Les données envoyées via un formulaire se retrouvent dans un array `$_POST`.
- De la même manière que pour les URL, il ne faut pas donner sa confiance absolue aux données que vous envoie l'utilisateur. Il pourrait très bien ne pas remplir tous les champs voire trafiquer le code HTML de la page pour supprimer ou ajouter des champs. Traitez les données avec vigilance.
- Que ce soit pour des données issues de l'URL (`$_GET`) ou d'un formulaire (`$_POST`), il faut s'assurer qu'aucun texte qui vous est envoyé ne contient du HTML si celui-ci est destiné à être affiché sur une page. Sinon, vous ouvrez une faille appelée XSS qui peut être néfaste pour la sécurité de votre site.
- Pour éviter la faille XSS, il suffit d'appliquer la fonction `htmlspecialchars` sur tous les textes envoyés par vos visiteurs que vous afficherez.
- Les formulaires permettent d'envoyer des fichiers. On retrouve les informations sur les fichiers envoyés dans un array `$_FILES`. Leur traitement est cependant plus complexe.

Variables super globales, sessions et cookies

Les variables super globales

Les variables superglobales sont des variables un peu particulières pour trois raisons :

- elles sont écrites en majuscules et commencent toutes, à une exception près, par un underscore (_). `$_GET` et `$_POST` en sont des exemples que vous connaissez ;
- les superglobales sont des `array` car elles contiennent généralement de nombreuses informations ;
- enfin, ces variables sont automatiquement créées par PHP à chaque fois qu'une page est chargée. Elles existent donc sur toutes les pages et sont accessibles partout : au milieu de votre code, au début, dans les fonctions, etc.

- `$_SERVER` : ce sont des valeurs renvoyées par le serveur. Elles sont nombreuses et quelques-unes d'entre elles peuvent nous être d'une grande utilité. Je vous propose de retenir au moins `$_SERVER['REMOTE_ADDR']`. Elle nous donne l'adresse IP du client qui a demandé à voir la page, ce qui peut être utile pour l'identifier.
- `$_ENV` : ce sont des variables d'environnement toujours données par le serveur. C'est le plus souvent sous des serveurs Linux que l'on retrouve des informations dans cette superglobale. Généralement, on ne trouvera rien de bien utile là-dedans pour notre site web.
- `$_SESSION` : on y retrouve les variables de session. Ce sont des variables qui restent stockées sur le serveur le temps de la présence d'un visiteur. Nous allons apprendre à nous en servir dans ce chapitre.
- `$_COOKIE` : contient les valeurs des cookies enregistrés sur l'ordinateur du visiteur. Cela nous permet de stocker des informations sur l'ordinateur du visiteur pendant plusieurs mois, pour se souvenir de son nom par exemple.
- `$_GET` : vous la connaissez, elle contient les données envoyées en paramètres dans l'URL.
- `$_POST` : de même, c'est une variable que vous connaissez et qui contient les informations qui viennent d'être envoyées par un formulaire.
- `$_FILES` : elle contient la liste des fichiers qui ont été envoyés via le formulaire précédent.

Fonctionnement des sessions

Comment sont gérées les sessions en PHP ? Voici les trois étapes à connaître.

1. Un visiteur arrive sur votre site. On demande à créer une session pour lui. PHP génère alors un numéro unique. Ce numéro est souvent très gros et écrit en hexadécimal¹, par exemple : a02bbff6198e6e0cc2715047bc3766f.
2. Une fois la session générée, on peut créer une infinité de variables de session pour nos besoins. Par exemple, on peut créer une variable `$_SESSION['nom']` qui contient le nom du visiteur, `$_SESSION['prenom']` qui contient le prénom, etc. Le serveur conserve ces variables même lorsque la page PHP a fini d'être générée. Cela veut dire que, quelle que soit la page de votre site, vous pourrez récupérer par exemple le nom et le prénom du visiteur via la superglobale `$_SESSION` !
3. Lorsque le visiteur se déconnecte de votre site, la session est fermée et PHP « oublie » alors toutes les variables de session que vous avez créées. Il est en fait difficile de savoir précisément quand un visiteur quitte votre site. En effet, lorsqu'il ferme son navigateur ou va sur un autre site, le vôtre n'en est pas informé. Soit le visiteur clique sur un bouton « Déconnexion » (que vous aurez créé) avant de s'en aller, soit on attend quelques minutes d'inactivité pour le déconnecter automatiquement : on parle alors de **timeout**. Le plus souvent, le visiteur est déconnecté par un timeout.



Il y a un petit piège : il faut appeler `session_start()` sur chacune de vos pages AVANT d'écrire le moindre code HTML (avant même la balise `<!DOCTYPE>`). Si vous oubliez de lancer `session_start()`, vous ne pourrez pas accéder aux variables superglobales `$_SESSION`.

EXAMPLE :

```

<?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 PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/x
↪ html1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
    <head>
        <title>Titre de ma page</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
↪ />
    </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>

```



Si vous voulez détruire manuellement la session du visiteur, vous pouvez faire un lien « Déconnexion » amenant vers une page qui fait appel à la fonction `session_destroy()`. Néanmoins, sachez que sa session sera automatiquement détruite au bout d'un certain temps d'inactivité.

Les cookies

Qu'est-ce qu'un cookie?

Un cookie, c'est un petit fichier que l'on enregistre sur l'ordinateur du visiteur. Ce fichier contient du texte et permet de « retenir » des informations sur le visiteur. Par exemple, vous inscrivez dans un cookie le pseudo du visiteur, comme ça la prochaine fois qu'il viendra sur votre site, vous pourrez lire son pseudo en allant regarder ce que son cookie contient.

Parfois les cookies ont une mauvaise image. On fait souvent l'erreur de penser que les cookies sont « dangereux ». Non, ce ne sont pas des virus, juste de petits fichiers texte qui permettent de retenir des informations. Au pire, un site marchand peut retenir que vous aimez les appareils photos numériques et vous afficher uniquement des pubs pour des appareils photos, mais c'est tout, ces petites bêtes sont inoffensives pour votre ordinateur.

Chaque cookie stocke généralement une information à la fois. Si vous voulez stocker le pseudonyme du visiteur et sa date de naissance, il est donc recommandé de créer deux cookies.

Créer un cookie qui va expirer après un an :

Voici donc comment on peut créer un cookie :

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

Je vous **recommande** donc de créer votre cookie plutôt comme ceci :

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

Le dernier paramètre **true** permet d'activer le mode **httpOnly** sur le cookie, et donc de le rendre en quelque sorte plus sécurisé. Ça ne coûte rien et vous diminuez le risque qu'un jour l'un de vos visiteurs puisse se faire voler le contenu d'un cookie à cause d'une faille XSS.

EXEMPLE :

```
| <?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 PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/x  
↪ html1/DTD/xhtml1-strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >  
  <head>  
    <title>Ma super page PHP</title>  
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"  
↪ />  
  </head>  
  <body>  
  
    etc.
```

AFFICHER UN 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 : écraser l'ancienne cookie

```
| setcookie('pays', 'Chine', time() + 365*24*3600, null, null, false, true);
```

En résumé

- Les variables superglobales sont des variables automatiquement créées par PHP. Elles se présentent sous la forme d'arrays contenant différents types d'informations.
- Dans les chapitres précédents, nous avons découvert deux superglobales essentielles : `$_GET` (qui contient les données issues de l'URL) et `$_POST` (qui contient les données issues d'un formulaire).
- La superglobale `$_SESSION` permet de stocker des informations qui seront automatiquement transmises de page en page pendant toute la durée de visite d'un internaute sur votre site. Il faut au préalable activer les sessions en appelant la fonction `session_start()`.
- La superglobale `$_COOKIE` représente le contenu de tous les cookies stockés par votre site sur l'ordinateur du visiteur. Les cookies sont de petits fichiers que l'on peut écrire sur la machine du visiteur pour retenir par exemple son nom. On crée un cookie avec la fonction `setcookie()`.

Lire et écrire dans un fichier :

Ouvrir et fermer un fichier :

```
<?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 :

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.



Vous n'êtes absolument pas obligés de donner l'extension .txt à votre fichier.
Vous pouvez l'appeler comme vous voulez : `compteur.cpt`, `compteur.num`,
ou même `compteur` tout court.

Procédure :

1. Vous l'ouvrez avec `fopen`.
2. Vous lisez par exemple la première ligne avec `fgets`.
3. Oui mais voilà : maintenant, le « curseur » de PHP se trouve à la fin de la première ligne (vu qu'il vient de lire la première ligne), comme dans la figure 14.3.

38472
↑

FIGURE 14.3 – Le curseur de PHP est à la fin de la première ligne

```
fseek($monfichier, 0)
```

38472
↑

FIGURE 14.4 – Le curseur de PHP est replacé à l'endroit choisi

Résumé :

En résumé

- PHP permet d'enregistrer des informations dans des fichiers sur le serveur.
- Il faut au préalable s'assurer que les fichiers autorisent PHP à les modifier. Pour cela, il faut changer les permissions du fichier (on parle de CHMOD) à l'aide d'un logiciel FTP comme FileZilla. Donnez la permission 777 au fichier pour permettre à PHP de travailler dessus.
- La fonction `fopen` permet d'ouvrir le fichier, `fgets` de le lire ligne par ligne et `fputs` d'y écrire une ligne.
- À moins de stocker des données très simples, l'utilisation des fichiers n'est pas vraiment la technique la plus adaptée pour enregistrer des informations. Il est vivement recommandé de faire appel à une base de données.

Stocker des informations dans une base de données :

PHP et MYSQL

- Une base de données est un outil qui stocke vos données de manière organisée et vous permet de les retrouver facilement par la suite.
- On communique avec MySQL grâce au langage SQL. Ce langage est commun à tous les systèmes de gestion de base de données ².
- PHP fait l'intermédiaire entre vous et MySQL.
- Une base de données contient plusieurs tables.
- Chaque table est un tableau où les colonnes sont appelées « champs » et les lignes « entrées ».

Type de donné MYSQL :

- 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).

Résumé :

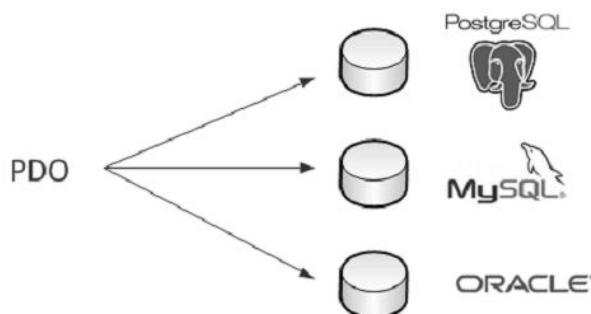
En résumé

- phpMyAdmin est un outil qui nous permet de visualiser rapidement l'état de notre base de données ainsi que de la modifier, sans avoir à écrire de requêtes SQL.
- On crée généralement un champ nommé **id** qui sert à numérotter les entrées d'une table. Ce champ doit avoir un index PRIMARY (on dit qu'on crée une clé primaire) et l'option AUTO_INCREMENT qui permet de laisser MySQL gérer la numérotation.
- MySQL gère différents types de données pour ses champs, à la manière de PHP. On trouve des types adaptés au stockage de nombres, de textes, de dates, etc.
- phpMyAdmin possède un outil d'importation et d'exportation des tables qui nous permettra notamment d'envoyer notre base de données sur Internet lorsque nous mettrons notre site en ligne.

Lire des données avec PHP

Comment se connecte-t-on à la base de données en PHP?

- L'extension `mysql_` : ce sont des fonctions qui permettent d'accéder à une base de données MySQL et donc de communiquer avec MySQL. Leur nom commence toujours par `mysql_`. Toutefois, ces fonctions sont vieilles et on recommande de ne plus les utiliser aujourd'hui.
- L'extension `mysqli_` : ce sont des fonctions améliorées d'accès à MySQL. Elles proposent plus de fonctionnalités et sont plus à jour.
- L'extension PDO : c'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle.



CODE :

```
<?php  
$bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '');  
?>
```



Le premier paramètre (qui commence par `mysql`) s'appelle le DSN : **Data Source Name**. C'est généralement le seul qui change en fonction du type de base de données auquel on se connecte.

Afficher les résultats d'un requête :

```
<?php  
try  
{  
    // On se connecte à MySQL  
    $bdd = new PDO('mysql:host=localhost;dbname=test', '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
?>

```

Le `fetch` renvoie faux (`false`) dans `$donnees` lorsqu'il est arrivé à la fin des données, c'est-à-dire que toutes les entrées ont été passées en revue. Dans ce cas, la condition du `while` vaut faux et la boucle s'arrête.

Mot clé LIMIT dans MYSQL

LIMIT

LIMIT nous permet de ne sélectionner qu'une partie des résultats⁴. C'est très utile lorsqu'il y a beaucoup de résultats et que vous souhaitez les paginer⁵.

4. Par exemple les 20 premiers.

5. C'est-à-dire par exemple afficher les 30 premiers résultats sur la page 1, les 30 suivants sur la page 2, etc.

EXEMPLE :

```
| SELECT * FROM jeux_video LIMIT 0, 20
```

- LIMIT 0, 20 : affiche les vingt premières entrées ;
- LIMIT 5, 10 : affiche de la sixième à la quinzième entrée ;
- LIMIT 10, 2 : affiche la onzième et la douzième entrée.

Les requêtes préparées :

AVEC DES MARQUEURS « ? »

```

<?php
$req = $bdd->prepare('SELECT nom FROM jeux_video WHERE possesseur = ?');
?>
```

```

<?php
$req = $bdd->prepare('SELECT nom FROM jeux_video WHERE possesseur = ?');
$req->execute(array($_GET['possesseur']));
?>
```

AVEC DES MARQUEURS « VARIABLE »

```

<?php
$req = $bdd->prepare('SELECT nom, prix FROM jeux_video WHERE possesseur
↪ = :possesseur AND prix <= :prixmax');
$req->execute(array('possesseur' => $_GET['possesseur'], 'prixmax'
↪ => $_GET['prix_max']));
?>

```

Traquer les erreurs :

```

<?php
$reponse = $bdd->query('SELECT nom FROM jeux_video') or die(print_r(
↪ $bdd->errorInfo()));
?>

```

Case d'une requête préparée :

```

$req->execute(array('possesseur' => 'Patrick', 'prixmax' => 20))
↪ or die(print_r($req->errorInfo()));
?>

```

Résumé :

En résumé

- Pour dialoguer avec MySQL depuis PHP, on fait appel à l'extension PDO de PHP.
- Avant de dialoguer avec MySQL, il faut s'y connecter. On a besoin de l'adresse IP de la machine où se trouve MySQL, du nom de la base de données ainsi que d'un login et d'un mot de passe.
- Les requêtes SQL commençant par SELECT permettent de récupérer des informations dans une base de données.
- Il faut faire une boucle en PHP pour récupérer ligne par ligne les données renvoyées par MySQL.
- Le langage SQL propose de nombreux outils pour préciser nos requêtes, à l'aide notamment des mots-clés WHERE (filtre), ORDER BY (tri) et LIMIT (limitation du nombre de résultats).
- Pour construire une requête en fonction de la valeur d'une variable, on passe par un système de requête préparée qui permet d'éviter les dangereuses failles d'injection SQL.

Ecrire des données

Insertion

```

// 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 de données variables grâce à une requête préparée

```
<?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 !';
?>
```

Mise à jour

```
<?php
$bdd->exec('UPDATE jeux_video SET prix = 10, nbre_joueurs_max = 32
→ WHERE nom = \'Battlefield 1942\'');
?>
```

Notez que cet appel renvoie le nombre de lignes modifiées. Essayez de récupérer cette valeur dans une variable et de l'afficher, par exemple comme ceci :

```
<?php
$nb_modifs = $bdd->exec('UPDATE jeux_video SET possesseur = \'Florent\'',
→ WHERE possesseur = \'Michel\'');
echo $nb_modifs . ' entrées ont été modifiées !';
?>
```

Mise à jour avec une requête préparée

```
<?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
));
?>
```

Suppression

```
| DELETE FROM jeux_video WHERE nom='Battlefield 1942'
```

Résumé :

En résumé

- On utilise différents mots-clés en fonction du type de modification que l'on souhaite effectuer :
 - **INSERT INTO** : ajout d'une entrée ;
 - **UPDATE** : modification d'une ou plusieurs entrées ;
 - **DELETE** : suppression d'une ou plusieurs entrées.
- Comme pour la sélection de données, on utilise les requêtes préparées pour personnaliser nos requêtes en fonction de variables.
- Lorsqu'on utilise **UPDATE** ou **DELETE**, il faut penser à filtrer avec un **WHERE** sinon toute la table sera affectée par l'opération !

Fonctions SQL

Utiliser les alias

```
<?php  
$reponse = $bdd->query('SELECT UPPER(nom) AS nom_maj FROM jeux_video');  
  
while ($donnees = $reponse->fetch())  
{  
    echo $donnees['nom_maj'] . '<br />';  
}
```

Résumé :

En résumé

- MySQL permet d'exécuter certaines fonctions lui-même, sans avoir à passer par PHP. Ces fonctions modifient les données renvoyées.
- Il existe deux types de fonctions :
 - les **fonctions scalaires** : elles agissent sur chaque entrée récupérée. Elles permettent par exemple de convertir tout le contenu d'un champ en majuscules ou d'arrondir chacune des valeurs ;
 - les **fonctions d'agrégat** : elles effectuent des calculs sur plusieurs entrées pour retourner une et une seule valeur. Par exemple : calcul de la moyenne, somme des valeurs, comptage du nombre d'entrées... .
- On peut donner un autre nom aux champs modifiés par les fonctions en créant des alias à l'aide du mot-clé **AS**.
- Lorsqu'on utilise une fonction d'agrégat, il est possible de regrouper des données avec **GROUP BY**.
- Après un groupement de données, on peut filtrer le résultat avec **HAVING**. Il ne faut pas le confondre avec **WHERE** qui filtre avant le groupement des données.

TYPE DE DONNEE MYSQL

Les différents types de dates

Voici les différents types de dates que peut stocker MySQL :

- 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.

Cela fait beaucoup de choix ! Dans la pratique, je vous invite à retenir surtout DATE (AAAA-MM-JJ) quand le moment de la journée importe peu, et DATETIME (AAAA-MM-JJ HH:MM:SS) quand vous avez besoin du jour et de l'heure précise à la seconde près.

NOW() : obtenir la date et l'heure actuelles

C'est probablement une des fonctions que vous utiliserez le plus souvent. Lorsque vous insérez un nouveau message dans la base, vous souhaiterez enregistrer la date actuelle les 99 % du temps. Pour cela, rien de plus simple avec la fonction NOW() :

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

La date sera alors automatiquement remplacée par la date et l'heure actuelles au format AAAA-MM-JJ HH:MM:SS.

Notez qu'il existe aussi les fonctions CURDATE() et CURTIME() qui retournent respectivement uniquement la date (AAAA-MM-JJ) et l'heure (HH:MM:SS).

DAY(), MONTH(), YEAR() : extraire le jour, le mois ou l'année

Extraire des informations d'une date ? C'est facile ! Voici un exemple d'utilisation :

```
| SELECT pseudo, message, DAY(date) AS jour FROM minichat
```

HOUR(), MINUTE(), SECOND() : extraire les heures, minutes, secondes

De la même façon, avec ces fonctions il est possible d'extraire les heures, minutes et secondes d'un champ de type DATETIME ou TIME.

DATE_FORMAT : formater une date

```
| SELECT pseudo, message, DATE_FORMAT(date, '%d/%m/%Y %Hh%imin%ss') AS date  
|   FROM minichat
```

Ainsi, on récupèrera les dates avec un champ nommé **date** sous la forme 11/03/2010 15h47min49s.

DATE_ADD et DATE_SUB : ajouter ou soustraire des dates

Il est possible d'ajouter ou de soustraire des heures, minutes, secondes, jours, mois ou années à une date. Il faut envoyer deux paramètres à la fonction : la date sur laquelle travailler et le nombre à ajouter ainsi que son type.

Par exemple, supposons que l'on souhaite afficher une date d'expiration du message. Celle-ci correspond à « la date où a été posté le message + 15 jours ». Voici comment écrire la requête :

```
SELECT pseudo, message, DATE_ADD(date, INTERVAL 15 DAY) AS date_expiration  
→ FROM minichat
```

Résumé :

En résumé

- MySQL propose plusieurs types de champs pour stocker des dates.
- Les deux types les plus couramment utilisés sont :
 - DATE : stocke une date au format AAAA-MM-JJ ;
 - DATETIME : stocke une date et une heure au format AAAA-MM-JJ HH:MM:SS.
- On peut trier et filtrer des champs contenant des dates comme s'il s'agissait de nombres.
- Il existe de nombreuses fonctions SQL dédiées au traitement des dates. La plus connue est NOW() qui renvoie la date et l'heure actuelles.

CREER DES IMAGES EN PHP

```
<?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);  
?>
```

Écrire 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 »². 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. Voici, en figure 24.3, le schéma de notre image orange de tout à l'heure, qui est de taille 200×50 :



Notez que les coordonnées de l'image s'arrêtent à (199, 49) et non à (200, 50) comme on pourrait s'y attendre. En effet, il ne faut pas oublier qu'on commence à compter à partir de 0 ! De 0 à 199 il y a bien 200 points.



FIGURE 24.3 – Coordonnées de l'image

- `$texte_a_ecrire`, c'est le... texte que vous voulez écrire. Non, non, il n'y a pas de piège.
- `$couleur`, c'est une couleur que vous avez créée tout à l'heure avec `imagecolorallocate`.

EXEMPLE :

```
<?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);

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

imagepng($image);
?>
```

ImageSetPixel

Son rôle : dessiner un pixel aux coordonnées (x, y) .

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

Illustration en figure 24.4, grâce au code :
ImageSetPixel (\$image, 100, 100, \$noir);

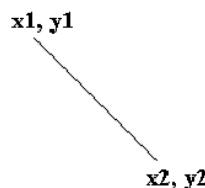
x, y

FIGURE 24.4 – ImageSetPixel

ImageLine

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

La preuve en image : figure 24.5. Le code utilisé est le suivant :
ImageLine (\$image, 30, 30, 120, 120, \$noir);



ImageEllipse

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

Voici une illustration en figure 24.6. Et voici son code :
ImageEllipse (\$image, 100, 100, 100, 50, \$noir);

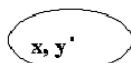


FIGURE 24.6 – ImageEllipse

ImageRectangle

Elle, elle dessine un rectangle, dont le coin en haut à gauche est de coordonnées (x_1, y_1) et celui en bas à droite (x_2, y_2) .

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

Figure 24.7, vous avez le résultat produit par le code suivant :
ImageRectangle (\$image, 30, 30, 160, 120, \$noir);

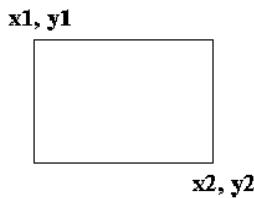


FIGURE 24.7 – ImageRectangle

Rendre une image transparente

```
| <?php  
| imagecolortransparent($image, $couleur);  
| ?>
```

Mélanger deux images :

imagecopymerge(dst_im, src_im, dst_x, dst_y, src_x, src_y, src_w, src_h, pct)

Script :

```
<?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);  
?>
```

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. ;-)

Redimensionner une image :

```
imagecopyresampled(dst_image, src_image, dst_x, dst_y, src_x, src_y, dst_w, dst_h, src_w, src_h)
```

```
<?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.

Résumé :

En résumé

- PHP permet de faire bien plus que générer des pages web HTML. En utilisant des extensions, comme la bibliothèque GD, on peut par exemple générer des images.
- Pour qu'une page PHP renvoie une image au lieu d'une page web, il faut modifier l'en-tête HTTP à l'aide de la fonction `header()` qui indiquera alors au navigateur du visiteur l'arrivée d'une image.
- Il est possible d'enregistrer l'image sur le disque dur si on le souhaite, ce qui évite d'avoir à la générer à chaque fois qu'on appelle la page PHP.
- On peut créer des images avec GD à partir d'une image vide ou d'une image déjà existante.
- GD propose des fonctions d'écriture de texte dans une image et de dessin de formes basiques.
- Des fonctions plus avancées de GD permettent de fusionner des images ou d'en redimensionner.

LES EXPRESSIONS RÉGULIÈRES :

Les types de REGEX

- **POSIX** : c'est un langage d'expressions régulières mis en avant par PHP, qui se veut un peu plus simple que PCRE (ça n'en reste pas moins assez complexe). Toutefois, son principal et gros défaut je dirais, c'est que ce « langage » est plus lent que PCRE ;
- **PCRE** : ces expressions régulières sont issues d'un autre langage (le Perl). Considérées comme un peu plus complexes, elles sont surtout bien plus rapides et performantes.

Les fonctions PCRE :

- preg_grep;
- preg_split;
- preg_quote;
- preg_match;
- preg_match_all;
- preg_replace;
- preg_replace_callback.

REMARQUE :

les **Regex** sont sensible à la case

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

Le symbole 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

Chaîne	Regex	Résultat
Bonjour petit zéro	#^Bonjour#	VRAI
Bonjour petit zéro	#zéro\$#	VRAI
Bonjour petit zéro	#^zéro#	FAUX
Bonjour petit zéro!!!	#zéro\$#	FAUX

Des classes simples

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 classe

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 21 ^e siècle	#^ [0-9] #	FAUX
<h1>Une balise de titre HTML</h1>	#<h[1-6]>#	VRAI

La négation

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
ScrrmmmblllGnnngnngnMmmmmffff	# [^aeiouy] #	VRAI

Les quantificateurs

- ? (point d'interrogation) : ce symbole indique que la lettre est facultative. **Elle peut y être 0 ou 1 fois.** Ainsi, #a?# reconnaît 0 ou 1 « a » ;
- + (signe plus) : la lettre est obligatoire. **Elle peut apparaître 1 ou plusieurs fois.** Ainsi, #a+# reconnaît « a », « aa », « aaa », « aaaa », etc. ;
- * (étoile) : la lettre est facultative. **Elle peut apparaître 0, 1 ou plusieurs fois.** Ainsi, #a*# reconnaît « a », « aa », « aaa », « aaaa », etc. Mais s'il n'y a pas de « a », ça fonctionne aussi !

REMARQUE :



Notez que ces symboles s'appliquent à la lettre se trouvant directement devant. On peut ainsi autoriser le mot « chien », qu'il soit au singulier comme au pluriel, avec la regex #chiens?# (fonctionnera pour « chien » et « chiens »).

#bor?is#

Ce code reconnaîtra « boris » et « bois » !

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

Les accolades :

Il y a trois façons d'utiliser les accolades.

- **{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 2^e 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. ⁵

REMARQUE :



Si vous faites attention, vous remarquez que :

- ? revient à écrire {0,1} ;
- + revient à écrire {1,} ;
- * revient à écrire {0,}.

Chaîne	Regex	Résultat
eeeeee	<code>#e{2,}#</code>	VRAI
Blablablabla	<code>#^Bla(bla){4}\$#</code>	FAUX
546781	<code>#^ [0-9]{6}\$#</code>	VRAI

Résumé :

En résumé

- Les expressions régulières sont des outils de recherche et de remplacement de texte très avancés qui permettent d'effectuer des recherches très précises, pour vérifier par exemple que le texte saisi par l'utilisateur correspond bien à la forme d'une adresse e-mail ou d'un numéro de téléphone.
- La fonction `preg_match` vérifie si un texte correspond à la forme décrite par une expression régulière.
- Une expression régulière est délimitée par un symbole (par exemple le dièse #).
- Les classes de caractères permettent d'autoriser un grand nombre de symboles (lettres et chiffres) selon un intervalle.
- Les quantificateurs permettent d'exiger la répétition d'une chaîne de texte un certain nombre de fois.

Les métacaractères

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

Echapper un métacaractère

EXEMPLE :

`#Quoi \?#`

REMARQUE :



C'est la même chose pour tous les autres métacaractères que je vous ai montrés plus haut (# ! ^ \$ () [] { } ? + * . \) : il faut mettre un antislash devant si vous voulez les utiliser dans votre recherche. Vous remarquerez que pour utiliser un antislash il faut... un antislash devant ! Comme ceci : \\.

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

Le cas des classes :

3 cas particuliers, cependant.

- « # » (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(les 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 mot. 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).
.	Le point indique n'importe quel caractère! Il autorise donc tout !

REMARQUE :



Pour le point, il existe une exception : il indique tout **sauf les entrées** (\n). Pour faire en sorte que le point indique tout, même les entrées, vous devrez utiliser l'option « s » de PCRE. Exemple : #[0-9]-.#s

EXEMPLE : REGEX D'UN EMAIL

#^ [a-z0-9_.-]+@[a-z0-9_.-]{2,}\. [a-z]{2,4}\$#

Testez donc des adresses comme :

- the_cypher@hotmail.com ;
- business_consultants@free4work.info ;
- mega-killer.le-retour@super-site.fr.st ;
- etc., etc.

Des regex avec MySQL

MySQL ne comprend que les regex en langage POSIX, et pas PCRE

Vous avez juste besoin de retenir ce qui suit pour faire une regex POSIX :

- il n'y a pas de délimiteur ni d'options. Votre regex n'est donc pas entourée de dièses ;
- il n'y a pas de classes abrégées comme on l'a vu plus haut, donc pas de \d, etc.

En revanche, vous pouvez toujours utiliser le point pour dire : « n'importe quel caractère ».

EXEMPLE :

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

Cela signifie : Sélectionne tous les noms de la table visiteurs dont l'IP commence par « 84.254 » et se termine par deux autres nombres de un à trois chiffre(s) (ex. : 84.254.6.177).

Capture et remplacement

EXEMPLE:

Bon, la théorie de tout ça est délicate à expliquer, alors je vais vous montrer de suite comment on fait pour mettre en gras tous les mots compris entre des [b] [/b] :

```
<?php  
$texte = preg_replace('#\[b\](.+)\[/b\]#i', '<strong>$1</strong>', $texte);  
?>
```

Voici comment s'utilise la fonction `preg_replace`.

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 : « \$1 » (je vous rappelle que `` 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 `` !
3. Enfin, dernier paramètre, c'est le texte dans lequel on fait notre recherche / remplacement (ça, vous connaissez déjà).

La fonction `preg_replace` renvoie le résultat après avoir fait les remplacements.

Si je schématisé le fonctionnement, ça donne la figure 26.1.

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

Remarque :



N'oubliez pas que c'est **l'ordre dans lequel les parenthèses sont ouvertes** qui est important.

Échapper une parenthèse :

```
#(anti)co(?:nsti)(tu(tion)nelle)ment#
```

1. **\$0** : anticonstitutionnellement
2. **\$1** : anti
3. **\$2** : tutionnelle
4. **\$3** : tion

Créer votre bbCode :

- [b] [/b] : pour mettre du texte en gras ;
- [i] [/i] : pour mettre du texte en italique ;
- [color=red] [/color] : pour colorer le texte (il faudra laisser le choix entre plusieurs couleurs).

Options :

- 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, grossièrement, ça ne marchera 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.

Voici donc le code correct pour mettre en gras et en italique avec le bbCode :

```
<?php
$texte = preg_replace('#\[b\](.+)\[/b\]#isU', '<strong>$1</strong>', $texte);
$texte = preg_replace('#\[i\](.+)\[/i\]#isU', '<em>$1</em>', $texte);
?>
```

EXAMPLE :

```

<?php
if (isset($_POST['texte']))
{
    $texte = stripslashes($_POST['texte']); // On enlève les slashes
    // 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>

```

Résumé :

En résumé

- Certains caractères sont spéciaux au sein d'une expression régulière : on parle de métacaractères. Si on souhaite les rechercher dans une chaîne, il faut les échapper en

plaçant un symbole antislash devant. Par exemple : \[.

- Il existe des classes abrégées, c'est-à-dire des classes toutes prêtes, comme par exemple \d qui revient à écrire [0-9].
- La fonction `preg_replace` permet d'effectuer des remplacements dans une chaîne de texte.
- Dans le cas d'un remplacement, les parenthèses au sein d'une expression régulière permettent de capturer une portion de texte pour la réutiliser dans une autre chaîne.

Mémento Regex

Base :

regex	Explication
#guitare#	Cherche le mot « guitare » dans la chaîne.
#guitare piano#	Cherche le mot « guitare » OU « piano ».
#^guitare#	La chaîne doit commencer par « guitare ».
#guitare\$#	La chaîne doit se terminer par « guitare ».
#^guitare\$#	La chaîne doit contenir uniquement « guitare ».

Classes de caractères :

regex	Explication
#gr[ioa]s#	Chaîne qui contient « gris », ou « gros », ou « gras ».
[a-z]	Caractères minuscules de a à z.
[0-9]	Chiffres de 0 à 9.
[a-e0-9]	Lettres de « a » à « e » ou chiffres de 0 à 9.
[0-57A-Za-z.-]	Chiffres de 0 à 5, ou 7, ou lettres majuscules, ou lettres minuscules, ou un point, ou un tiret.
#[^0-9]#	Chaîne ne contenant PAS de chiffres.
#^[^0-9]#	Chaîne ne commençant PAS par un chiffre.

Quantificateurs :

regex	Explication
#a?#	« a » peut apparaître 0 ou 1 fois.
#a+#	« a » doit apparaître au moins 1 fois.
#a*#	« a » peut apparaître 0, 1 ou plusieurs fois.
#bor?is#	« bois » ou « boris ».
#Ay(ay oy)*#	Fonctionne pour Ay, Ayay, Ayoy, Ayayayoyayayoyoy, etc.
#a{3}#	« a » doit apparaître 3 fois exactement (« aaa »).
#a{3,5}#	« a » doit apparaître de 3 à 5 fois (« aaa », « aaaa », « aaaaa »).
#a{3,}#	« a » doit apparaître au moins 3 fois (« aaa », « aaaa », « aaaaa », « aaaaaa », etc.).

Métacarctères :

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

Pour utiliser un métacaractère dans une recherche, il faut l'échapper avec un antislash : \.

regex	Explication
#Hein?#	Cherche « Hei » ou « Hein ».
#Hein\?#	Cherche « Hein ? ».

Classes abrégées

Classe abrégée	Correspondance
\d	[0-9]
\D	[^0-9]
\w	[a-zA-Z0-9_]
\W	[^a-zA-Z0-9_]
\t	Tabulation.
\n	Nouvelle ligne.
\r	Retour chariot.
\s	Espace blanc (correspond à \t \n \r).
\S	Ce qui n'est PAS un espace blanc (\t \n \r).
.	Classe universelle.

Capture et remplacement

En utilisant la fonction preg_replace on peut automatiquement faire des remplacements à l'aide de regex.

```
<?php
$texte = preg_replace('#\[b\](.+)\[/b\]#i', '<strong>$1</strong>', $texte);
?>
```

Options

Il existe de nombreuses options que l'on peut utiliser avec les regex PCRE. Parmi les trois que nous sommes le plus souvent amenés à utiliser, il y a :

- **i** : la regex ne fera plus la différence entre majuscules / minuscules ;
- **s** : le point (classe universelle) fonctionnera aussi pour les retours à la ligne (\n) ;
- **U** : mode « Ungreedy » (pas gourmand). Utilisé pour que la regex s'arrête le plus tôt possible. Pratique par exemple pour le bbCode [b] [/b] : la regex s'arrêtera à la première occurrence de [/b].

PHP ET LA POO

Créer une classe

```
<?php
class Membre
{
    private $pseudo;
    private $email;
    private $signature;
    private $actif;
}
?>
```

REMARQUE :



Les développeurs PHP ont l'habitude de donner l'extension .class.php à leurs fichiers contenant des classes pour bien les distinguer. Quelques règles à ce sujet : ne définissez qu'une classe par fichier et donnez au fichier le même nom que votre classe. Le nom de votre classe devrait par ailleurs commencer par une majuscule.

Getters et Setters

```
<?php
class Membre
{
    private $pseudo;
    private $email;
    private $signature;
    private $actif;

    public function getPseudo()
    {
        return $this->pseudo;
    }

    public function setPseudo($nouveauPseudo)
    {
        $this->pseudo = $nouveauPseudo;
    }
}
```

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 !');
    }

    ...
}
```

Instanciation

```
<?php

include_once('Membre.class.php');

$membre = new Membre();
$membre->setPseudo('M@teo21');
echo $membre->getPseudo() . ', je vais te bannir !';
$membre->bannir();

?>
```

Constructeur : __construct

```
<?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.
```

Détruire un objet :

```
<?php
unset($membre);
?>
```

L'héritage :

```
<?php  
include_once('Membre.class.php');  
  
class Admin extends Membre  
{  
  
}  
?>
```

L'encapsulation :

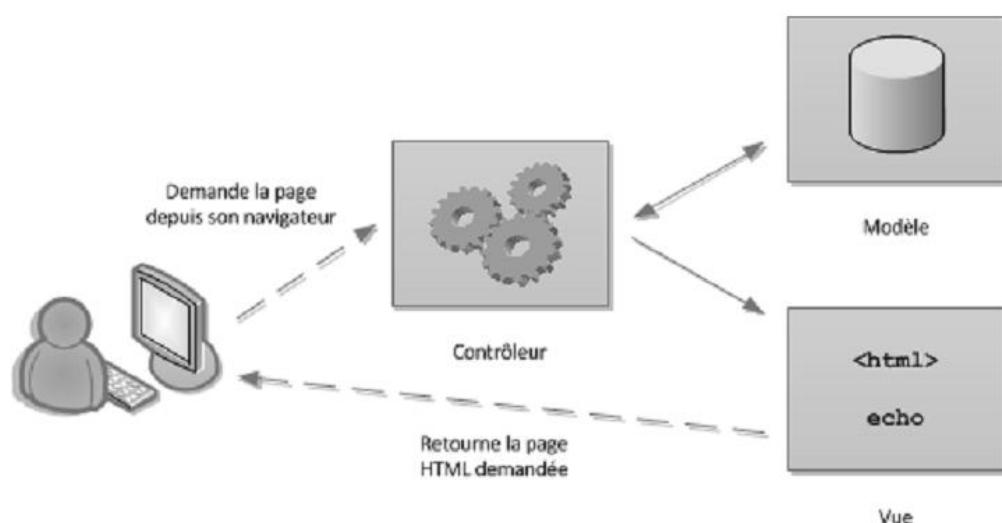
Toutes les variables d'une classe doivent toujours être privées ou protégées.

Résumé :

En résumé

- La programmation orientée objet est une autre façon de concevoir son code. Elle permet de concevoir les éléments de son site comme des objets à qui l'on donne des ordres et qui sont capables de modifier leur état.
- La **classe** est le modèle à partir duquel on peut créer plusieurs **objets**.
- Une classe est constituée de variables membres et de fonctions membres. Les fonctions modifient l'état des variables membres.
- Le constructeur (nommé `__construct()`) est une fonction appelée par PHP lorsqu'on crée un objet basé sur la classe. De même pour `__destruct()`, appelée lors de la suppression de l'objet.
- Une classe peut hériter d'une autre classe pour récupérer toutes ses propriétés et rajouter de nouvelles fonctionnalités spéciales.
- On oblige le programmeur qui utilise la classe à passer par des fonctions pour modifier l'objet. C'est le **principe d'encapsulation**.

ORGANISER SON CODE SELON L'ARCHITECTURE MVC

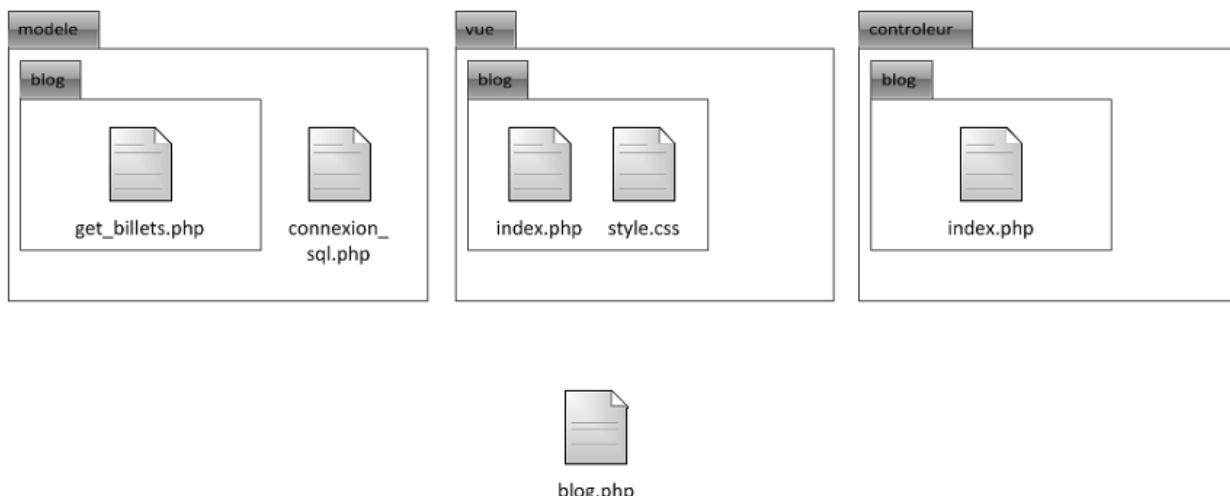


- **Modèle** : cette partie gère les *données* de votre site. Son rôle est d’aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu’elles puissent ensuite être traitées par le contrôleur. On y trouve donc les requêtes SQL¹.
- **Vue** : cette partie se concentre sur l’*affichage*. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu’elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple la liste des messages des forums.
- **Contrôleur** : cette partie gère la logique du code qui prend des *décisions*. C’est en quelque sorte l’intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C’est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d’accès).

Organisation :

Résumé des fichiers créés

Je pense qu’un schéma synthétisant l’architecture des fichiers que nous venons de créer ne sera pas de refus ! La figure 28.4 devrait vous permettre de mieux vous repérer.



EXEMPLE :

Le modèle

```

<?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 %H%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;
}
  
```

Le Contrôleur

```
<?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');
```

La Vue

```
<?php

include_once('modele/connexion_sql.php');

if (!isset($_GET['section']) OR $_GET['section'] == 'index')
{
    include_once('contrôleur/blog/index.php');
}
```

Utiliser Framework

Un **framework** est un ensemble de bibliothèques, une sorte de *kit* prêt à l'emploi pour créer plus rapidement son site web, tout en respectant des règles de qualité. Vous vous souvenez de la bibliothèque GD qui nous permettait de créer des images ? Les *frameworks* sont des assemblages de bibliothèques comme celle-ci. C'est dire si ce sont des outils puissants et complets !

Voici quelques *frameworks* PHP célèbres qu'il faut connaître :

- CodeIgniter ;
- CakePHP ;
- Symfony ;
- Jelix ;
- Zend Framework.

Résumé :

En résumé

- MVC est un *design pattern*, une bonne pratique de programmation qui recommande de découper son code en trois parties qui gèrent les éléments suivants :
 - Modèle : stockage des données ;
 - Vue : affichage de la page ;
 - Contrôleur : logique, calculs et décisions.
- Utiliser l'architecture MVC sur son site web en PHP est recommandé, car cela permet d'avoir un code plus facile à maintenir et à faire évoluer.
- De nombreux *frameworks* PHP, tels que Symfony et le Zend Framework, vous permettent de mettre rapidement en place les bases d'une architecture MVC sur votre site. Ce sont des outils appréciés des professionnels qui nécessitent cependant un certain temps d'adaptation.

FONCTION UTILE :

[GETIMAGESIZE\(\)](#)

```
<?php
$image_sizes = getimagesize($_FILES['icone']['tmp_name']);
if ($image_sizes[0] > $maxwidth OR $image_sizes[1] > $maxheight)
$erreur = "Image trop grande";
?>
```

Envoyé un mail : [MAIL\(TO, SUBJECT, MESSAGE\)](#)

Supprimer un fichier : [unlink\("FICHIER.TXT"\)](#)

calculer la taille libre de disque en octet : [DISKFREESPACE\('CHEMAIN'\)](#)

taille d'un fichier en octet : [FILESIZE\("INDEX.PHP"\)](#)

vérifier l'existence d'un dossier : [IS_DIR\("UPLOADS"\)](#)

créer un dossier avec un privilège : [mkdir\("DOSSIER",777\)](#)

suppression : [rmdir\("DOSSIER"\)](#)

renommer un fichier : [rename\('D','D2'\)](#)

créer un fichier temporaire : [tempnam\(dir, prefix\)](#)

nombre de valeur d'une array : [array_count_values\(input\)](#)

combiner des array : [array_merge\(array1,array2\)](#)

ajouter des valeurs à une array : [array_push\(array, var\)](#)

inverser une array : [array_reverse\(array\)](#)

supprimer la 1ère valeur : [array_shift\(array\)](#)

renvoi les valeur unique sans doublons : [\\$var=array_unique\(array\)](#)

Renvoi les valeur d'une array : [\\$var=array_values\(input\)](#)

testez si un valeur existe (true or false) : [in_array\(\\$val, array\)](#)

Debuging avec Xdebug

Active l'affichage des errors lorsqu'il sont désactive dans la config PHP

error_reporting(FLAG)

Modifie la valeur d'une option de configuration

```
ini_set("displa_errors",true);
```

Changer le Header s'il y a une faille XSS

```
Header('HTTP/1.1 418 Forbidden');
```

CSRF(Cross Site Request Forgery)

Pour protéger :

Utiliser le jeton de validité

Utiliser de Captcha

Protéger simplement les session

Utiliser la fonctionne `session_regenerate_id()` pour générer un nouvel ID lors de l'authentification