

<Découvrez le fonctionnement d'un site écrit en PHP

→ Il existe deux types de sites web :

- Les sites statiques
- Les sites dynamiques

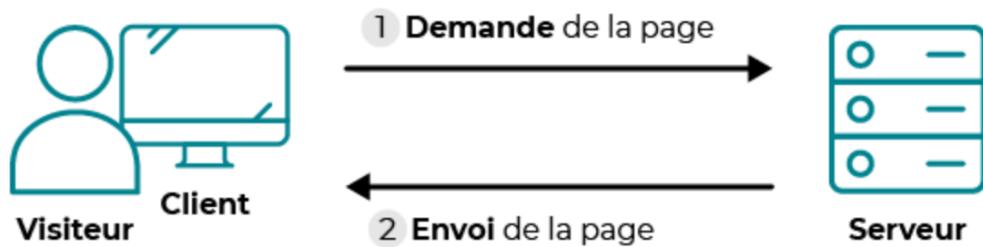
→ Un site statique est réalisé en HTML & CSS

→ Site statique adapté pour un site "vitrine"

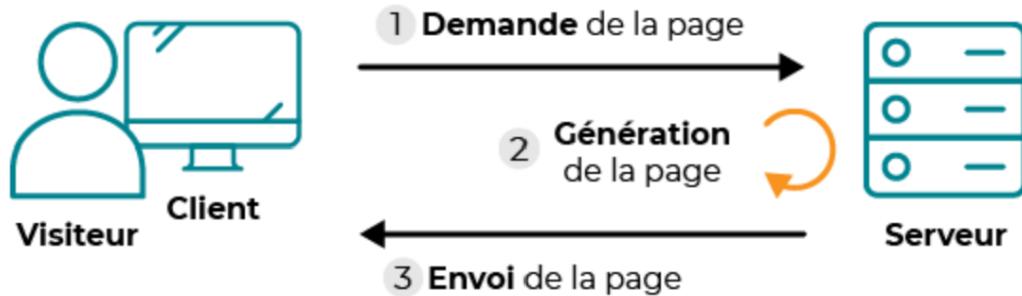
→ Le principe d'un site dynamique est qu'il utilise d'autres langages en plus de HTML et CSS, comme PHP et MySQL

→ Il est dit dynamique car il peut changer sans intervention du webmaster

Un site statique :



Un site dynamique :



→ La page est générée à chaque fois que le client la réclame.

Exemple code PHP :

```
1 <?php echo "Vous êtes le visiteur n°" . $nbre_visiteurs; ?>
```

- Installation de Mamp (faites sur le PC)
- Installation de Visual Studio Code
- IDE pour PHP : PHPStorm

Ecrivez votre premier script



Vous savez que le code source d'une page HTML est constitué de **balises** (aussi appelées **tags**, en anglais). Par exemple, `` est une balise.

Le code PHP viendra s'insérer au milieu du code HTML. On va progressivement placer dans nos pages web des morceaux de code PHP à l'intérieur du HTML. Ces bouts de code PHP seront les parties dynamiques de la page, c'est-à-dire les parties qui peuvent changer toutes seules.

- Dans les balises HTML on insère des données dynamiques au milieu de ces dernières

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <title>Ma page web</title>
6   </head>
7
8   <body>
9     <h1>Ma page web</h1>
10
11    <p>
12      Bonjour <!-- Insérer le pseudo du visiteur ici --> !
13    </p>
14
15  </body>
16 </html>
```

Reconnaître une balise PHP :

- Elle commence par `<?php`
- Et se termine par `?>`

Balise PHP vide :

```
1 <?php ?>
```

A l'intérieur de cette balise on écrit du code PHP :

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

Autres formes de balise PHP :

```
<? ?>
```

```
<% %>
```

```
<?= ?>
```

Instruction basique : `echo`

```
1 <?php echo "Ceci est du texte"; ?>
2
3 <!-- Ou bien, avec des parenthèses -->
4 <?php echo("Ceci est du texte"); ?>
```

Configurez PHP pour visualiser les erreurs

→ Pour localiser le fichier de configuration PHP du serveur web, il faut créer un fichier contenant la commande : `phpinfo()`

```
1 <?php
2
3 phpinfo();
```

En enregistrant la page sous un nom `info.php`, on obtient cette page :

PHP Version 7.4.1



System	Windows NT ALEXANDRA 6.2 build 9200 (Windows 8 Home Premium Edition) i586
Build Date	Jan 20 2020 22:10:35
Compiler	Visual C++ 2017
Architecture	x86
Configure Command	cscript /nologo configure.js "--disable-all" "--enable-cli" "--enable-cgi" "--enable-apache2-2handler" "--enable-apache2-4handler" "--with-bz2=shared" "--enable-mbstring=shared" "--enable-exif=shared" "--enable-pdo=shared" "--with-pdo-sqlite=shared" "--with-sqlite3=shared" "--with-curl=shared" "--with-openssl=shared" "--with-gd=shared" "--enable-json=static" "--enable-session=shared" "--with-mysqli=mysqlnd,shared" "--with-mysqli=mysqlnd" "--enable-opcache=shared" "--enable-apcu=shared" "--with-tidy=shared" "--enable-ftp=shared" "--with-gettext=shared" "--with-gmp=shared" "--with-imap=shared" "--enable-intl=shared" "--with-ldap=shared" "--with-pgsql=shared" "--with-pdo-fb=shared" "--with-pdo-odbc=shared" "--with-pdo-pgsql=shared" "--enable-shmop=shared" "--enable-phar" "--enable-hash" "--with-simplexml" "--enable-soap" "--with-libxml" "--with-iconv" "--with-xml" "--enable-zlib" "--enable-zip" "--enable-ctype" "--enable-filter" "--enable-sockets" "--enable-calendar" "--enable-bcmath" "--enable-fileinfo" "--enable-xmlreader" "--enable-xmlwriter" "--with-dom" "--enable-tokenizer" "--enable-com-dotnet" "--enable-odbc" "--with-wddx" "--enable-mbregex" "--with-imagick=shared,f:\prog\ImageMagick-6.8.9,f:\prog\ImageMagick-6.8.9\VisualMagick\lib" "--with-xsl=shared" "--with-pdo-mysql=mysqlnd,shared" "--enable-memcache=shared" "--with-oauth=shared"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\MAMP\conf\php7.4.1\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,TS,VC15
PHP Extension Build	API20190902,TS,VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled

Si l'on retire une `)` de la méthode on obtient cette erreur affichée :

Parse error: syntax error, unexpected ';' in **C:\MAMP\htdocs\tests\info.php** on line 3

Décrivez les éléments de votre projet à l'aide de variables

C'est quoi une variable ?

Une **variable**, c'est une petite information stockée en mémoire **temporairement**.

En PHP, la variable (l'information) existe tant que la page est en cours de génération. Dès que la page PHP est générée, toutes les variables sont supprimées de la mémoire car elles ne servent plus à rien.

Ce n'est donc pas un fichier qui reste stocké sur le disque dur, mais une petite information temporaire présente en mémoire vive.

Exemples de variables :

Valeur	Type	Exemple
Chaîne de caractères	string	\$authorName = 'Mathieu';
Nombres entiers	int	\$authorAge = 3;
Nombres décimaux	float	\$productPrice = 14.738;
Booléens	bool	\$isAllowed = true;
Rien	NULL	\$authorQuality = NULL;

Plus en détails :

- **Les chaînes de caractères (string)**: c'est le nom informatique qu'on donne au texte.
- **Les nombres entiers (int)**: ce sont les nombres du type 1, 2, 3, 4, etc. On compte aussi parmi eux les entiers relatifs : -1, -2, -3...
- **Les nombres décimaux (float)**: ce sont les nombres à virgule, comme 14,738. Attention, les nombres doivent être écrits avec un point au lieu de la virgule (c'est la notation anglaise).
- **Les booléens (bool)**: c'est un type très important qui permet de stocker soit vrai soit faux.
- **Rien (NULL)**: aussi bizarre que cela puisse paraître, on a parfois besoin de dire qu'une variable ne contient rien. Ce n'est pas vraiment un type de données, mais plutôt **l'absence** de type.

Exemple d'une méthode :

```
1 <?php
2 $userAge = 17;
3 ?>
```

Décrypter le code (exemple au-dessus) :

- Le \$: il précède toujours le nom d'une variable
- le = : il indique que le \$userAge (ici) vaut 17
- le ; : termine l'instruction

Concaténer une variable :

En fait, écrire "Mathieu Nebra" tout seul comme on l'a fait n'est pas très parlant. On aimerait écrire du texte autour pour dire : « Bienvenue Mathieu Nebra ». La concaténation est justement un moyen d'assembler du texte et des variables.

Concaténer avec guillemets et séparés avec des points :

```
1 <?php
2 $fullname = 'Mathieu Nebra';
3 echo 'Bonjour ' . $fullname . ' et bienvenue sur le site !'; // OK
4 ?>
```

Les opérations de bases :

Symbol	Signification
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo

Le modulo :

- Représente le reste de la division entière
- Il y a d'autres opérations plus complexes telles que : la racine carrée, l'exponentielle, la factorielle, etc.

Adaptez le comportement de votre application à l'aide des conditions

- Les conditions permettent de créer réellement un site dynamique

La structure de base `if... else :`

Une condition peut être écrite en PHP sous différentes formes. On parle de structures **conditionnelles**.

Les symboles à retenir :

Symbol	Signification
==	Est égal à
>	Est supérieur à
<	Est inférieur à
>=	Est supérieur ou égal à
<=	Est inférieur ou égal à
!=	Est différent de

Attention :

Il y a deux symboles « égal » (==) sur la première ligne.

Il ne faut pas confondre ça avec le simple = (que je vous ai appris dans le chapitre sur les variables).

Ici, le double égal sert à tester l'égalité, à dire « Si c'est égal à... ». Dans les conditions, on utilisera toujours le double égal (==).

La structure if... else :

- Pour introduire une condition on utilise le mot if
- On ajoute à la suite entre parenthèses la condition en elle-même
- Enfin, on ouvre des accolades (on mettra à l'intérieur les instructions à exécuter)

Exemple :

```
1 <?php
2 $isEnabled = true; // La condition d'accès
3
4 if ($isEnabled == true) {
5     echo "Vous êtes autorisé(e) à accéder au site ✓";
6 }
7 ?>
```

- La quasi-totalité des cas, c'est sur une variable qu'on fait la condition

Exemple amélioré :

```
1 <?php
2 $isEnabled = true;
3
4 if ($isEnabled == true) {
5     echo "Vous êtes autorisé(e) à accéder au site ✓";
6 }
7 else {
8     echo "Accès refusé ✗";
9 }
10 ?>
```

Exemple avec une autre variable :

```
1 <?php
2 $isAllowedToEnter = "Oui";
3
4 // SI on a l'autorisation d'entrer
5 if ($isAllowedToEnter == "Oui") {
6     // instructions à exécuter quand on est autorisé à entrer
7 } // SINON SI on n'a pas l'autorisation d'entrer
8 elseif ($isAllowedToEnter == "Non") {
9     // instructions à exécuter quand on n'est pas autorisé à entrer
10 } // SINON (la variable ne contient ni Oui ni Non, on ne peut pas agir)
11 else {
12     echo "Euh, je ne comprends pas ton choix, tu peux me le rappeler s'il te plaît ?";
13 }
14 ?>
```

→ le mot-clé `elseif` qui veut dire “sinon si”

Dans l'exemple, PHP rencontre les conditions suivantes :

1. Si `$isAllowedToEnter` est égale à « Oui », tu exécutes ces instructions...
2. Sinon, si `$isAllowedToEnter` est égale à « Non », tu exécutes ces autres instructions...
3. Sinon, tu redemandes l'âge pour savoir si on a ou non l'autorisation d'entrer.

Les booléens :

Se sont des variables qui valent :

- Soit “true”
- Soit “false”

Exemple de test d'une variable d'une booléenne :

```
1 <?php
2 $isAllowedToEnter = true;
3
4 if ($isAllowedToEnter) {
5     echo "Bienvenue petit nouveau. :o)";
6 }
7 else {
8     echo "T'as pas le droit d'entrer !";
9 }
10 ?>
```

L'avantage du booléen on n'est pas obligé d'ajouter le `== true`.

→ Le symbole  permet de vérifier si la variable vaut false

Exemple :

```
1 <?php
2 $isAllowedToEnter = true;
3
4 // Si pas autorisé
5 if (! $isAllowedToEnter) {
6
7 }
8 ?>
```

Posez des conditions multiples :

→ Poser plusieurs conditions à la fois

Nouveaux mots clés à retenir :

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

Exemple avec `&&` :

```
1 <?php
2 $isEnabled = true;
3 $isOwner = false;
4
5 if ($isEnabled && $isOwner) {
6     echo 'Accès à la recette validé ✅';
7 } else {
8     echo 'Accès à la recette interdit ! ❌';
9 }
```

Explication de l'exemple au-dessous :

- Si l'utilisateur est actif et qu'il est l'auteur il peut accéder à la recette validée
- Sinon il verra le message refusé

Exemple avec || :

```
1 <?php
2 $isEnabled = true;
3 $isOwner = false;
4 $isAdmin = true;
5
6 if (($isEnabled && $isOwner) || $isAdmin) {
7     echo 'Accès à la recette validé ✅';
8 } else {
9     echo 'Accès à la recette interdit ! ❌';
10 }
```

“Astuce” bonus, les deux exemples de code ci-dessous donne le même résultat :

```
1 <?php
2 $chickenRecipesEnabled = true;
3
4 if ($chickenRecipesEnabled) {
5     echo '<h1>Liste des recettes à base de poulet</h1>';
6 }
7 ?>
```

```
1 <?php $chickenRecipesEnabled = true; ?>
2
3 <?php if ($chickenRecipesEnabled): ?> <!-- Ne pas oublier le ":" --&gt;
4
5 &lt;h1&gt;Liste des recettes à base de poulet&lt;/h1&gt;
6
7 &lt;?php endif; ?&gt;<!-- Ni le ";" après le endif --&gt;</pre>
```

Utilisez la condition **switch** pour optimiser le code :

- Le switch permet de simplifier le code

Exemple de code qui peut être optimiser :

```
1 <?php
2 $grade = 16;
3
4 if ($grade == 0) {
5     echo "Tu es vraiment un gros nul !!!";
6 }
7
8 elseif ($grade == 5) {
9     echo "Tu es très mauvais";
10}
11
12 elseif ($grade == 7) {
13     echo "Tu es mauvais";
14 }
15
16 elseif ($grade == 10) {
17     echo "Tu as pile poil la moyenne, c'est un peu juste...";
18 }
19
20 elseif ($grade == 12) {
21     echo "Tu es assez bon";
22 }
23
24 elseif ($grade == 16) {
25     echo "Tu te débrouilles très bien !";
26 }
27
28 elseif ($grade == 20) {
29     echo "Excellent travail, c'est parfait !";
30 }
31
32 else {
33     echo "Désolé, je n'ai pas de message à afficher pour cette note";
34 }
35 ?>
```

Même code avec le “switch” :

```
1 <?php
2 $grade = 10;
3
4 switch ($grade) // on indique sur quelle variable on travaille
5 {
6     case 0: // dans le cas où $grade vaut 0
7         echo "Tu es vraiment un gros nul !!!";
8         break;
9
10    case 5: // dans le cas où $grade vaut 5
11        echo "Tu es très mauvais";
12        break;
13
14    case 7: // dans le cas où $grade vaut 7
15        echo "Tu es mauvais";
16        break;
17
18    case 10: // etc. etc.
19        echo "Tu as pile poil la moyenne, c'est un peu juste...";
20        break;
21
22    case 12:
23        echo "Tu es assez bon";
24        break;
25
26    case 16:
27        echo "Tu te débrouilles très bien !";
28        break;
29
30    case 20:
31        echo "Excellent travail, c'est parfait !";
32        break;
33
34    default:
35        echo "Désolé, je n'ai pas de message à afficher pour cette note";
36 }
37 ?>
```

- Le dernier est moins lourd avec le switch, les cases et les breaks.
- L'avantage du switch plus besoin de mettre le ==
- Le switch ne peut tester que l'égalité

L'instruction break :

L'instruction `break` demande à PHP de sortir du `switch`.

Dès que PHP tombe sur `break`, il sort des accolades et donc il ne lit pas les `case` qui suivent.

Découvrez les ternaires : des conditions condensées

- Une forme de condition beaucoup moins fréquentes : les ternaires

Un ternaire est une condition condensée qui sert à faire deux choses sur une seule ligne :

1. Tester la valeur d'une variable dans une condition.
2. Affecter une valeur à une variable selon que la condition est vraie ou non.

Exemple :

```

1 <?php
2 $userAge = 24;
3
4 if ($userAge >= 18) {
5     $isAdult = true;
6 }
7 else {
8     $isAdult = false;
9 }
10 ?>
```

On peut écrire le même code en une seule ligne avec une structure ternaire :

```

1 <?php
2 $userAge = 24;
3
4 $isAdult = ($userAge >= 18) ? true : false;
5
6 // Ou mieux, dans ce cas précis
7 $isAdult = ($userAge >= 18);
8 ?>
```

Explication :

Ici, tout notre test précédent a été fait sur une seule ligne !

La condition testée est `$userAge >= 18`.

Si c'est vrai, alors la valeur indiquée après le point d'interrogation (ici `true`) sera affectée à la variable `$isAdult`.

Sinon, c'est la valeur qui suit le symbole `:` (ici `false`) qui sera affectée à `$isAdult`.

Affichez une liste de recettes à l'aide des boucles

Un concept très important, les boucles :

Types de boucles

- While
- For

Utilisation des tableaux :

→ Se sont des structures capables de conserver en mémoire plusieurs éléments. Et c'est grâce aux boucles qu'on va pouvoir :

- Parcourir différentes recettes (projet fil rouge)
- Les afficher à l'aide du langage HTML

→ Un tableau se déclare entre crochets []

→ Il commence à l'indice 0

→ On accède à un élément du tableau à partir de ces clés

Exemple de tableau :

```
1 <?php
2
3 // Premier utilisateur
4 $userName1 = 'Mickaël Andrieu';
5 $userEmail1 = 'mickael.andrieu@exemple.com';
6 $userPassword1 = 'S3cr3t';
7 $userAge1 = 34;
8
9 // Deuxième utilisatrice
10 $userName2 = 'Laurène Castor';
11 $userEmail2 = 'laurene.castor@exemple.com';
12 $userPassword2 = 'P4ssW0rD';
13 $userAge2 = 28;
14
15 // ... et ainsi de suite pour les autres utilisateurs.
```

On peut aussi construire des tableaux de tableaux :

```
1 <?php
2
3 $mickael = ['Mickaël Andrieu', 'mickael.andrieu@exemple.com', 'S3cr3t', 34];
4 $mathieu = ['Mathieu Nebra', 'mathieu.nebra@exemple.com', 'devine', 33];
5 $laurene = ['Laurène Castor', 'laurene.castor@exemple.com', 'P4ssW0rD', 28];
6
7 $users = [$mickael, $mathieu, $laurene];
8
9 echo $users[1][1]; // "mathieu.nebra@exemple.com"
```

Utilisez une boucle simple : `while`

C'est quoi une boucle ?

C'est une structure qui fonctionne sur le même principe qu'une condition

`if... else .`

D'ailleurs, vous allez voir qu'il y a beaucoup de similitudes avec le chapitre sur les conditions.

Concrètement, une boucle permet de répéter des instructions plusieurs fois. En clair : c'est un gain de temps, c'est très pratique, et bien souvent indispensable.

Ce qui se passe dans une boucle :

- 1) Les instructions sont exécutés dans l'ordre, de haut en bas
- 2) A la fin des instructions, on retourne à la première
- 3) On recommence à lire les instructions dans l'ordre
- 4) Et on retour à la première, etc

- Souci : dans ce schéma, la boucle ne s'arrête jamais. Les instructions sont exécutées à l'infini !
- Il faut, quelque soit le type de boucle (while ou for), indiquer une condition
- Quand la condition n'est plus remplie, on sort de la boucle

Exemple boucle simple “while” :

```
<?php
while ($isValid) {
    // instructions à exécuter dans la boucle
}
?>
```

- Les instructions qui sont répétées en boucles se trouvent entre les accolades {}

Exemple :

```
1 <?php
2 $lines = 1;
3
4 while ($lines <= 100) {
5     echo 'Je ne dois pas regarder les mouches voler quand j\'apprends le PHP.<br />';
6     $lines++; // $lines = $lines + 1
7 }
8 ?>
```

Le résultat donne cela :

```
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
Je ne dois pas regarder les mouches voler quand j'apprends le PHP.
```

Des lignes affichées grâce à une boucle PHP

→ La boucle pose la condition :

TANT QUE `$lines` est inférieur ou égal à 100.

Dans cette boucle, il y a deux instructions :

1. `echo` permet d'afficher du texte en PHP. À noter qu'il y a une balise HTML `
` à la fin : cela permet d'aller à la ligne (vu que vous connaissez le HTML, ça n'a rien de surprenant : chaque phrase sera écrite sur une seule ligne).
2. `$lines++;` est une façon plus courte d'ajouter 1 à la variable. On appelle cela **l'incrémentation** (ce nom barbare signifie tout simplement que l'on a ajouté 1 à la variable).

Chaque fois qu'on fait une boucle, la valeur de la variable augmente : 1, 2, 3, 4... 99, 100...

Dès que la variable atteint 101, on arrête la boucle.

Et voilà, on a écrit 100 lignes en un clin d'œil.

→ Il faut toujours s'assurer que la condition soit fausse au moins une fois, sinon la boucle s'exécute à l'infini !

Exemple la valeur de la variable augmente à chaque passage dans la boucle :

```
1 <?php
2 $lines = 1;
3
4 while ($lines <= 100)
5 {
6     echo 'Ceci est la ligne n°' . $lines . '<br />';
7     $lines++;
8 }
9 ?>
10
11 <!--
12
13 Ceci est la ligne n°1
14 Ceci est la ligne n°2
15 ...
16 -->
```

Exemple code fonctionnel pour faire un tableau :

```
1 <?php
2
3 $lines = 3; // nombre d'utilisateurs dans le tableau
4 $counter = 0;
5
6 while ($counter < $lines) {
7     echo $users[$counter][0] . ' ' . $users[$counter][1] . '<br />';
8     $counter++; // Ne surtout pas oublier la condition de sortie !
9 }
```

Utilisez une boucle simple : **for**

→ Définition :

for est un autre type de boucle qui a une forme un peu plus condensée et plus commode à écrire. Elle est donc fréquemment utilisée.

→ La boucle for & while donnent le même résultat et servent à la même chose : répéter des instructions en boucle.

Exemple boucle for :

```
1 <?php
2 for ($lines = 0; $lines <= 2; $lines++)
3 {
4     echo $users[$lines][0] . ' ' . $users[$lines][1] . '<br />';
5 }
6 ?>
```

Comment savoir lequel choisir entre “while” et “for” ?

while est plus simple et plus flexible : on peut faire tous les types de boucles avec, mais on peut oublier de faire certaines étapes, comme l'incrémentation de la variable.

for est bien adapté quand on doit compter le nombre de fois que l'on répète les instructions, et il permet de ne pas oublier de faire l'incrémentation pour augmenter la valeur de la variable !

Organisez vos données à l'aide des tableaux

En PHP il existe deux tableaux :

Tableau numéroté		Tableau associatif	
Clé	Valeur	Nom	Valeur
0	Cassoulet	Titre	Cassoulet
1	Couscous	Titre	Couscous
2	Escalope Milanaise	Titre	Escalope Milanaise
3	Salade César	Titre	Salade César
4	Bò bún	Titre	Bò bún
5	...	Titre	...

→ Un array numéroté à chaque case identifiée par un numéro, ce numéro est appelé clé

Exemple array `$recipes` :

```
1 <?php
2
3 $users = ['Cassoulet', 'Couscous', 'Escalope Milanaise', 'Salade César',];
4
5 // La fonction array permet aussi de créer un array
6 $users = array('Cassoulet', 'Couscous', 'Escalope Milanaise');
7 ?>
```

PHP peut écrire le numéro de la case pour nous :

```
1 <?php
2 $recipes[] = 'Cassoulet'; // Créera $recipes[0]
3 $recipes[] = 'Couscous'; // Créera $recipes[1]
4 $recipes[] = 'Escalope Milanaise'; // Créera $recipes[2]
5 ?>
```

Ici pour afficher "Couscous", on écrira ce code :

```
1 <?php
2 echo $recipes[1]; // Cela affichera : Couscous
3 ?>
```

Construire un tableau associatif :

```
1 <?php
2 // Une bien meilleure façon de stocker une recette !
3 $recipe = [
4     'title' => 'Cassoulet',
5     'recipe' => 'Etape 1 : des flageolets, Etape 2 : ...',
6     'author' => 'john.doe@example.com',
7     'enabled' => true,
8 ];
9
10 ?>
```

→ Le signe => veut dire “associé à”

1. Les **tableaux numérotés** permettent de stocker une série d'éléments du même type, comme des prénoms. Chaque élément du tableau contiendra alors un prénom.
2. Les **tableaux associatifs** permettent de découper une donnée en plusieurs sous-éléments. Par exemple, une adresse peut être découpée en nom, prénom, nom de rue, ville...

Parcourir un tableau :

→ Il y a 3 moyens pour explorer un tableau :

- 1) La boucle “for”
- 2) La boucle “foreach”
- 3) La fonction “print_r” (utilisée principalement pour le débogage)

Avec la boucle `foreach` :

→ Cette boucle passe en revue chaque ligne du tableau

→ Lors de chaque passage elle met la valeur de cette ligne dans une variable temporaire

Exemple tableau avec la boucle “foreach” :

```
1 <?php
2
3 // Déclaration du tableau des recettes
4 $recipes = [
5     ['Cassoulet','[...]', 'mickael.andrieu@example.com', true, ],
6     ['Couscous','[...]', 'mickael.andrieu@example.com', false, ],
7 ];
8
9 foreach ($recipes as $recipe) {
10     echo $recipe[0]; // Affichera Cassoulet, puis Couscous
11 }
```

→ La boucle “foreach” permet de parcourir les tableaux associatifs

Autre exemple :

```
1 <?php
2
3 $recipes = [
4   [
5     'title' => 'Cassoulet',
6     'recipe' => '',
7     'author' => 'mickael.andrieu@example.com',
8     'is_enabled' => true,
9   ],
10  [
11    'title' => 'Couscous',
12    'recipe' => '',
13    'author' => 'mickael.andrieu@example.com',
14    'is_enabled' => false,
15  ],
16  [
17    'title' => 'Escalope milanaise',
18    'recipe' => '',
19    'author' => 'mathieu.nebra@example.com',
20    'is_enabled' => true,
21  ],
22  [
23    'title' => 'Salade Romaine',
24    'recipe' => '',
25    'author' => 'laurene.castor@example.com',
26    'is_enabled' => false,
27  ],
28];
29
30 foreach($recipes as $recipe) {
31   echo $recipe['title'] . ' contribué(e) par : ' . $recipe['author'] . PHP_EOL;
32 }
```

→ On peut récupérer la valeur mais aussi récupérer la valeur de l'élément

Pour cela on écrit :

```
1 <?php foreach($recipe as $property => $PropertyValue) ?>
```

Explication :

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

1. `$property` qui contiendra la clé de l'élément en cours d'analyse (« title », « author », etc.).
2. `$PropertyValue` qui contiendra la valeur de l'élément en cours (« Cassoulet », « laurene.castor@example.com », etc.).

Afficher rapidement un tableau avec `print_r` :

- La commande “print_r” a un défaut, elle ne renvoie pas de code HTML
- Il faut utiliser la balise HTML “pre”
- Il faut mettre la balise après le echo et bien penser à la refermer (comme une balise HTML)

Exemple :

```
1 <?php
2
3 $recipes = [
4     [
5         'title' => 'Cassoulet',
6         'recipe' => '',
7         'author' => 'mickael.andrieu@example.com',
8         'is_enabled' => true,
9     ],
10    [
11        'title' => 'Couscous',
12        'recipe' => '',
13        'author' => 'mickael.andrieu@example.com',
14        'is_enabled' => false,
15    ],
16 ];
17
18 echo '<pre>';
19 print_r($recipes);
20 echo '</pre>';
```

Recherchez dans un tableau :

- Il y a 3 types de recherches, basées sur des fonctions en PHP :
 - 1) “array_key_exists” pour vérifier si une clé existe dans le tableau
 - 2) “in_array” pour vérifier si une valeur existe dans le tableau
 - 3) “array_search” pour récupérer la clé d'une valeur dans le tableau
- Pour vérifier si une clé existe on va utiliser la fonction “array_key_exists”
- Cette fonction va parcourir le tableau et vérifier s'il contient la clé recherchée
- Pour cela on doit lui donner :
 - Le nom de la clé à rechercher
 - Puis le nom du tableau dans lequel on fait la recherche

Le code se présente ainsi :

```
1 <?php array_key_exists('cle', $array); ?>
```

La fonction renvoie un booléen :

- `true` (vrai) si la clé est dans le tableau ;
- `false` (faux) si la clé ne s'y trouve pas.

Vérifiez si une valeur existe dans une tableau avec “in_array” :

→ Le principe est le même que dans “array_key_exists” mais cette fois on recherche dans les **valeurs**

`in_array` renvoie :

- `true` si la valeur se trouve dans le tableau ;
- `false` si elle ne s'y trouve pas.

Exemple code :

```
1 <?php
2 $users = [
3     'Mathieu Nebra',
4     'Mickaël Andrieu',
5     'Laurène Castor',
6 ];
7
8 if (in_array('Mathieu Nebra', $users))
9 {
10     echo 'Mathieu fait bien partie des utilisateurs enregistrés !';
11 }
12
13 if (in_array('Arlette Chabot', $users))
14 {
15     echo 'Arlette fait bien partie des utilisateurs enregistrés !';
16 }
```

Voici ce que renvoie la fonction :

- Si elle a trouvé la valeur, `array_search` renvoie la clé correspondante (dans le cas d'un tableau numéroté, la clé sera un numéro ; dans le cas d'un tableau associatif, la clé sera un nom).
- Si elle n'a pas trouvé la valeur, `array_search` renvoie `false` .

→ PHP_EOL : indique une fin de ligne, on retourne à la ligne après

Affichez des recettes (Version 2) (work) :

Exploitez toute la puissance de PHP

i Fonction

Série d'instructions qui effectue des actions et retourne une valeur.

fonction **allowRecipe**

Nom	Valeur
Recette	

Nom	Valeur
Auteur	Mathieu Nebra
Active	true

Nom	Valeur
Auteur	Mathieu Nebra
Email	mathieu@openclassrooms.com
Rôle	Admin
Âge	21

→ Les fonctions deviennent pratiques quand on a besoin de l'information plusieurs fois, qu'il y a plusieurs conditions, plusieurs boucles à respecter

C'est quoi une fonction ?

Une fonction est une série d'instructions qui effectue des actions et qui retourne une valeur.

En général, dès que vous avez besoin d'effectuer des opérations un peu longues dont vous aurez à nouveau besoin plus tard, il est conseillé de vérifier s'il n'existe pas déjà une fonction qui fait cela pour vous. Et si la fonction n'existe pas, vous avez la possibilité de la créer.

→ Les fonctions sont capables de s'adapter en fonction des informations qu'on leur envoi

→ Les fonctions permettent de récupérer des informations (date, horaire, heure actuelle, chiffrer des données, envoyer des emails, faire des recherches dans du texte, etc)

Appelez une fonction :

Pour appeler une fonction, on utilise son nom :

```
1 <?php
2 allowRecipe();
```

→ On passe en paramètre des informations, ainsi la fonction sait qu'elle doit travailler avec ces dernières

Exemple :

```
1 <?php
2 /**
3  * Il n'est pas nécessaire de déclarer une variable $recipe
4  * pour passer l'information en tant que paramètre d'une fonction.
5  */
6 allowRecipe([
7     'title' => 'Escalope milanaise',
8     'recipe' => '',
9     'author' => 'mathieu.nebra@example.com',
10    'is_enabled' => true,
11]);
```

→ Les fonctions acceptent plusieurs paramètres, pour cela il faut les séparer par des virgules :

```
1 <?php
2 fonctionImaginaire(17, 'Vert', true, 41.7);
```

Récupérez la valeur de retour de la fonction :

Il y a en effet deux types de fonctions :

1. Celles qui ne retournent aucune valeur (ça ne les empêche pas d'effectuer des actions).
2. Celles qui retournent une valeur.

→ Si une fonction retourne une valeur (exemple de “allowRecipe”), on la récupère dans une variable comme ceci :

```
1 <?php
2 $isAllowed = allowRecipe([
3     'title' => 'Escalope milanaise',
4     'recipe' => '',
5     'author' => 'mathieu.nebra@example.com',
6     'is_enabled' => true,
7 ]);
8
9 if ($isAllowed) {
10     echo 'La recette doit être affichée !';
11 } else {
12     echo 'La recette doit être cachée !';
13 }
```

Sur une ligne comme celle-ci, il se passe en fait les deux choses suivantes (dans l'ordre, et de droite à gauche) :

1. La fonction `allowRecipe` est appelée avec un tableau en paramètre.
2. Le résultat renvoyé par la fonction (lorsqu'elle a terminé) est stocké dans la variable `$isAllowed`.

La variable `$isAllowed` aura donc pour valeur `true` après l'exécution de cette ligne de code !

Utilisez les fonctions prêtes à l'emploi de PHP :

→ Couvre la quasi-totalité des besoins

Liste de fonctions en PHP :

- `str_replace` pour [rechercher et remplacer](#) des mots dans une variable ;
- `move_uploaded_file` pour [envoyer un fichier sur un serveur](#) ;
- `imagecreate` pour [créer des images miniatures](#) (aussi appelées "thumbnails") ;
- `mail` pour [envoyer un mail](#) avec PHP (très pratique pour faire une newsletter) ;
- de [nombreuses options](#) pour modifier des images, y écrire du texte, tracer des lignes, des rectangles, etc. ;
- `crypt` pour [chiffrer des mots de passe](#) ;
- `date` pour [renvoyer l'heure, la date](#), etc.

Manipulez du texte avec les fonctions :

→ Il y a de nombreuses fonctions qui permettent de manipuler le texte

3 exemples :

1. `strlen` pour calculer la longueur d'une chaîne de caractères ;
2. `str_replace` pour rechercher et remplacer une chaîne de caractères ;
3. `sprintf` pour formater une chaîne de caractères.

Calculez la longueur d'une chaîne de caractères avec “`strlen`” :

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 :

```
1 <?php
2 $recipe = 'Etape 1 : des flageolets ! Etape 2 : de la saucisse toulousaine';
3 $length = strlen($recipe);
4
5
6 echo 'La phrase ci-dessous comporte ' . $length . ' caractères :' . PHP_EOL . $recipe;
```

→ La fonction “count” permet aussi de compter le nombre d'éléments dans un tableau (en PHP une chaîne de caractères c'est un tableau de caractères)

Recherchez et remplacez une chaîne de caractères avec “`str_replace`” :

`str_replace` remplace une chaîne de caractères par une autre.

Exemple :

```
1 <?php
2 echo str_replace('c', 'C', 'le cassoulet, c\'est très bon');
```

Formatez une chaîne de caractères avec “`sprintf`” :

→ La fonction “`sprintf`” permet de formater une chaîne de caractères

→ Elle est très pratique lorsqu'on a besoin de passer plusieurs variables et elle peut remplacer la concaténation pour des raisons de lisibilité du code

Exemple :

```
1 <?php
2 $recipe = [
3     'title' => 'Salade Romaine',
4     'recipe' => 'Etape 1 : Lavez la salade ; Etape 2 : euh ...',
5     'author' => 'laurene.castor@example.com',
6 ];
7
8 echo sprintf(
9     '%s par %s : %s',
10    $recipe['title'],
11    $recipe['author'],
12    $recipe['recipe']
13 );
```

Récupérez la date :

→ Fonction “date”

Paramètres à connaître :

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

Affichez l'année :

```
1 <?php
2 $year = date('Y');
3 echo $annee;
```

Affichez l'année, la date complète et l'heure :

```
1 <?php
2 // Enregistrons les informations de date dans des variables
3
4 $day = date('d');
5 $month = date('m');
6 $year = date('Y');
7
8 $hour = date('H');
9 $minut = date('i');
10
11 // Maintenant on peut afficher ce qu'on a recueilli
12 echo 'Bonjour ! Nous sommes le ' . $day . '/' . $month . '/' . $year . ' et il est ' . $hour. ' h ' .
    $minut;
13 ?>
```

Créez vos propres fonctions :

→ Même si il existe des centaines de fonctions, quelques fois il faut écrire la fonction soi-même

Quand écrire une fonction ?

En général, si vous effectuez des opérations un peu complexes que vous pensez avoir besoin de refaire régulièrement, il est conseillé de créer une fonction.

Nous allons découvrir la création de fonctions à travers trois exemples :

1. Vérifier si la recette est valide.
2. Récupérer des recettes à afficher.
3. Récupérer le nom d'un utilisateur en fonction de l'e-mail associé à la création d'une recette.

Exemple 1 : vérifiez la validité d'une recette

→ Etablir une fonction qui retourne :

- 1) "true" si la recette est valide
- 2) "false" si la recette ne l'est pas

La fonction correspondante pour vérifier la validité d'une recette :

```
1 <?php
2
3 function isValidRecipe(array $recipe) : bool
4 {
5     if (array_key_exists('is_enabled', $recipe)) {
6         $isEnabled = $recipe['is_enabled'];
7     } else {
8         $isEnabled = false;
9     }
10
11    return $isEnabled;
12 }
```

Pour créer une fonction,

1. Vous devez taper `function` (en français, ça veut dire « fonction »).
2. Ensuite, donnez-lui un nom. Par exemple, celle-ci s'appelle `isValidRecipe`.

Ce qui est plus particulier après, c'est ce qu'on met entre parenthèses : il y a une variable. C'est le **paramètre** dont a besoin la fonction pour travailler, ici il s'agit de notre tableau.

Nous pouvons – et c'est une bonne pratique – définir le type de la variable attendue : ici, nous souhaitons un tableau donc nous préfixons la variable `$recipes` par le mot-clé `array`.

→ La fonction créée doit forcément être appelée avec un paramètre (une recette ici en exemple), sans quoi elle va générer une erreur

→ Bonne pratique indiquer le type de la fonction, ici dans l'exemple il est de type **bool**

Exemple :

```
1 <?php
2
3 // 2 exemples
4 $romanSalad = [
5     'title' => 'Salade Romaine',
6     'recipe' => 'Etape 1 : Lavez la salade ; Etape 2 : euh ...',
7     'author' => 'laurene.castor@example.com',
8     'is_enabled' => true,
9 ];
10
11 $sushis = [
12     'title' => 'Sushis',
13     'recipe' => 'Etape 1 : du saumon ; Etape 2 : du riz',
14     'author' => 'laurene.castor@example.com',
15     'is_enabled' => false,
16 ];
17
18
19 // Répond true !
20 $isRomanSaladValid = isValidRecipe($romanSalad);
21
22 // Répond false !
23 $isSushiValid = isValidRecipe($sushis);
```

Exemple 2 : récupérez les recettes “valides”

- On a créé la fonction permettant de vérifier qu'une recette est valide
- Il faut maintenant boucler sur la liste des recettes

Exemple code :

```
1 <?php
2
3 $recipes = [...]; // Les recettes
4
5 // AVANT
6
7 foreach ($recipes as $recipe) {
8     if ($recipe['is_enabled']) {
9         // echo $recipe['title'] ..
10    }
11 }
12
13 // APRES
14
15 function getRecipes(array $recipes) : array
16 {
17     $validRecipes = [];
18
19     foreach($recipes as $recipe) {
20         if (isValidRecipe($recipe)) {
21             $validRecipes[] = $recipe;
22         }
23     }
24
25     return $validRecipes;
26 }
27
28 // construire l'affichage HTML des recettes
29 foreach(getRecipes($recipes) as $recipe) {
30     // echo $recipe['title'] ..
31 }
```

- La fonction contient le code nécessaire à la récupération des recettes valides

Il n'est pas nécessaire d'assigner le résultat d'une fonction à une variable ! Nous voyons ici que nous passons directement la fonction `getRecipes` dans la boucle (nous **savons** que c'est un tableau parce que nous avons défini le type de retour).

Exemple 3 : affichez le nom de l'auteur

→ Créer une fonction pour améliorer l'affichage

Résultat (exemple code trop long à afficher ici) :

Liste des recettes de cuisine

Cassoulet

Mickaël Andrieu(34 ans)

Escalope milanaise

Mathieu Nebra(34 ans)

→ Source PHP : php.net/manual/fr

Au secours ! Mon script plante !

Liste d'erreurs que l'on va voir dans ce cours :

- “Parse error”
- “Undefined error”
- “Wrong parameter count”

→ **“Parse error” = si vous formulez mal un instruction**

Le message d'erreur obtenu ressemble à cela :

```
1 Parse error: syntax error in error.php on line 7
```

Généralement, cela veut dire que votre problème se situe à la ligne 7, mais ce n'est pas toujours le cas (trop facile, sinon). Parfois, c'est la ligne précédente qui a un problème : pensez donc à regarder autour de la ligne indiquée.

→ Une “Parse error” est en fait une instruction PHP mal formée

→ Il peut y avoir plusieurs causes

→ oublier le point-virgule à la fin de l'instruction

Exemple :

```
1 $id_news = 5
```

php

... générera une parse error. Si vous mettez le point-virgule à la fin, tout rentrera dans l'ordre !

```
1 $id_news = 5;
```

php

→ Oublier de fermer un guillemet (une apostrophe ou une parenthèse)

Par exemple :

```
1 echo "Bonjour !;
```

php

Il suffit de fermer correctement les guillemets et vous n'aurez plus de problème :

```
1 echo "Bonjour !";
```

php

→ Se tromper dans la concaténation et un oublier un point

```
1 echo "J'ai " . $age " ans";
```

php

Une fois l'erreur corrigée, ça donne :

```
1 echo "J'ai " . $age . " ans";
```

php

→ Mal fermer une accolade

Cela peut être le cas pour une structure en `if` , par exemple.

Vérifiez que vous avez correctement fermé toutes vos accolades.

→ “Undefined function” = si vous utilisez une fonction non reconnue

→ Autre erreur classique, la fonction inconnue

On obtient ce message d'erreur :

```
1 Fatal Error: Call to undefined function: is_valid_recipe() in fichier.php on line 27
```

Là, il faut comprendre qu'on a utilisé une fonction qui n'existe pas !

Deux hypothèses :

1. Soit **la fonction n'existe vraiment pas**. Vous avez probablement fait une faute de frappe, vérifiez si une fonction à l'orthographe similaire existe.
2. Soit la fonction existe vraiment, mais PHP ne la reconnaît pas. C'est parce que cette fonction se trouve dans **une extension de PHP que vous n'avez pas activée**. Par exemple, si vous essayez d'utiliser la fonction `imagepng` alors que vous n'avez pas activé la bibliothèque nécessaire pour les images en PHP, on vous dira que la fonction n'existe pas. Activez la bibliothèque qui utilise la fonction, et tout sera réglé.

→ **“Wrong parameter count” = si vous entrez un nombre incorrect de paramètres pour une fonction**

On obtient une erreur de ce type :

```
1 Warning: Wrong parameter count for fonction() in fichier.php on line 112
```

Cela signifie que :

- vous avez oublié des paramètres pour la fonction ;
- ou même que vous en avez trop mis.

Exemple avec la fonction “fopen”, elle requiert au minimum deux paramètres :

- 1) Le premier pour le nom du fichier à ouvrir
- 2) Et le second pour le mode d'ouverture (en lecture seule, écriture, etc.)

→ Si l'on met que le nom du fichier à ouvrir comme ceci :

```
1 $fichier = fopen("fichier.txt");
```

→ On obtiendra l'erreur “Wrong parameter count”, il faut donc rajouter le paramètre manquant :

```
1 $fichier = fopen("fichier.txt", "r");
```

Découvrez ces erreurs plus rares :

Quelques exemples d'erreurs plus rares :

- 1) "Headers already sent by..."
- 2) "L'image contient des erreurs"
- 3) "Maximum execution time exceeded"

→ **"Headers already sent by..." = si vous écrivez du code au mauvais endroit**

Une erreur classique quand on travaille avec les sessions ou avec les cookies :

```
1 Cannot modify header information - headers already sent by ...
```

Les "**headers**" sont des informations d'en-tête qui sont envoyées avant toute chose au navigateur du visiteur.

Elles permettent de dire :

→ "Ce que tu vas recevoir est une page HTML"

ou

→ "Ce que tu vas recevoir est une image PNG", ou encore "Inscris comme un cookie"

Toutes ces choses-là doivent être effectuées avant que le moindre code HTML ne soit envoyé.

→ En PHP, la fonction qui permet d'envoyer des informations de "headers" s'appelle "**header()**"

Il y a d'autres fonctions envoient des "headers" toutes seules, par exemple :

- "session_start()"
- "setcookie()"

Ce que vous devez retenir, c'est que chacune de ces fonctions doit être **utilisée au tout début de votre code PHP**. Il ne faut RIEN mettre avant, sinon ça provoquera l'erreur « Headers already sent by... ».

Exemple de code qui génère l'erreur :

```
1 <html>
2 <?php session_start(); ?>
```

→ On a mis du HTML avant le “`session_start()`” et cela a provoqué l'erreur

La bonne écriture est celle-ci :

```
1 <?php session_start(); ?>
2 <html>
```

→ **“L'image contient des erreurs” = si vous avez fait une erreur d'utilisation de la librairie GD**

C'est le navigateur qui vous donne ce message d'erreur, et non pas PHP.

Ce message survient lorsque **vous travaillez avec la bibliothèque GD**.

→ [Bibliothèque GD](#)

Si vous avez fait une erreur dans votre code (par exemple une banale « parse error »), cette erreur sera **inscrite dans l'image**. Du coup, l'image ne sera pas valide et le navigateur ne pourra pas l'afficher.

→ Comment faire pour faire "apparaître" l'erreur ?

On peut supprimer la ligne suivante dans le code :

```
1 <?php header ("Content-type: image/png"); ?>
```

L'erreur apparaîtra à la place du message « L'image contient des erreurs » ;

- vous pouvez aussi afficher le code source de l'image (comme si vous alliez regarder la source HTML de la page, sauf que là, il s'agit d'une image).

Dans les deux cas, vous verrez le message d'erreur apparaître. À partir de là, il ne vous restera plus qu'à corriger le bug !

→ “**Maximum execution time exceeded**” = si vous avez fait une boucle infinie

→ Arrive le plus souvent à cause d'une boucle infinie :

```
1 Fatal error: Maximum execution time exceeded in fichier.php on line 57
```

Heureusement, PHP limite le temps d'exécution d'une page PHP à 30 secondes par défaut.

Si une page met plus de 30 secondes à se générer, PHP arrête tout en signalant que c'est trop long. Et il fait bien, parce que sinon cela pourrait ralentir tout le serveur et rendre votre site inaccessible !

Exemple boucle “while” qui ne s'arrête jamais :

```
1 <?php
2 $counter = 5;
3 while ($counter == 5)
4 {
5     echo 'Zéro ';
6 }
7 ?>
```

→ \$counter ici vaut toujours 5, donc boucle infinie

Organisez les pages de votre site en blocs fonctionnels

→ Le projet va contenir plusieurs pages :

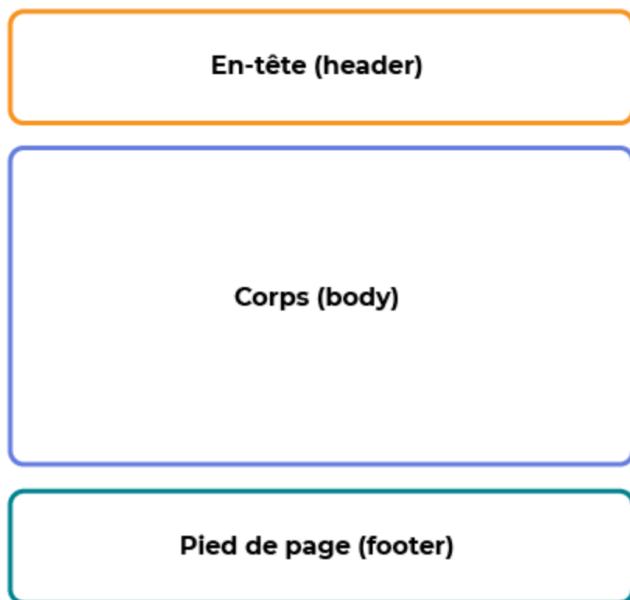
- Une page “recettes”
- Une page d'édition et de création de recette
- Une page avec un formulaire de contact

→ Pour accéder à ces pages, il faudra nécessairement quelques liens HTML regroupés dans un menu principal

Une des fonctionnalités les plus simples et les plus utiles de PHP est l'**inclusion de pages**.

On peut très facilement inclure toute une page, ou un bout de page à l'intérieur d'une autre. Cela vous évitera d'avoir à copier le même code HTML plusieurs fois.

Visualiser le découpage d'une page web :



D'une page à l'autre, ce site contiendra à chaque fois le même code pour l'en-tête, le menu et le pied de page ! En effet, seul le contenu du corps change, en temps normal.

Pensez votre contenu web en blocs fonctionnels :

En PHP, nous pouvons facilement insérer d'autres pages. Mais on peut aussi insérer des morceaux de pages, à l'intérieur d'une page.

Le principe de fonctionnement des **inclusions** en PHP est plutôt simple à comprendre. Vous avez un site web composé de, disons, vingt pages. Sur chaque page, il y a un menu, toujours le même. Pourquoi ne pas écrire ce menu (et seulement lui) une seule fois dans une page `header.php` ?

En PHP, vous allez pouvoir inclure votre menu sur toutes vos pages.

Lorsque vous voudrez modifier votre menu, vous n'aurez qu'à modifier `header.php`, et l'ensemble des pages de votre site web sera automatiquement mis à jour !

Réorganisez votre site de recettes :

(voir travail dans sur Git, dans le dossier [Site de recettes de cuisine PHP et HTML](#))

Ecoutez la requête de vos utilisateurs grâce aux URL

→ Définition URL : Uniform Resource Locator, désigne une adresse web

Envoyez des paramètres dans l'URL :

→ Les URL permettent de transmettre des informations

Formez une URL pour envoyer des paramètres :

Imaginons :

- votre site s'appelle `monsite.com` ;
- et possède une page PHP de contact : `contact.php` .

Pour accéder à la page de contact, vous devez aller à l'URL suivante :

`http://www.monsite.com/contact.php`

Jusque-là, rien de bien nouveau. Ce que je vous propose d'apprendre à faire, c'est d'**envoyer** des informations à la page `contact.php` .

Ces informations vont figurer à la fin de l'URL, comme ceci :

`http://www.monsite.com/contact.php?nom=Dupont&prenom=Jean`

Ce que vous voyez après le point d'interrogation, ce sont des **paramètres** que l'on envoie à la page PHP. Celle-ci peut récupérer ces informations dans des variables.

Créez un lien avec des paramètres :

- `index.php` (l'accueil) ;
- `bonjour.php` .

Nous voulons faire un lien de `index.php` à `bonjour.php` pour transmettre des informations dans l'URL :



Créez un formulaire avec la méthode HTTP GET :

La deuxième solution pour faire passer des informations dans l'URL, c'est de proposer à l'utilisateur de soumettre un formulaire avec la méthode HTTP GET.

Nous pouvons faire cela avec une balise `form` qui a pour attribut `method` la valeur **GET**.

```
1 <form action="contact.php" method="GET">
2     <!-- données à faire passer à l'aide d'inputs -->
3     <input name="nom">
4     <input name="prenom">
5 </form>
```

html

Les données soumises à l'aide du formulaire se retrouveront dans l'URL, comme pour un lien.

Récupérez les paramètres PHP :

Formulaire de contact :

```
1 <Form action="submit_contact.php" method="GET">
2     <div>
3         <label for="email">Email</label>
4         <input type="email" name="email">
5     </div>
6     <div>
7         <label for="message">Votre message</label>
8         <textarea placeholder="Exprimez vous" name="message"></textarea>
9     </div>
10    <button type="submit">Envoyer</button>
11 </form>
```

Lors de la soumission, une variable **superglobale** appelée `$_GET` va contenir les données envoyées. Il s'agit d'un tableau associatif dont les clés correspondent aux noms des paramètres envoyés dans l'URL :

→ Attention, les visiteurs peuvent trafiquer les URL

Testez la présence d'un paramètre :

Allons plus loin. Qu'est-ce qui empêche le visiteur de supprimer tous les paramètres de l'URL ? Par exemple, il peut très bien tenter d'accéder à :

`http://localhost/submit_contact.php`

Que va afficher la page `submit_contact.php` ?

Faites le test ! Elle va afficher quelque chose comme :

The screenshot shows a terminal window with two lines of output:
Notice: Undefined index: email in `submit_contact.php` on line 15
Warning: Undefined array key "email" in `submit_contact.php` on line 15



Que s'est-il passé ?

On a essayé d'afficher la valeur de `$_GET['email']` et celle de `$_GET['message']`.

Mais comme on vient de les supprimer de l'URL, ces variables n'ont pas été créées, et donc elles n'existent pas ! PHP nous avertit qu'on essaie d'utiliser des variables qui n'existent pas, d'où les « `Undefined index` ».

→ La fonction “`isSet()`” teste si une variable existe

Exemple de code utilisant la fonction “`isSet()`” :

```
1 <?php
2
3 if (!isSet($_GET['email']) || !isSet($_GET['message']))
4 {
5     echo('Il faut un email et un message pour soumettre le formulaire.');
6
7     // Arrête l'exécution de PHP
8     return;
9 }
10
11 ?>
```

Il teste si les variables `$_GET['email']` et `$_GET['message']` existent.

1. Si elles existent, on affiche la confirmation de prise en compte du message.
2. S'il nous manque une des variables (ou les deux), on affiche un message d'erreur : "Il faut un e-mail et un message pour soumettre le formulaire", et on arrête l'exécution de la page.

Essayez maintenant d'accéder à la page `submit_contact.php` sans les paramètres.

Vous allez voir qu'on gère bien le cas où le visiteur aurait retiré les paramètres de l'URL :

localhost:1234/P3C1/submit_contact.php

Il faut un email et un message pour soumettre le formulaire.

- Toujours gérer le cas si un utilisateur mal intentionné tente d'accéder ou de modifier les paramètres

Contrôlez la valeur des paramètres :

- Une fois les valeurs soumises et correctement récupérées, il faut vérifier qu'elles correspondent à ce qui est attendu

Ex : l'utilisateur s'est trompé et que l'email envoyé ne soit pas valide, on ne pourra pas le recontacter pour répondre à son besoin !

Il faut donc :

- Contrôler si l'email passé est bien valide grâce à la fonction “**filter_var**”
- Vérifier que le message n'est pas vide grâce à la fonction “**empty**”

Le code correspondant est :

```
1 <?php
2 if (
3     (!isset($_GET['email']) || !filter_var($_GET['email'], FILTER_VALIDATE_EMAIL))
4     || (!isset($_GET['message']) || empty($_GET['message'])))
5 )
6 {
7     echo('Il faut un email et un message valides pour soumettre le formulaire.');
8     return;
9 }
```

- Toujours vérifier que tous les paramètres attendus soient bien là
- Vérifier qu'ils contiennent des valeurs correctes, comprises dans des intervalles raisonnables

Administrez des formulaires de façon sécurisée

- Les formulaires constituent le principal moyen pour les visiteurs d'entrer des informations sur notre site

La base du formulaire :

- On utilise la balise “<form>” pour insérer un formulaire

Exemple comment on utilise la balise <form> :

```
1 <!-- index.php -->
2 <form method="post" action="submit_form.php">
3
4 <p>
5     On insérera ici les éléments de notre formulaire.
6 </p>
7
8 </form>
```

Important :

Il y a deux attributs très importants à connaître pour la balise <form> :

1. La méthode : `method`.
2. Et la cible : `action`.

→ Privilégez la méthode “post” pour envoyer les données du formulaire

Il existe deux méthodes pour envoyer des données d'un formulaire :

1. `get` : les données transiteront par l'URL, comme on l'a appris précédemment. On pourra les récupérer grâce au tableau (array) `$_GET`. Cette méthode est assez peu utilisée car on ne peut pas envoyer beaucoup d'informations dans l'URL (je vous disais dans le chapitre précédent qu'il était préférable de ne pas dépasser 256 caractères).
2. `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.**

→ Bonne pratique : privilégez la méthode “Post” pour les formulaires

Choisissez la page appelée par le formulaire en définissant 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.

Selon la méthode utilisée, ce ne sera pas la même variable spéciale qui aura accès aux données :

- Si la méthode est `GET` (comme dans le chapitre précédent), alors c'est la **supervariable `$_GET`** qui aura les données ;
- Si la méthode est `POST` (bonne pratique), alors c'est la **supervariable `$_POST`** qui recevra les données.

Retenez donc bien que vous travaillez normalement sur deux pages différentes :

1. La page qui contient le formulaire (`form.php` dans notre exemple) ;
2. Et celle qui reçoit les données du formulaire pour les traiter (`submit_form.php`).

Ajoutez un ou plusieurs champs input avec un attribut name :

→ La variable “`$_GET`” sera complétée avec la valeur de l’attribut “name” en tant que clé et aura pour valeur ce qui aura été soumis par l’utilisateur

Code :

```
1 <input type="text" name="nom" value="Mateo21" />
2
3 <?php
4
5 // Après soumission du formulaire
6 echo $_GET['nom']; // "Mateo21"
7
8 // OU
9
10 echo $_POST['nom']; // "Mateo21"
```

Il y aura autant de clés dans le tableau `$_GET` que de champs avec un attribut name dans le formulaire soumis.

Faites attention avec les champs cachés :

→ Les champs cachés constituent un type de champs à part

En quoi cela consiste-t-il ?

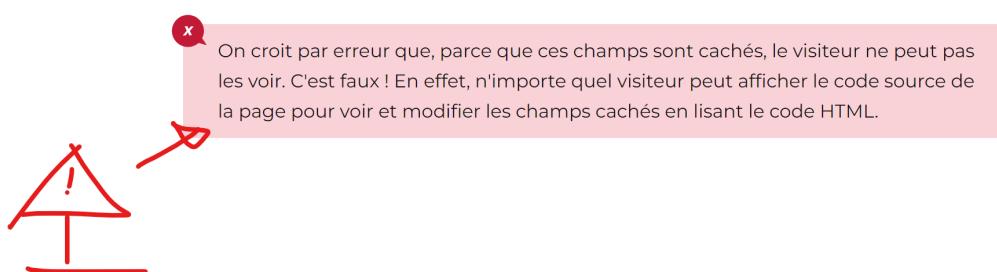
C'est un code dans votre formulaire qui n'apparaîtra pas aux yeux du visiteur, mais qui va quand même créer une variable avec une valeur. On peut s'en servir pour transmettre des informations fixes.

Exemple : On doit “retenir” le pseudo du visiteur “Mateo21”, on va taper ce code :

```
1 <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.



Ne faites jamais confiance aux données reçues : la faille XSS

→ La faille XSS : attention au code HTML que l'on reçoit !

→ La sécurité des applications web est très importante. On peut en savoir plus avec le cours “[Sécurisez vos applications web avec l'OWASP](#)”.

Qu'est-ce que la faille XSS ?

La **faille XSS** (pour **cross-site scripting**) est vieille comme le monde (ehu, comme le Web) 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.

Sécurisez votre code en bloquant l'exécution de code Javascript

→ Pour ignorer le code HTML, on utilise la fonction “htmlspecialchars”

On dira dans ce cas qu'on "échappe" le code HTML, c'est-à-dire que la fonction JavaScript `alert()` n'en tiendra pas compte.

→ Cette fonction va transformer les chevrons des balises HTML < & > en “<” et “>” respectivement

Cela provoquera l'affichage de la balise plutôt que son exécution :

```
1 <p><b>Message</b> : <?php echo htmlspecialchars($_POST['message']); ?></p>
```

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

```
1 <p><b>Message</b> : &lt;strong&gt;Badaboum&lt;/strong&gt; ?></p>
```

Il faut penser à utiliser cette fonction sur tous les textes envoyés par l'utilisateur qui sont susceptibles d'être affichés sur une page web.

Bref, tout ce qui est affiché et qui vient, à la base, d'un visiteur, vous devez penser à le protéger avec `htmlspecialchars`.

→ Si on préfère retirer les balises HTML que le visiteur a tenté d'envoyer plutôt que de les afficher on peut utiliser la fonction "strip_tags"

Activez le partage de fichiers

→ Pour qu'un utilisateur puisse soumettre des fichiers, on va utiliser la supervariable `"$_FILES"`

Comment cela fonctionne :

1. Le visiteur arrive sur votre formulaire et le remplit (en indiquant le fichier à envoyer).
Une simple page HTML suffit pour créer le formulaire.
2. PHP réceptionne les données du formulaire et, s'il y a des fichiers dedans, il les « enregistre » dans un des dossiers du serveur.

Paramétrez le formulaire d'envoi de fichier

→ Pour cela il faut ajouter l'attribut `enctype="multipart/form-data"` à la balise `<form>`.

→ Grâce à "enctype" le navigateur du visiteur sait qu'il s'apprête à envoyer des fichiers.

Le <form> ressemble à cela :

```
1 <form action="submit_contact.php" method="POST" enctype="multipart/form-data">
2     <!-- champs de formulaire -->
3 </form>
```

A l'intérieur il faudra ajouter une balise simple : `<input type="file" />`

Traitez l'envoi en PHP

- Au moment où PHP s'exécute, le fichier a été envoyé sur le serveur mais il est stocké dans un dossier temporaire, on doit décider si on accepte définitivement le fichier ou non
- Si le fichier est bon on l'accepte grâce “move_uploaded_file”
- Pour chaque fichier envoyé, une variable “\$_FILES['nom_du_fichier']” est créée
- Dans notre exemple, la variable s'appellera “\$_FILES['screenshot']”

Cette variable est un tableau qui contient plusieurs informations sur le fichier :

Variable	Signification
<code>\$_FILES['screenshot']['name']</code>	Contient le nom du fichier envoyé par le visiteur.
<code>\$_FILES['screenshot']['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['screenshot']['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. La taille de l'envoi est limitée par PHP. Par défaut, impossible d'uploader des fichiers de plus de 8 Mo.
<code>\$_FILES['screenshot']['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['screenshot']['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.

Si vous avez mis un second champ d'envoi de fichier dans votre formulaire, il y aura une seconde variable `$_FILES['nom_de_votre_autre_champ']` découpée de la même manière que le tableau qu'on vient de voir ici.

`$_FILES['nom_de_votre_autre_champ']['size']` contiendra donc la taille du second fichier, et ainsi de suite.

→ Pour tester qu'un fichier a bien été envoyé on utilise “`$_FILES['screenshot']`” avec la fonction “`isset()`”

→ Pour vérifier que le fichier ne dépasse pas 1 MO (exemple ici) on utilise “`$_FILES['screenshot']['size']`”

→ Pour vérifier l'extension du fichier dans une variable on procède ainsi :

La fonction `pathinfo` renvoie un tableau (array) contenant entre autres l'extension du fichier dans `$fileInfo['extension']`.

On stocke ça dans une variable `$extension`.

Une fois l'extension récupérée, on peut la comparer à un tableau d'extensions autorisées, et vérifier si l'extension récupérée fait bien partie des extensions autorisées à l'aide de la fonction `in_array()`.

→ Pour valider l'upload du fichier on procède ainsi :

Si tout est bon, on accepte le fichier en appelant `move_uploaded_file()`.

Cette fonction prend deux paramètres :

1. Le nom temporaire du fichier (on l'a avec `$_FILES['screenshot']['tmp_name']`).
2. Le chemin qui est le nom sous lequel sera stocké le fichier de façon définitive. On peut utiliser le nom d'origine du fichier `$_FILES['screenshot']['name']` ou générer un nom au hasard.

Implémentez un système de connexion

Protégez le contenu d'une page par un mot de passe

→ Pour que les contributeurs et contributrices de recettes puissent se connecter sur le site et être reconnus, il faut un formulaire de connexion avec email et mot de passe

Travaillez au brouillon :

1. Posez le problème

On doit soumettre un e-mail et un mot de passe dans un formulaire de connexion.

- Si le formulaire est valide, nous affichons un message de succès, et sinon un message d'erreur. La liste de recettes n'est affichée qu'à un utilisateur qui s'est connecté avec succès.

2. Schématissez le code

Pour que l'utilisateur puisse entrer le mot de passe, le plus simple est de créer un formulaire. Celui-ci sera directement intégré dans la page d'accueil du site telle que nous la connaissons déjà.

Trois situations peuvent survenir :

1. Vous n'êtes **pas connecté** : auquel cas, le formulaire de contact s'affiche, et la liste des recettes ne s'affiche pas.
2. Vous avez soumis le formulaire avec le **bon mot de passe** pour l'utilisateur : le message de succès s'affiche, le formulaire de connexion ne s'affiche pas et les recettes s'affichent.
3. Vous avez soumis le formulaire avec le **mauvais mot de passe** pour l'utilisateur : le message d'erreur s'affiche, le formulaire de connexion s'affiche et les recettes ne s'affichent pas.

Vous devez donc créer une nouvelle page et adapter la page d'accueil :

- **login.php** : contient un simple formulaire comme vous savez les faire ;
- **home.php** : qui doit maintenant inclure une formulaire de connexion et une condition sur l'affichage des recettes.

3. Mobilisez les connaissances requises

Nous avons détaillé les connaissances requises au début de ce chapitre. Vous allez voir que ce n'est qu'une simple application pratique de ce que vous connaissez déjà, mais cela sera une bonne occasion de vous entraîner.

Bon ! On a préparé le terrain ensemble ; maintenant, vous savez tout ce qu'il faut pour réaliser le script !

Vous êtes normalement capable de trouver le code à écrire par vous-même, et c'est ce que je vous invite à faire. Ça ne marchera peut-être pas du premier coup, mais ne vous en faites pas : c'est le métier qui rentre !

→ On commence par implémenter la page login.php, pour la partie mot de passe on utilise "password"

→ Un bon mot de passe est long et contient des caractères différents, minuscules, majuscules, chiffres, etc

Pour moi, un bon mot de passe c'est long, avec plein de caractères bizarres, des majuscules, des minuscules, des chiffres, etc. Par exemple, `k7hYTe40Lm8Mf` est un bon mot de passe qui a peu de chances d'être trouvé « par hasard ».

Conservez des données grâce aux sessions et aux cookies

→ Les sessions permettent de conserver des variables sur toutes les pages de notre site

Comment sont gérées les sessions PHP ?

Etape 1 : création d'une session unique

- 1) Un visiteur arrive sur le site
- 2) On demande à créer une session pour lui
- 3) PHP génère alors un numéro unique

→ Ce numéro est souvent très grand

Exemple :

a02bbfffc6198e6e0cc2715047bc3766f.

Ce numéro sert d'identifiant ; c'est ce qu'on appelle un « ID de session » ou `PHPSESSID`.

PHP transmet automatiquement cet ID de page en page, en utilisant généralement un cookie.

Etape 2 : création de variables pour la session

→ Une fois la session générée, on peut créer une infinité de variables de session pour nos besoins

Par exemple :

- Une variable qui contient le nom du visiteur : “`$_SESSION['nom']`”
- Une autre qui contient son prénom : “`$_SESSION['prenom']`”

Le serveur conserve ces variables même lorsque la page PHP a fini d'être générée. Autrement dit : quelle que soit la page de votre site, vous pourrez récupérer le nom et le prénom du visiteur via la superglobale `$_SESSION` !

Etape 3 : suppression de la session

- Lorsque le visiteur se déconnecte du site, la session est fermée et PHP “oublie” alors toutes les variables de session qui ont été créées.
- Il est difficile de savoir précisément quand un visiteur quitte le site
- Soit le visiteur clique sur le bouton “déconnexion”
- Soit on attend quelques minutes d'inactivité pour le déconnecter automatiquement
- On parle alors de “timeout” (c'est ce qui arrive le plus souvent, l'utilisateur est déconnecté par un “timeout”)

Pour activer ou détruire une sessions, deux fonctions à connaître :

1. `session_start()` : démarre le système de sessions. Si le visiteur vient d'arriver sur le site, alors un numéro de session est généré pour lui.
2. `session_destroy()` : ferme la session du visiteur. Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page de votre site pendant plusieurs minutes (c'est le timeout), mais vous pouvez aussi créer une page « Déconnexion » si le visiteur souhaite se déconnecter manuellement.

Il faut appeler `session_start()` sur chacune de vos pages AVANT d'écrire le moindre code HTML ou PHP (avant même la balise `<!DOCTYPE>`).

Si vous oubliez de lancer `session_start()`, vous ne pourrez pas accéder à la variable **superglobale** `$_SESSION`.

Mettez en place une session :

Vidéo dans le cours, suivi [ici](#).

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()`.

- Les sessions peuvent servir à de nombreux cas sur un site

Exemples :

- Imaginez un script qui demande un identifiant et un mot de passe pour qu'un visiteur puisse se « connecter » (s'authentifier). On peut enregistrer ces informations dans des variables de session et se souvenir de l'identifiant du visiteur sur toutes les pages du site !
- Puisqu'on retient son identifiant et que la variable de session n'est créée que s'il a réussi à s'authentifier, on peut l'utiliser pour restreindre certaines pages de notre site à certains visiteurs uniquement. Cela permet de créer toute une zone d'administration sécurisée : si la variable de session login existe, on affiche le contenu, sinon on affiche une erreur. Cela devrait vous rappeler l'exercice sur la protection d'une page par mot de passe, sauf qu'ici, on peut se servir des sessions pour protéger automatiquement plusieurs pages.
- On se sert activement des sessions sur les sites de vente en ligne. Cela permet de gérer un « panier » : on retient les produits que commande le client quelle que soit la page où il est. Lorsqu'il valide sa commande, on récupère ces informations et... on le fait payer.

Conservez les données grâce aux cookies / comprenez le fonctionnement d'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.

→ Les cookies ne sont pas dangereux, ce sont juste des petits fichiers textes qui permettent de retenir des informations

Écrivez un cookie

→ Comme une variable, un cookie à un nom et une valeur

→ Dans notre exemple, le cookie utilisateur aura le nom “LOGGED_USER”

Pour écrire un cookie, on utilise la fonction PHP **setcookie** (qui signifie « Placer un cookie », en anglais).

→ “setcookie”

On lui donne en général 3 paramètres, dans l'ordre suivant :

- 1) Le **nom** du cookie (par exemple : LOGGED_USER)
- 2) La **valeur** du cookie (par exemple : utilisateur@example.com)
- 3) La **date d'expiration** du cookie, sous forme de “timestamp” (exemple : 1090521508)

Le paramètre correspondant à la date d'expiration du cookie mérite quelques explications. Il s'agit d'un **timestamp**, c'est-à-dire du nombre de secondes écoulées depuis le 1er janvier 1970. Le timestamp est une valeur qui augmente de 1 toutes les secondes. Pour obtenir le timestamp actuel, on fait appel à la fonction **time()**. Pour définir une date d'expiration du cookie, il faut ajouter au « moment actuel » le nombre de secondes au bout duquel il doit expirer.

Sécurisez un cookie avec les propriétés “httpOnly” et “secure”

Sans rentrer dans les détails, cela rendra votre cookie inaccessible en JavaScript sur tous les navigateurs qui supportent cette option (c'est le cas de tous les navigateurs récents). Cette option permet de réduire drastiquement les risques de faille XSS sur votre site, au cas où vous auriez oublié d'utiliser **htmlspecialchars** à un moment.

Exemple comment créer un cookie de façon sécurisée :

```
1 <?php
2 // retenir l'email de la personne connectée pendant 1 an
3 setcookie(
4     'LOGGED_USER',
5     'utilisateur@example.com',
6     [
7         'expires' => time() + 365*24*3600,
8         'secure' => true,
9         'httponly' => true,
10    ]
11 );
```

→ En écrivant les cookies de cette façon, vous diminuez le risque qu'un jour l'un de vos visiteurs puisse se faire voler le contenu d'un cookies à cause d'une faille XSS.

Ne placez donc **JAMAIS** le moindre code HTML avant d'utiliser **setcookie** !

La plupart des gens qui ont des problèmes avec **setcookie** ont fait cette erreur.

Affichez et récupérez un cookie :

Avant de commencer à travailler sur une page, PHP lit les cookies du client pour récupérer toutes les informations qu'ils contiennent. Ces informations sont placées dans la **superglobale** `$_COOKIE` sous forme d'un tableau (array), comme d'habitude.

Si on veut ressortir l'email du visiteur qu'on avait inscrit dans un cookie, il suffit d'écrire : `$_cookie['LOGGED_USER']`

Le code PHP se présente ainsi :

```
1 Bonjour <?php echo $_COOKIE['LOGGED_USER']; ?> !
```

→ L'avantage c'est que les superglobales sont accessibles

À noter que si le cookie n'existe pas, la variable superglobale n'existe pas. Il faut donc faire un `isset` pour vérifier si le cookie existe ou non.

Modifiez un cookie existant :

→ Pour modifier un cookie déjà existant il faut refaire appel à “setcookie” **en gardant le même nom de cookie**, ce qui “écrasera l'ancien”

Exemple :

```
1 <?php
2 setcookie(
3     'LOGGED_USER',
4     'laurene.castor@example.com',
5     [
6         'expires' => time() + 365*24*3600,
7         'secure' => true,
8         'httponly' => true,
9     ]
10 );
```

Travaillez avec une base de données

- La base de données permet d'enregistrer des données de façon organisées et hiérarchisées
- Les variables restent en mémoire seulement le temps de génération de la page

Découvrez le langage SQL et les bases de données :

- La base de données (BDD) est un système qui enregistre des informations
- Les données sont toujours organisées

Nous allons utiliser le **Système de Gestion de Base de Données (SGBD)** avec MySQL, mais sachez que l'essentiel de ce que vous allez apprendre fonctionnera de la même manière avec un autre SGBD. Cette partie est construite afin que vous ayez le moins de choses possible à apprendre de nouveau si vous choisissez de changer de SGBD.

Donnez des ordres au SGBD en langage SQL :

- Le langage SQL est un langage standard

Exemple langage SQL :

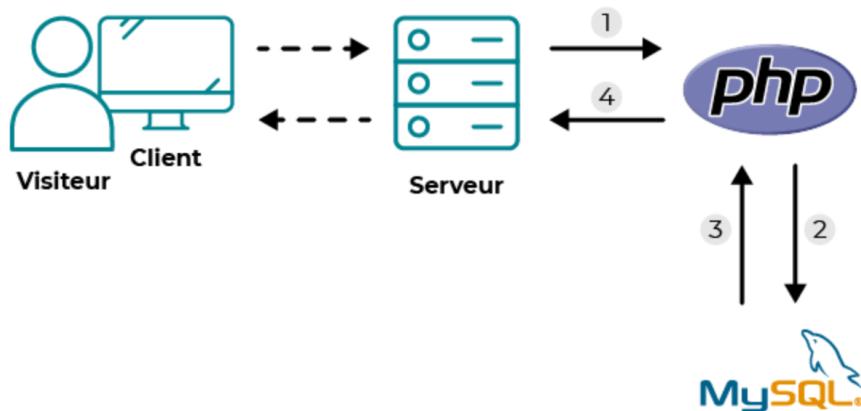
```
1 SELECT id, auteur, message, datemsg FROM livre ORDER BY datemsg DESC
```

MySQL et le SQL sont des compétences très précieuses et utiles tout au long de votre carrière en développement web, et si vous souhaitez progresser sur le sujet, n'hésitez pas à suivre le cours intitulé "**Administrez vos bases de données avec MySQL**" en complément de ce cours.

Comprenez comment PHP fait la jonction entre nous et MySQL :

- PHP va faire l'intermédiaire entre nous et MySQL

Schéma détaillant ces informations :



Explications de ce qu'il se passe lorsque le serveur a reçu une demande d'un client qui veut poster un message :

1. Le serveur utilise toujours PHP, il lui fait donc passer le message.
2. PHP effectue les actions demandées et se rend compte qu'il a besoin de MySQL. En effet, le code PHP contient à un endroit "Va demander à MySQL d'enregistrer ce message". Il fait donc passer le travail à MySQL.
3. MySQL fait le travail que PHP lui a soumis et lui répond "OK, c'est bon !".
4. PHP renvoie au serveur que MySQL a bien fait ce qui lui était demandé.

Structurez votre base de données :

- Avec les bases de données, il faut utiliser un vocabulaire précis
- Imaginer l'image d'une armoire

Imaginez ce qui suit :

- L'armoire est appelée "**la base**" dans le langage SQL. C'est le gros meuble dans lequel les secrétaires ont l'habitude de classer les informations
- Dans une armoire, il y a plusieurs tiroirs. Un tiroir, en SQL, c'est ce qu'on appelle "**une table**". Chaque tiroir contient des données différentes. Exemple : un tiroir contient les pseudonymes et informations sur un visiteurs, un autre contient les messages postés sur un forum
- Les tables sont l'endroit où sont enregistrées les données, sous la forme d'un tableau. Dans ce tableau, les colonnes sont appelées des "**champs**" et les lignes sont appelées des "**entrées**"

- Une table est représentée sous forme d'un tableau

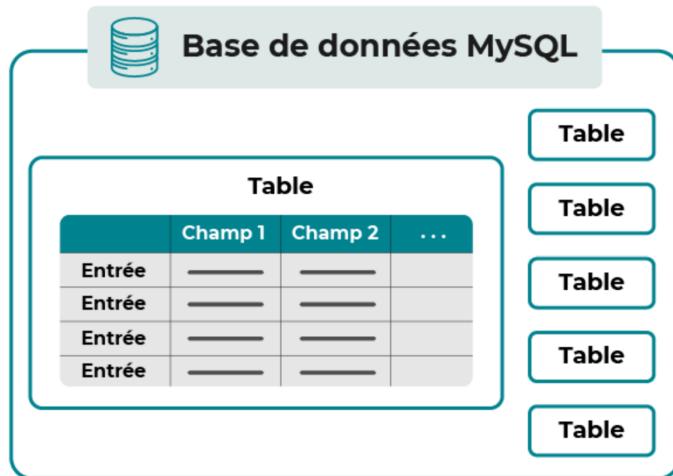
Exemple :

Number	Full name	Email	Age	Password
1	Mathieu Nebra	mathieu.nebra@exemple.com	34	P4ssW0rd
2	Laurène Castor	laurene.castor@exemple.com	28	jm_les_cookies
3	Mickaël Andrieu	mickael.andrieu@exemple.com	34	s3cr3t
4	Vous	vous@exemple.com	29	123456
...

Très souvent, on crée un champ "Number", aussi appelé "ID" (identifiant).

Comme nous le verrons plus tard, il est très pratique de numérotter ses entrées, même si ce n'est pas obligatoire.

Le schéma décrivant toutes ces "notions" :



→ La base de données contient plusieurs tables (on peut en mettre autant qu'on veut)

Exemples de nom de tables qu'on va créer pour le projet fil rouge :

- **Users** : stocke tous les comptes utilisateur du site
- **Recipes** : stocke toutes les recettes du site
- **Comments** : stocke tous les commentaires liés aux recettes

Enregistrez les données :

- MySQL enregistre les données dans des fichiers
- Les fichiers sont enregistrés sur le disque dur, il ne faut pas les modifier directement
- MySQL se charge d'extraire et de modifier les informations dans les fichiers

Dans mon cas, j'utilise MAMP, donc les informations sont stockées ici : C:\MAMP\db\mysql\mysql (pas sûre)

Dans la pratique, **on n'ira jamais toucher à ces fichiers directement**. On demandera TOUJOURS à MySQL d'enregistrer, ou d'aller lire des choses. Après, c'est lui qui se débrouille pour classer ça comme il veut dans ses fichiers.

Mettez en place une base de données avec phpMyAdmin

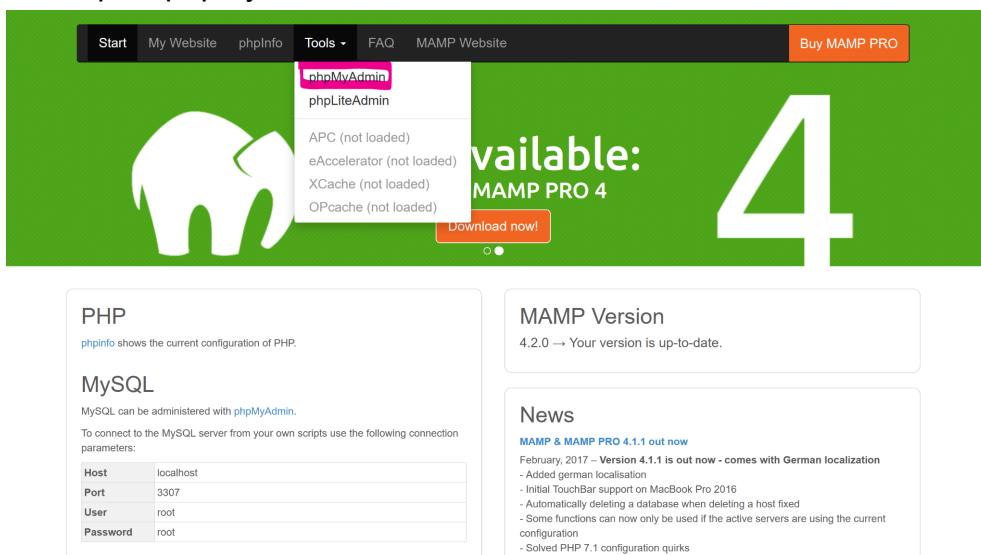
- Utilisation de phpMyAdmin

Créez une table pour les recettes :

Pour accéder à phpMyAdmin il faut aller sur :

<http://localhost/MAMP/> (prendre sa version, ici c'est la mienne)

Choisir Tools puis phpMyAdmin :



The screenshot shows the MAMP control panel interface. At the top, there's a navigation bar with links for Start, My Website, phpInfo, Tools (which is currently selected), FAQ, and MAMP Website. A prominent orange button on the right says "Buy MAMP PRO". Below the navigation, there's a large green banner with a white elephant logo on the left and a large number "4" on the right, indicating "available: MAMP PRO 4". In the center, there's a "phpMyAdmin" link with a dropdown menu showing "phpInfo" and "phpLiteAdmin" options, along with status information for APC, eAccelerator, XCache, and OPCache. The main content area is divided into several sections: "PHP" (with a "phpinfo" link), "MySQL" (with a note that MySQL can be administered via phpMyAdmin and connection parameters for Host: localhost, Port: 3307, User: root, Password: root), "MAMP Version" (showing 4.2.0 as up-to-date), and "News" (listing changes for Version 4.1.1, including German localization, TouchBar support, database deletion fix, and PHP 7.1 quirks).

On crée la table “users” et on lui donne 5 champs :

The screenshot shows the phpMyAdmin interface for creating a new table named 'users'. The 'Structure' tab is selected. The table has 5 columns, all defined as INT type with 'None' as the default value. The 'Collation' is set to 'None' and the 'Storage Engine' is 'InnoDB'. A 'PARTITION definition' section is present but empty. The 'Attributes' column for the first column contains a checked checkbox under 'Null' and a dropdown for 'Index'. The 'A_I' column for the first column also has a checked checkbox. At the bottom right, there are 'Preview SQL' and 'Save' buttons.

- On crée 5 champs et non pas 4 car une va servir à créer la clé primaire (primary key) appelée ID.
- L'ID permet à SQL de retrouver plus facilement une entrée du tableau
- Elle a toujours une valeur unique
- On indique dans le champs en question (ici user_id) que l'id s'auto-incrémente :

This screenshot shows the 'Structure' tab of the 'users' table in phpMyAdmin. The 'user_id' column is selected. An 'Add index' dialog is open, with 'Index name:' set to 'PRIMARY' and 'Index choice:' set to 'PRIMARY'. In the 'Attributes' column for the 'user_id' column, the 'Null' checkbox is unchecked and the 'Index' checkbox is checked. A pink circle highlights the 'Index' checkbox. The 'A_I' column for the 'user_id' column also has a checked checkbox. The 'Column' dropdown in the 'Advanced Options' section is set to 'user_id [int]'. At the bottom right, there are 'Go' and 'Cancel' buttons.

→ Cela a pour action que dès qu'une nouvelle entrée est créée, SQL générera automatiquement une valeur de cette dernière

Table créé :

The screenshot shows the 'Structure' tab of the MySQL Workbench interface. The table name is 'users'. The columns are defined as follows:

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A.I.
user_id	INT		None			□	PRIMARY	<input checked="" type="checkbox"/>
full_name	VARCHAR	128	None			□	---	□
email	VARCHAR	256	None			□	---	□
password	VARCHAR	128	None			□	---	□
age	INT		None			□	---	□

Table comments: Collation: Storage Engine: InnoDB

PARTITION definition: Partition by: Expression or column list

Partitions:

Buttons: Preview SQL, Save

On va ensuite la preview :

The 'Preview SQL' dialog box displays the following SQL code:

```
CREATE TABLE `partage_de_recettes`.`users` ( `user_id` INT NULL ,
`full_name` VARCHAR(128) NOT NULL , `email` VARCHAR(256) NOT NULL ,
`password` VARCHAR(128) NOT NULL , `age` INT NOT NULL , PRIMARY KEY
(`user_id`) ) ENGINE = InnoDB;
```

Buttons: Close

Après avoir généré la table on voit qu'elle existe :

The screenshot shows the 'Structure' tab of the MySQL Workbench interface. The table 'users' is selected. The columns and their properties are listed in the main table, and an index is shown in the 'Indexes' section.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	user_id	int(11)		No	None			AUTO_INCREMENT	<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>
2	full_name	varchar(128)	utf8_general_ci	No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>	
3	email	varchar(256)	utf8_general_ci	No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>	
4	password	varchar(128)	utf8_general_ci	No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>	
5	age	int(11)		No	None			<input type="button" value="Change"/> <input type="button" value="Drop"/> <input type="button" value="More"/>	

Buttons: Check all, With selected: Browse, Change, Drop, Primary, Unique, Index, Fulltext

Buttons: Print, Propose table structure, Move columns, Normalize

Buttons: Add 1 column(s) after age, Go

Indexes:

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
<input type="button" value="Edit"/> <input type="button" value="Drop"/>	PRIMARY	BTREE	Yes	No	user_id	0	A	No	

Create an index on 1 columns Go

Pour ajouter une entrée on clique sur "insert" :



On rempli les informations :

Column	Type	Function	Null	Value
user_id	int(11)			
full_name	varchar(128)			Mickaël Andrieu
email	varchar(256)			mickaël.andrieu@exemple.com
password	varchar(128)			devine
age	int(11)			34

Go

On retrouve les informations rentrées :

```
✓ 1 row inserted.  
Inserted row id: 1  
INSERT INTO `users` (`user_id`, `full_name`, `email`, `password`, `age`) VALUES (NULL, 'Mickaël Andrieu', 'mickaël.andrieu@exemple.com', 'devine', '34');
```

On retrouve bien les informations dans la table users :

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges

✓ Showing rows 0 - 0 (1 total, Query took 0.0012 seconds.)

```
SELECT * FROM `users`
```

Show all | Number of rows: 25 | Filter rows: Search this table

	+ Options	user_id	full_name	email	password	age
<input type="checkbox"/>	<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	Mickaël Andrieu	mickaël.andrieu@exemple.com	devine	34

Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Les types de champs MySQL :

→ MySQL propose une grande quantité de types de données

En pratique, on utilisera que 4 d'entre elles :

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

Les clés primaires :

→ Toute table doit posséder un champ qui joue le rôle de clé primaire
→ En général on utilise |

Prenez le réflexe de créer à chaque fois ce champ « id » en lui donnant l'index **PRIMARY**, ce qui aura pour effet d'en faire une clé primaire.

Vous en profiterez en général pour cocher la case **AUTO_INCREMENT** pour que ce champ gère lui-même les nouvelles valeurs automatiquement (1, 2, 3, 4...).

e champs “id” pour remplir le rôle de clé primaire

Modifiez une table :

Pour changer une entrée, il faut aller dans “structure” puis cliquer sur “add” :

The screenshot shows the MySQL Workbench interface for the 'users' table. The 'Structure' tab is active, displaying the columns: user_id (int(11)), full_name (varchar(128)), email (varchar(256)), password (varchar(128)), and age (int(11)). The 'user_id' column is highlighted as the primary key. In the bottom right, a modal window titled 'Add' is open, showing '1' column(s) after 'age'. The 'Indexes' tab at the bottom shows a primary index named 'PRIMARY' on the 'user_id' column.

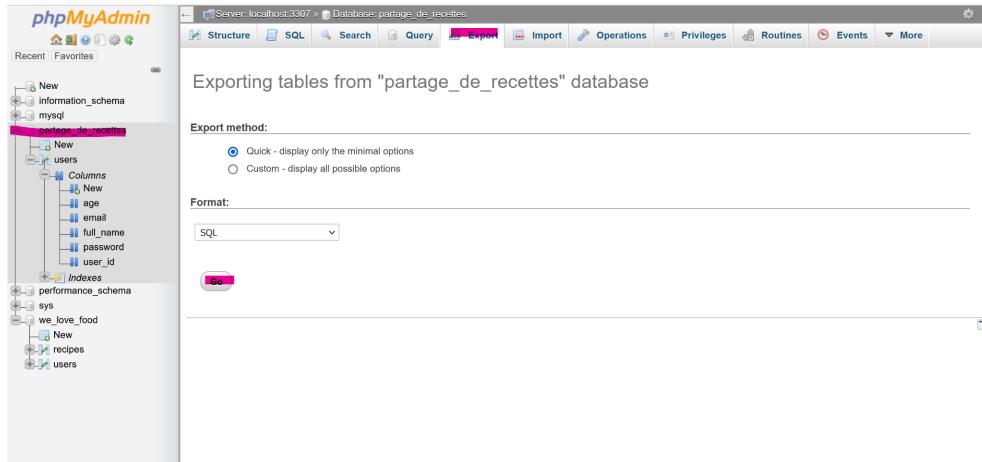
→ Si on a supprimé une entrée, on peut la rentrer à nouveau, néanmoins il faudra rajouter à la main les données (ici on a supprimé la ligne “âge”)

→ Vidéo explicative “[Modifiez une table](#)”

Importez et exportez des données :

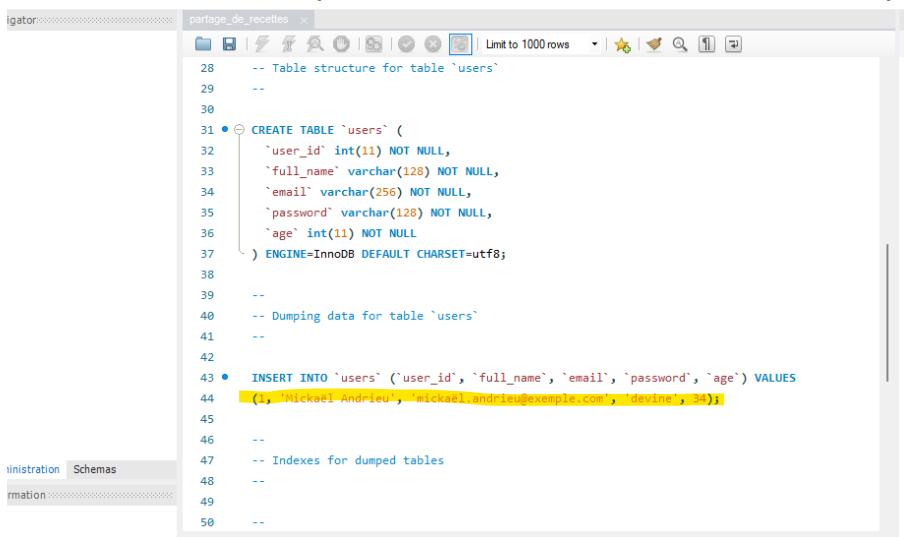
→ Vidéo explicative "[Importez et exportez des données](#)"

Pour exporter la table créée :



The screenshot shows the phpMyAdmin interface with the 'Structure' tab selected. The left sidebar lists databases: 'New', 'information_schema', 'mysql', and 'partage_de_recettes'. Under 'partage_de_recettes', there are tables: 'New' and 'users'. The 'users' table has columns: 'user_id' (PK), 'full_name', 'email', 'password', and 'age'. The 'Indexes' section is also visible. The top navigation bar includes tabs for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, and More. The 'Export' tab is currently active.

Si l'on ouvre le doc exporté, on retrouve les données rentrées précédemment :



```
28 -- Table structure for table `users`
29 --
30
31 • CREATE TABLE `users` (
32     `user_id` int(11) NOT NULL,
33     `full_name` varchar(128) NOT NULL,
34     `email` varchar(256) NOT NULL,
35     `password` varchar(128) NOT NULL,
36     `age` int(11) NOT NULL
37 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
38
39 --
40 -- Dumping data for table `users`
41 --
42
43 • INSERT INTO `users` (`user_id`, `full_name`, `email`, `password`, `age`)
44     VALUES
45     (1, 'Mickael Andrieu', 'mickael_andrieu@example.com', 'devine1', 30);
46
47 --
48 -- Indexes for dumped tables
49 --
50 --
```

→ Le fichier quand il est importé ou exporté possède l'extension ".sql"

A quoi sert ce fichier ?

1. **Transmettre votre base de données sur Internet** : pour le moment, votre base de données se trouve sur votre disque dur. Mais lorsque vous voudrez héberger votre site sur Internet, il faudra utiliser la base de données en ligne de votre hébergeur ! Le fichier **.sql** que vous allez générer vous permettra de **reconstruire** la base de données à l'identique.
2. **Faire une copie de sauvegarde de la base de données** : on ne sait jamais, si vous faites une bêtise ou si quelqu'un réussit à détruire toutes les informations sur votre site, vous serez bien content d'avoir une copie de secours sur votre disque dur !

Ce PC > Local Disk (C:) > MAMP > db > mysql > partage_de_recettes				
	Nom	Modifié le	Type	Taille
nts	db.opt	10/02/2022 12:02	Fichier OPT	1 Ko
	partage_de_recettes.sql	10/02/2022 13:20	SQL Text File	2 Ko
	users.frm	10/02/2022 12:59	Fichier FRM	9 Ko
	users.ibd	10/02/2022 12:59	Fichier IBD	96 Ko

Accédez aux données en PHP avec PDO

Connectez-vous à la base de données en PHP :

Pour se connecter à une base de données MySQL, vous allez devoir utiliser une extension PHP appelée **PDO** ("PHP Data Objects"). Cette extension est fournie avec PHP (en français, "les fonctions PDO sont à votre disposition"), mais parfois il vous faudra activer l'extension.

→ Chemin où se trouve le fichier de configuration sur mon ordinateur :

Loaded Configuration File	C:\MAMP\conf\php7.4.1\php.ini
---------------------------	-------------------------------

PDO support doit être marqué en "enabled" :

PDO

PDO support	enabled
PDO drivers	sqlite, mysql

Connectez PHP à MySQL avec PDO :

On va connecter MySQL, on va avoir besoin de 4 renseignements :

Maintenant que nous sommes certains que PDO est activé, nous pouvons nous connecter à MySQL. Nous allons avoir besoin de quatre renseignements :

- **Le nom de l'hôte** : c'est l'adresse IP de l'ordinateur où MySQL est installé. Le plus souvent, MySQL est installé sur le même ordinateur que PHP : dans ce cas, mettez la valeur `localhost`.
- **La base** : c'est le nom de la base de données à laquelle vous voulez vous connecter. Dans notre cas, la base s'appelle `my_recipes`. Vous l'avez créée avec phpMyAdmin dans le chapitre précédent.
- **L'identifiant et le mot de passe** : ils permettent de vous identifier. Renseignez-vous auprès de votre hébergeur pour les connaître.

Instruction PDO pour se connecter à la base "my recipes" :

```
1 <?php
2 // Souvent on identifie cet objet par la variable $conn ou $db
3 $mysqlConnection = new PDO(
4     'mysql:host=localhost;dbname=my_recipes;charset=utf8',
5     'root',
6     'root'
7 );
8 ?>
```

→ Cette ligne de code va créer une connexion à la base de données

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.

Testez la présence d'erreurs :

→ En cas d'erreur, PDO renvoie ce qu'on appelle un exception, qui permet de "capturer" l'erreur

Exemple :

```
1 <?php
2 try
3 {
4     $db = new PDO('mysql:host=localhost;dbname=my_recipes;charset=utf8', 'root', 'root');
5 }
6 catch (Exception $e)
7 {
8     die('Erreur : ' . $e->getMessage());
9 }
10 ?>
```

→ PHP essaie d'exécuter les instructions à l'intérieur du bloc "try" :

- S'il y a une erreur, il rentre dans le bloc "catch" et fait ce qu'on lui demande (ici, on arrête l'exécution de la page en affichant un message décrivant l'erreur)
- Si au contraire tout se passe bien, PHP poursuit l'exécution du code et ne lit pas ce qu'il y a dans le bloc "catch" (la page PHP ne doit pas afficher quoi que ce soit)

Effectuez une première requête SQL : effectuez une première requête SQL

→ La première requête SQL que l'on utilise :

```
1 SELECT * FROM recipes
```

Explication de cette requête :

- **SELECT** : en langage SQL, le premier mot indique quel type d'opération doit effectuer MySQL. Dans ce chapitre, nous ne verrons que `SELECT`. Ce mot-clé demande à MySQL d'afficher ce que contient une table.
- ***** : après le `SELECT`, on doit indiquer quels champs MySQL doit récupérer dans la table. Si on n'est intéressé que par les champs « nom » et « possesseur », il faudra taper :
`SELECT nom, possesseur FROM recipes`
Si vous voulez prendre tous les champs, tapez `*`. Cette petite étoile peut se traduire par « tout » : « Prendre tout ce qu'il y a... ».
- **FROM** : c'est un mot de liaison qui se traduit par « dans ». `FROM` fait la liaison entre le nom des champs et le nom de la table.
- **recipes** : c'est le nom de la table dans laquelle il faut aller piocher.

La requête :

```
1 <?php
2 $recipesStatement = $db->prepare('SELECT * FROM recipes');
3 ?>
```

Affichez le résultat d'une requête SQL :

Le problème, c'est que `$recipesStatement` contient quelque chose d'inexploitable directement : un objet [PDOStatement](#). Cet objet va contenir la requête SQL que nous devons exécuter, et par la suite, les informations récupérées en base de données.

Pour récupérer les données, demandez à cet objet d'exécuter la requête SQL et de récupérer toutes les données dans un format "exploitable" pour vous, c'est-à-dire sous forme d'un tableau PHP.

Cela s'écrit ainsi :

```
1 <?php
2 $recipesStatement->execute();
3 $recipes = $recipesStatement->fetchAll();
4 ?>
```

Résultat cela ressemble à :

```
1 <?php
2 try
3 {
4     // On se connecte à MySQL
5     $mysqlClient = new PDO('mysql:host=localhost;dbname=my_recipes;charset=utf8', 'root', 'root');
6 }
7 catch(Exception $e)
8 {
9     // En cas d'erreur, on affiche un message et on arrête tout
10    die('Erreur : '.$e->getMessage());
11 }
12
13 // Si tout va bien, on peut continuer
14
15 // On récupère tout le contenu de la table recipes
16 $sqlQuery = 'SELECT * FROM recipes';
17 $recipesStatement = $mysqlClient->prepare($sqlQuery);
18 $recipesStatement->execute();
19 $recipes = $recipesStatement->fetchAll();
20
21 // On affiche chaque recette une à une
22 foreach ($recipes as $recipe) {
23 ?>
24     <p><?php echo $recipe['author']; ?></p>
25 ?>
26 }
27 ?>
```

Filtrez les données :

→ On souhaite lister les recettes qui sont activées soit celles dont le champ “**is_enabled**” vaut “true”

→ 3 mots clés à retenir :

- WHERE
- ORDER BY
- LIMIT

Grâce au mot-clé **WHERE** , vous allez pouvoir trier vos données.

Puisque l'on souhaite récupérer uniquement les recettes avec le champ **is_enabled** à **TRUE** , alors la requête au début sera la même qu'avant, mais vous rajouterez à la fin **WHERE is_enabled = TRUE** .

Cela vous donne la requête :

```
1 <?php
2 $sqlQuery = 'SELECT * FROM recipes WHERE is_enabled = TRUE';
```

→ Travail avec vidéo, disponible [ici](#)

Pour trouver des entrées dans la base de données, on commence à travailler avec la requête “WHERE” :

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the path is "Server: localhost:3307 > Database: we_love_food_work > Table: recipes". The "SQL" tab is selected. A pink circle highlights the SQL query in the editor:

```
SELECT * FROM `recipes` WHERE author = 'mickael.andrieu@example.com'
```

Below the editor, there are several buttons: SELECT*, SELECT, INSERT, UPDATE, DELETE, Clear, Format, Get auto-saved query, Bind parameters, Delimiter, Show this query here again, Retain query box, Rollback when finished, Enable foreign key checks, and a Go button.

On obtient ce résultat en filtrage :

The screenshot shows the results of the query in the MySQL Workbench interface. The results are displayed in a table:

recipe_id	title	recipe	author	is_enabled
1	Cassoulet	Le cassoulet est une spécialité régionale du Langu...	mickael.andrieu@example.com	1
2	Couscous	Le couscous est d'une part une semoule de blé dur ...	mickael.andrieu@example.com	0

Below the table, there are buttons for Show all, Number of rows: 25, Filter rows: Search this table, Sort by key: None, Options, Edit, Copy, Delete, and a toolbar with Print, Copy to clipboard, Export, Display chart, and Create view.

Autre exemple :

The screenshot shows the results of another query in MySQL Workbench. The results are displayed in a table:

recipe_id	title	recipe	author	is_enabled
1	Cassoulet	Le cassoulet est une spécialité régionale du Langu...	mickael.andrieu@example.com	1
3	Escalope milanaise	L'escalope à la milanaise, ou escalope milanaise e...	mathieu.nebra@example.com	1

Below the table, there are buttons for Show all, Number of rows: 25, Filter rows: Search this table, Sort by key: None, Options, Edit, Copy, Delete, and a toolbar with Print, Copy to clipboard, Export, Display chart, and Create view.

Autre exemple, avec conditions WHERE & AND is_enabled :

The screenshot shows the phpMyAdmin interface for a database named 'we_love_food_work'. The current table is 'recipes'. A SQL query is run:

```
1 SELECT * FROM `recipes`
2 WHERE author = 'mickael.andrieu@example.com' AND is_enabled = 1
```

The results show one row:

recipe_id	title	recipe	author	is_enabled
1	Cassoulet	Le cassoulet est une spécialité régionale du Langu...	mickael.andrieu@example.com	1

Exemple avec LIMIT (ici avec 2) :

The screenshot shows the phpMyAdmin interface for a database named 'we_love_food_work'. The current table is 'recipes'. A SQL query is run with a LIMIT clause:

```
1 SELECT * FROM `recipes` LIMIT 2
```

The results show two rows:

recipe_id	title	recipe	author	is_enabled
1	Cassoulet	Le cassoulet est une spécialité régionale du Langu...	mickael.andrieu@example.com	1
2	Couscous	Le couscous est d'une part une semoule de blé dur ...	mickael.andrieu@example.com	0

Exemple avec OFFSET (à partir du combientième élément on reçoit les recettes) :

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** The query entered is `SELECT * FROM `recipes` LIMIT 2 OFFSET 1;`.
- Results:** The results show two rows from the `recipes` table:

recipe_id	title	recipe	author	is_enabled
2	Couscous	Le couscous est d'une part une semoule de blé dur ...	mickael.andrieu@example.com	0
4	Salade Romaine	La salade César est une recette de cuisine de sala...	laurene.castor@example.com	0
- Information Bar:** Shows "Showing rows 1 - 2 (2 total, Query took 0.0004 seconds.)".
- Table View:** A preview of the table structure is shown below the results.

Exemple avec ORDER BY :

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** The query entered is `SELECT * FROM `recipes` ORDER BY title;`.
- Results:** The results show four rows from the `recipes` table:

recipe_id	title	recipe	author	is_enabled
1	Cassoulet	Le cassoulet est une spécialité régionale du Langu... mickael.andrieu@example.com 1		
2	Couscous	Le couscous est d'une part une semoule de blé dur ... mickael.andrieu@example.com 0		
3	Cassolet	Le cassolet est une spécialité régionale du Langu... mickael.andrieu@example.com 1		
4	Romanaine	La salade César est une recette de cuisine de sala... laurene.castor@example.com 0		
- Information Bar:** Shows "Showing rows 0 - 3 (4 total, Query took 0.0025 seconds.) [title: CASSOULET... - SALADE ROMAINE...]".
- Table View:** A preview of the table structure is shown below the results.

Construisez des requêtes en fonction de variables : identifiez vos variables à l'aide des marqueurs

→ Les marqueurs sont des identifiants reconnus par PDO pour être remplacés lors de la préparation de la requête par les variables PHP

→ Vidéo explicative [ici](#)

On ne concatène **JAMAIS** une requête SQL pour passer des variables, au risque de créer des injections SQL ! Le sujet des injections SQL est un peu trop complexe pour être détaillé ici. Si vous souhaitez en apprendre plus à ce sujet, je vous invite à consulter [Wikipedia](#).

Traquez les erreurs :

→ Pour trouver les erreurs, il faut activer “\$db” :

```
1 <?php
2 $db = new PDO(
3     'mysql:host=localhost;dbname=my_recipes;charset=utf8',
4     'root',
5     'root',
6     [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION],
7 );
8 ?>
```

Lorsque vous avez un problème avec une requête et que vous voulez demander de l'aide, pensez **toujours** à activer les erreurs lors de la connexion à la base de données comme je viens de vous montrer, cela vous permettra d'avoir un message d'erreur détaillé.

N'oubliez pas que personne ne peut vous aider si vous donnez juste le message par défaut `Call to a member function fetchAll() on a non-object` !

Ajoutez, modifiez et supprimez des recettes

→ Requêtes SQL fondamentales :

- INSERT
- UPDATE
- DELETE

Implémentez l'ajout des recettes :

→ Ajoutez une recette avec l'instruction “INSERT INTO”

Exemple code :

```
1 <?php
2 $sqlQuery = 'INSERT INTO recipes(title, recipe, author, is_enabled) VALUES (:title, :recipe, :author,
:is_enabled)';
3
```

→ “INSERT INTO” indique qu'on va insérer une entrée

→ On précise ici le nom de la table (ici c'est “recipes”), puis on liste entre parenthèses les noms et champs dans lesquels on souhaite placer des informations
→ Ensuite on inscrit “VALUES” suivi des valeurs à insérer dans le même ordre que les champs qu'on a indiqué

Exemple requête SQL au sein d'un script PHP :

```
1 <?php
2 try
3 {
4     $db = new PDO('mysql:host=localhost;dbname=my_recipes;charset=utf8', 'root', 'root');
5 }
6 catch (Exception $e)
7 {
8     die('Erreur : ' . $e->getMessage());
9 }
10
11 // Ecriture de la requête
12 $sqlQuery = 'INSERT INTO recipes(title, recipe, author, is_enabled) VALUES (:title, :recipe, :author,
:is_enabled)';
13
14 // Préparation
15 $insertRecipe = $mysqlClient->prepare($sqlQuery);
16
17 // Exécution ! La recette est maintenant en base de données
18 $insertRecipe->execute([
19     'title' => 'Cassoulet',
20     'recipe' => 'Etape 1 : Des flageolets ! Etape 2 : Euh ...',
21     'author' => 'contributeur@example.com',
22     'is_enabled' => 1, // 1 = true, 0 = false
23 ]);
```

→ On n'insère pas de valeur pour le champ “recipe_id”, ce champs a la propriété “auto-increment”

Si vous voulez passer les valeurs PHP `true` ou `false`, alors il faudra déclarer le champ `is_enabled` avec le type MySQL `BOOL`.

→ On récupère des variables de “`$_post`” (issues d'un formulaire) pour insérer une entrée dans la base de données

→ Vidéo explicative [ici](#)

Editez une recette avec l'instruction “UPDATE” :

→ Mots-clés :

- `UPDATE`
- `SET`

Si formulaire d'édition et autorisation pour les utilisateurs à éditer les champs “title” & “recipe”, la requête SQL correspondante est celle-ci :

```
mysql
1 UPDATE recipes SET title = :title, recipe = :recipe WHERE recipe_id = :id
```

Comment cela marche :

- Tout d'abord, le mot-clé `UPDATE` permet de dire qu'on va modifier une entrée.
- Ensuite, le nom de la table (`recipes`).
- Le mot-clé `SET` sépare le nom de la table de la liste des champs à modifier.
- Viennent ensuite les champs qu'il faut modifier, séparés par des virgules. Ici, on modifie le champ `title`, puis on fait de même pour le champ `recipe`. Les autres champs ne seront pas modifiés.
- Enfin, le mot-clé `WHERE` est tout simplement indispensable. Il nous permet de dire à MySQL quelle entrée il doit modifier (sinon, toutes les entrées seraient affectées !). On se base très souvent sur le champ `recipe_id` pour indiquer **quelle entrée** doit être modifiée.

Pour désactiver les recettes d'un utilisateur on utilisera cette requête :

```
mysql
1 UPDATE recipes SET is_enabled = 0 WHERE author = 'mickael.andrieu@example.com'
```

→ Vidéo explicative [ici](#)

Supprimez des données avec “DELETE” :

Rapide et simple à utiliser, elle est quand même un poil dangereuse : après suppression, il n'y a aucun moyen de récupérer les données, alors faites bien attention !

Exemple requête pour supprimer une recette à partir de son identifiant :

```
1 DELETE FROM recipes WHERE recipe_id=:id
```

mySQL

Explications :

- `DELETE FROM` : pour dire « supprimer dans » ;
- `recipes` : le nom de la table ;
- `WHERE` : indispensable pour indiquer quelle(s) entrée(s) doi(ven)t être supprimée(s).

→ Vidéo explicative [ici](#)

Traitez les erreurs SQL :

→ Il peut y avoir des erreurs en SQL

→ Par exemple comme type d'erreur : requête mal écrite, table souhaitée n'existe pas, etc

→ Pour trouver une erreur, il faut repérer la requête qui plante et forcez l'affichage de l'erreur :

```
1 <?php
2 $deleteRecipeStatement = $db->prepare('DELETE FROM recipes WHERE recipe_id = :id');
3 $deleteRecipeStatement->execute([
4     'id' => $id,
5 ]) or die(print_r($db->errorInfo()));
```

Ajoutez des commentaires grâce aux jointures SQL

→ Dans la pratique on travaille avec plusieurs tables dans une base, la plupart sont interconnectées.

Cela permet de :

- Mieux découper les informations
- Eviter les répétitions
- Rendre les données plus faciles à gérer

→ Ajouter une gestion des commentaires sur la page d'une recette, il faut lier le commentaire à un utilisateur et à une recette

→ Notion de jointure

Modéliser une relation :

Si on considère une page qui affiche une recette avec possibilité pour les utilisateurs de commenter ou évaluer la recette, un commentaire a les propriétés suivantes :

- Un identifiant unique
- Une recette
- Un auteur
- Une date de publication
- Une note

Pour cela, le mieux est de créer un champs `user_id` dans la table “comments” qui fait référence au champs `user_id` dans la table “users” :

id	recipe	user_id	created_at	ranking	comment
1	Cassoulet	1	03-08-2021 18:00:00	2	Bof 😐
2	Cassoulet	2	01-08-2021 12:00:00	4	Super recette 😍
3	Couscous	3	30-08-2021 12:00:00	0	Pas bon du tout ! 🙄
4	Couscous	1

Comprenez le principe de jointure :

Les informations sont séparées dans des tables différentes, et c'est bien. Cela évite de dupliquer des informations sur le disque.

Il existe plusieurs types de jointures qui permettent de choisir exactement les données que l'on veut récupérer.

En voici 2 importantes :

- 1) **Les jointures internes** : elles ne sélectionnent que les données qui ont une correspondance entre les deux tables
- 2) **Les jointures externes** : elles sélectionnent toutes les données, même si certaines n'ont pas de correspondance dans l'autre table

Exemple :

Pour cela, imaginons que nous ayons une 4e personne dans la table des utilisateurs, un certain Manuels Vache, qui n'a jamais publié de commentaires :

user_id	full_name	email	password	age
1	Mickaël Andrieu	mickael.andrieu@exemple.com	...	34
2	Laurène Castor	laurene.castor@exemple.com	...	28
3	Mathieu Nebra
4	Manuels Vache	58

Manuels Vache est référencé dans la table `users` mais il n'apparaît nulle part dans la table `comments` car il n'a jamais commenté.

→ Si on récupère les données des deux tables à l'aide d'une jointure interne, Manuels n'apparaîtra pas dans les résultats de la requête. La jointure interne force les données d'une table à avoir une correspondance dans l'autre

→ Si on récupère les données des deux tables à l'aide d'une jointure externe, on aura toutes les données de la table des utilisateurs, même s'il n'y a pas de correspondance dans l'autre table des commentaires; donc Manuels, qui n'a jamais commenté, apparaîtra

La jointure externe est donc plus complète car elle est capable de récupérer plus d'informations, tandis que la jointure interne est plus stricte car elle ne récupère que les données qui ont une équivalence dans l'autre table.

Exemple données récupérées avec une jointure interne :

full_name	comment
Mickaël Andrieu	Bof 😐
Laurène Castor	Super recette 😋
Mathieu Nebra	Pas bon du tout ! 🥺
...	...

Exemple avec jointure externe :

full_name	comment
Mickaël Andrieu	Bof 😐
Laurène Castor	Super recette 😋
Mathieu Nebra	Pas bon du tout ! 🙄
Manuels Vache	NULL
...	...

→ Manuels apparaît désormais mais comme il n'a jamais commenté la valeur associée est "NULL"

Effectuez des jointures internes :

→ Une jointure interne peut être effectuée à l'aide du mot-clé "JOIN"

```
1 # Liste des noms et commentaires associés
2 SELECT u.full_name, c.comment
3 FROM users u
4 INNER JOIN comments c
5 ON u.user_id = c.user_id
```

→ On récupère les données depuis une table principale (ici "users")

→ On fait une jointure interne ("INNER JOIN") avec une autre table (ici "comments")

→ La liaison entre les champs est faite dans la clause "ON"

Si vous voulez :

- filtrer (`WHERE`),
- ordonner (`ORDER BY`)
- ou limiter les résultats (`LIMIT`),

... vous devez le faire à la fin de la requête, après le « `ON u.user_id = c.user_id` ».

Exemple :

```
1 # Liste des noms et commentaires associés
2 # à la recette Cassoulet
3 # du plus vieux au plus récent
4 # limité à 10
5 SELECT u.full_name, c.comment
6 FROM users u
7 INNER JOIN comments c
8 ON u.user_id = c.user_id
9 WHERE u.recipe = "Cassoulet"
10 ORDER BY c.created_at DESC
11 LIMIT 10
```

Effectuez des jointures externes :

Les jointures externes permettent de récupérer toutes les données, même celles qui n'ont pas de correspondance.

→ La seule syntaxe disponible est à la base de “JOIN”

Deux écritures à connaître :

- LEFT JOIN
- RIGHT JOIN

Récupérez toute la table gauche avec “LEFT JOIN” :

Exemple :

```
1 # Liste des noms et commentaires associés
2 SELECT u.full_name, c.comment
3 FROM users u
4 LEFT JOIN comments c
5 ON u.user_id = c.user_id
```

`users` est appelée la « table de gauche » et `comments` la « table de droite ».

Le `LEFT JOIN` demande à récupérer tout le contenu de la table de gauche, donc tous les utilisateurs, même si ces derniers n'ont pas d'équivalence dans la table `comments` :

full_name	comment
Mickaël Andrieu	Bof 😕
Laurène Castor	Super recette 😋
Mathieu Nebra	Pas bon du tout ! 🙄
Manuels Vache	NULL
...	...

Manuels apparaît désormais dans les résultats de la requête grâce à la jointure externe. Comme il n'a jamais commenté, la colonne du commentaire est vide.

Récupérez toute la table gauche avec “RIGHT JOIN” :

→ Le “RIGHT JOIN” demande à récupérer toutes les données de la table dite “de droite”, même si celle-ci n'a pas d'équivalent dans l'autre table

Exemple :

```
1 # Liste des noms et commentaires associés
2 SELECT u.full_name, c.comment
3 FROM users u
4 RIGHT JOIN comments c
5 ON u.user_id = c.user_id
```

→ La table de droite est “comments”, on récupérerait donc tous les commentaires, même ceux qui n'ont pas d'auteur associé

Allez plus loin avec les fonctions SQL

→ Fonctionnalité utile dans le langage SQL : les fonctions

En effet, le langage SQL permet d'effectuer des calculs directement sur ses données, à l'aide de fonctions toutes prêtes.

Celles-ci sont moins nombreuses qu'en PHP, mais elles sont spécialement dédiées aux bases de données, et se révèlent très puissantes dans la pratique.

Ainsi, en effectuant directement certains calculs en SQL plutôt qu'en PHP, nous allégeons un petit peu la quantité de code à écrire.

Les fonctions SQL peuvent être classées en deux catégories :

- 1) **Les fonctions scalaires** : elles agissent sur chaque entrée. Exemple : on peut transformer en majuscules la valeur de chacune des entrées d'un champs
- 2) **Les fonctions d'agrégat** : lorsqu'on utilise ce type de fonctions, des calculs sont faits sur l'ensemble de la table pour retourner une valeur. Exemple : calculer la moyenne des prix retourne une valeur : le prix moyen

Utilisez une fonction scalaire SQL :

→ Fonction SQL de type scalaire, la fonction “DATE_FORMAT”

On écrit les noms des fonctions SQL en majuscules, comme on le fait déjà pour la plupart des mots-clés comme `SELECT`, `INSERT`, etc.

Ce n'est pas une obligation mais plutôt une convention, une habitude qu'ont prise les programmeurs.

Exemple de requête contenant fonction scalaire “DATE_FORMAT” :

```
1 SELECT *, DATE_FORMAT(c.created_at, "%d/%m/%Y") FROM recipes r LEFT JOIN comments c on r.recipe_id = c.recipe_id WHERE r.recipe_id = 1
```

→ La fonction “DATE_FORMAT” modifie uniquement la valeur envoyée à PHP, on ne touche pas au contenu de la table

→ Cela crée un “champ virtuel” qui n'existe que le temps de la requête

On utilise le mot-clé “AS” :

```
1 SELECT DATE_FORMAT(c.created_at, "%d/%m/%Y") AS comment_date FROM recipes r LEFT JOIN comments c on r.recipe_id = c.recipe_id WHERE r.recipe_id = 1
```

→ On récupère les dates de commentaires via un champ virtuel appelé “comment_date”

Ce champ virtuel est appelé "**alias**".

Voici le tableau que retournera MySQL après la requête précédente :

comment_date
16/07/2021
14/07/2021
12/07/2021

PHP ne récupère que le champ nommé `comment_date` (même s'il n'existe pas dans la table).

En affichant le contenu de ce champ, on ne récupère que les dates de publication des commentaires correctement formatées.

Bien entendu, vous pouvez aussi récupérer le contenu des autres champs comme avant, sans forcément leur appliquer une fonction :

```
mysql
1 SELECT *, DATE_FORMAT(c.created_at, "%d/%c/%Y") AS comment_date FROM recipes r LEFT JOIN comments c
on r.recipe_id = c.recipe_id WHERE r.recipe_id = 1
```

Utilisez une fonction d'agrégat :

Ces fonctions diffèrent assez des précédentes. Plutôt que de modifier des valeurs une à une, elles font des opérations sur plusieurs entrées pour retourner une seule valeur.

- Par exemple “ROUND()” est une fonction scalaire qui permet d’arrondir une valeur
- On récupère autant d’entrées qu’il y en avait dans la table
- La fonction “AVG()” renvoie une seule entrée : la valeur moyenne de toutes les lignes
- Elle calcule la moyenne d’un champ contenant des nombres

Exemple requête :

```
1 SELECT AVG(c.review) as rating FROM recipes r LEFT JOIN comments c on r.recipe_id = c.recipe_id WHERE r.recipe_id = 1
```

- On donne encore un alias au résultat donné par la fonction, la particularité est que cette requête ne va retourner qu’une seule entrée, à savoir l’évaluation moyenne de la recette :

rating
2.33333

- Comme en PHP, on peut appeler une fonction dans une fonction

Ne mélangez pas une fonction d'agrégat avec d'autres champs :

Vous **ne devez pas** récupérer d’autres champs de la table quand vous utilisez une fonction d’agrégat, contrairement à tout à l’heure avec les fonctions scalaires.

Regroupez des données avec “GROUP BY” et “HAVING” :

Pour faire cela, on doit utiliser un nouveau mot-clé : **GROUP BY** qui signifie « grouper par ».

On utilise cette clause en combinaison d'une fonction d'agrégat (comme **AVG**) pour obtenir des informations intéressantes sur des groupes de données.

Voici un exemple d'utilisation de **GROUP BY** :

```
1 SELECT AVG(review) AS rating, recipe_id FROM comments GROUP BY recipe_id
```

sql

Il faut utiliser **GROUP BY** en même temps qu'une fonction d'agrégat, sinon il ne sert à rien.

Ici, on récupère la note moyenne des commentaires, que l'on regroupe ensuite par recette. Par conséquent, on obtiendra la liste des recettes de la table et la note moyenne des commentaires !

rating	recipe_id
2.50	1
3.00	2
4.50	3

HAVING est un peu l'équivalent de **WHERE**, mais il agit sur les données une fois qu'elles ont été regroupées. C'est donc une façon de filtrer les données à la fin des opérations.

Voyez la requête suivante :

```
1 SELECT AVG(review) AS rating, recipe_id FROM comments GROUP BY recipe_id HAVING rating >= 3
```

sql

Avec cette requête, on récupère seulement les recettes dont la note moyenne est supérieure ou égale à 3.

HAVING ne doit s'utiliser que sur le résultat d'une fonction d'agrégat. Voilà pourquoi on l'utilise ici sur **rating** et non sur **recipe_id**.



Je ne comprends pas la différence entre **WHERE** et **HAVING**. Les deux permettent de filtrer, non ?

Oui, mais pas au même moment :

- **WHERE** agit en premier, avant le groupement des données ;
- **HAVING** agit en second, après le groupement des données.

On peut d'ailleurs très bien combiner les deux, regardez l'exemple suivant :

```
1 SELECT AVG(review) AS rating, recipe_id FROM comments WHERE user_id = 1 GROUP BY recipe_id HAVING rating >= 2
```

sql

Oui, mais pas au même moment :

- **WHERE** agit en premier, avant le groupement des données ;
- **HAVING** agit en second, après le groupement des données.

On peut d'ailleurs très bien combiner les deux, regardez l'exemple suivant :

```
sql
1 SELECT AVG(review) AS rating, recipe_id FROM comments WHERE user_id = 1 GROUP BY recipe_id HAVING
rating >= 2
```

Ça commence à faire de la requête costaude. 😊

Ici, on demande à récupérer la note moyenne par recette de Mickaël (**WHERE**), dont la valeur moyenne est supérieure ou égale à 2 (**HAVING**).

Autres cours à voir :

- [Adoptez une architecture MVC en PHP](#)
- [Programmez en orienté objet en PHP](#)
- [Construisez un site web à l'aide du framework Symfony 5](#)