**Mathieu Nebra**
Avec la participation de Mickaël Andrieu

CONCEVEZ VOTRE SITE WEB AVEC

# PHP et MYSQL

4e édition

Éditions EYROLLES

CONCEVEZ VOTRE SITE WEB AVEC

# PHP et MYSQL

4e édition

Vous connaissez le HTML et vous avez toujours rêvé de créer un site web dynamique, avec votre propre blog, vos forums et votre espace membres ? Ne cherchez plus ! Découvrez dans cet ouvrage dédié aux débutants comment utiliser les outils les plus célèbres du web dynamique : PHP et MySQL !

## QU'ALLEZ-VOUS APPRENDRE ?

**Les bases de PHP**

- Les variables et conditions
- Les boucles, tableaux et fonctions
- Au secours ! Mon script plante !
- Inclure des portions de page

**Transmettre des données de page en page**

- Transmettre des données avec l'URL et les formulaires
- Protéger une page par mot de passe
- Variables superglobales
- Sessions et cookies

**Stocker des informations dans une base de données**

- phpMyAdmin
- Lire et écrire des données
- Les fonctions SQL
- Les jointures entre tables

## À PROPOS DE L'AUTEUR

Co-fondateur d'OpenClassrooms, Mathieu Nebra se passionne depuis l'âge de 13 ans pour la création de cours en ligne. Son objectif : partager la connaissance d'une façon nouvelle, chaleureuse et enfin accessible à tous. Auteur de plusieurs best-sellers, il publie régulièrement des cours en ligne et expérimente de nouvelles approches pédagogiques avec la communauté de plus d'un million de membres qu'il a fédérée.

www.editions-eyrolles.com

# PHP et MYSQL

## DANS LA MÊME COLLECTION

M. Nebra. – **Réalisez votre site web avec HTML 5 et CSS 3.**
N° 0100975, 3e édition, 2023, 288 pages.

M. Nebra, M. Schaller. – **Programmez avec le langage C++.**
N°0100886, 2023, 674 pages.

V. Thuillier. – **Programmez en orienté objet en PHP.**
N°14472, 2e édition, 2017, 474 pages.

J. Pardanaud, S. de la Marck. – **Découvrez le langage JavaScript.**
N°14399, 2017, 478 pages.

A. Bacco. – **Développez votre site web avec le framework Symfony3.**
N°14403, 2016, 536 pages.

M. Chavelli. – **Découvrez le framework PHP Laravel.**
N°14398, 2016, 336 pages.

R. De Visscher. – **Découvrez le langage Swift.**
N°14397, 2016, 128 pages.

M. Lorant. – **Développez votre site web avec le framework Django.**
N°21626, 2015, 285 pages.

E. Lalitte. – **Apprenez le fonctionnement des réseaux TCP/IP.**
N°21623, 2015, 300 pages.

## SUR LE MÊME THÈME

C. Soutou. – **Programmer avec MySQL.**
N°00368, 6e édition, 2021, 576 pages.

R. Bruchez. – **Les bases de données NoSQL.**
N°67866, 3e édition, 2021, 304 pages.

C. Pierre de Geyer et G. Ponçon. –
**Mémento PHP 7 & SQL.** N°67885, 2019, 14 pages.

J. Engels. – **PHP 7. Cours et exercices.**
N°67360, 2017, 585 pages.

P. Martin, J. Pauli, C. Pierre de Geyer, É. Daspet. – **PHP 7 avancé.**
N°14357, 2016, 732 pages.

E. Biernat, M. Lutz. – **Data science : fondamentaux et études de cas.**
N°14243, 2015, 312 pages.

R. Rimelé. – **Mémento MySQL 5.**
N°14039, 4e édition, 2014, 14 pages.

Mathieu Nebra

CONCEVEZ VOTRE SITE WEB AVEC

# PHP et MYSQL

4e édition

Éditions EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com



ISBN : 978-2-416-00885-6

# Avant-propos

PHP permet de développer toutes sortes de sites web : blogs, réseaux sociaux, sites e-commerces… car ce langage a justement été conçu pour créer des sites « vivants ». On les appelle des **sites dynamiques**.

Cela vous semble compliqué ? Pas de panique : c'est un cours pour vrais débutants en PHP et vous pourrez avancer à votre rythme, étape par étape. À la fin, vous serez capable de créer votre propre site.

Pour vous accompagner dans votre apprentissage, nous avons intégré un projet fil rouge sur lequel vous allez progresser de chapitre en chapitre. Chaque chapitre vous permettra de compléter un morceau du TP. C'est pourquoi nous vous conseillons de les suivre dans l'ordre.

Au cours de ce TP, nous allons réaliser ensemble un site web dynamique qui permettra aux utilisateurs de contribuer en partageant des recettes de cuisine ; de quoi voir les bases et vous lancer dans l'aventure PHP en toute autonomie.

À la fin de ce cours, vous serez capable de :

- faire vos premiers pas en PHP ;
- réaliser un site web dynamique ;
- transmettre des données de page en page ;
- stocker des informations dans une base de données.

**Prérequis** : pour utiliser PHP, il faut connaître au préalable les langages HTML et CSS.

**Outils nécessaires** :

- un éditeur de code comme Visual Studio Code ;
- sous Windows ou macOS, il faut installer MAMP ;
- sous Linux, il faut installer XAMPP.

## Mettez-vous à niveau en HTML et CSS si besoin

Les sites web que vous visitez aujourd'hui, y compris OpenClassrooms, sont pour la plupart des sites dynamiques. Le seul prérequis pour apprendre à créer ce type de site est de savoir réaliser des sites statiques en HTML et CSS.

## Pratiquez en suivant le projet fil rouge

L'objectif de cet ouvrage est de vous permettre de réaliser des sites web dynamiques, pas à pas.

Pour cela, nous avons mis au point un projet fil rouge. Cela signifie que vous allez avancer chapitre par chapitre en apprenant comment **réaliser un site web dynamique de partage de recettes de cuisine**. Chaque chapitre vous donnera des clés supplémentaires pour avancer dans ce projet pratique.

## Structure de l'ouvrage

Le plan de ce livre a été conçu pour faciliter votre apprentissage du PHP et de MySQL. Quatre parties vous sont ainsi proposées pour apprendre les bases de PHP, écrire son premier script, transmettre des données de page en page, stocker des informations dans une base de données et pour enrichir vos (nouvelles) connaissances sur PHP et les bases de données SQL.

## Comment lire ce livre ?

### Suivez l'ordre des chapitres

Lisez ce livre comme on lit un roman. Il a été conçu pour cela. Contrairement à beaucoup de livres techniques où il est courant de lire en diagonale et de sauter certains chapitres, il est ici fortement recommandé de suivre l'ordre du livre, à moins que vous ne soyez déjà, au moins un peu, expérimenté.

### Pratiquez en même temps

Pratiquez régulièrement. N'attendez pas d'avoir fini de lire ce livre pour allumer votre ordinateur.

### Les compléments web

Pour télécharger le code source des exemples de cet ouvrage, veuillez-vous rendre à cette adresse : http://www.editions-eyrolles.com/dl/0100885.

# Table des matières

# Première partie

# Introduction

Avant d'entrer dans le vif du sujet, voici quelques informations clés à connaître pour apprendre à développer en douceur.

# 1 Fonctionnement d'un site écrit en PHP

Ce qui fait le succès du Web aujourd'hui, c'est à la fois sa simplicité et sa facilité d'accès. Un internaute lambda n'a pas besoin de savoir comment ça fonctionne concrètement et heureusement pour lui.

En revanche, un apprenti webmaster doit, avant toute chose, connaître les bases du fonctionnement d'un site web. Qu'est-ce qu'un serveur ? Un client ? Comment rend-on son site dynamique ? Que signifient PHP et MySQL ?

Ce premier chapitre est là pour répondre à toutes ces questions et vous montrer que vous êtes capable d'apprendre à créer des sites web dynamiques.

## Différences entre sites statiques et dynamiques

On considère qu'il existe deux types de sites web : les sites **statiques** et les sites **dynamiques**.

- **Les sites statiques** sont réalisés uniquement à l'aide des langages HTML et CSS. Ils fonctionnent très bien mais leur contenu n'est pas mis à jour automatiquement : il faut que le propriétaire du site (le webmaster) modifie le code source pour y ajouter des nouveautés. Ce n'est pas très pratique quand on doit mettre à jour son site plusieurs fois dans la même journée. Les sites statiques sont donc bien adaptés pour réaliser des sites « vitrines », pour présenter par exemple son entreprise, sans aller plus loin. Ce type de site se fait de plus en plus rare aujourd'hui car dès qu'on ajoute un élément d'interaction (comme un formulaire de contact), on ne parle plus de site statique mais de site dynamique.
- **Les sites dynamiques,** plus complexes, utilisent d'autres langages, en plus de HTML et CSS, tels que PHP et MySQL. Le contenu de ces sites web est dit « dynamique » parce qu'il peut changer sans l'intervention du webmaster !

Vous pouvez lire sur OpenClassrooms le cours HTML 5 et CSS pour vous mettre à niveau : *https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3*.


Logo PHP

## Comment fonctionne un site web ?

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

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

- **Les clients :** ce sont les ordinateurs des internautes comme vous. Votre ordinateur fait donc partie de la catégorie des clients. Chaque client représente un visiteur d'un site web.
- **Les serveurs :** ce sont des ordinateurs puissants qui stockent et délivrent des sites web aux internautes, c'est-à-dire aux clients. Les internautes n'ont pour la plupart jamais vu un serveur de leur vie. Pourtant, les serveurs sont indispensables au bon fonctionnement du Web.

La plupart du temps, le serveur est dépourvu d'écran : il reste allumé et travaille tout seul sans intervention humaine, 24h/24, 7j/7. Un vrai forçat du travail.

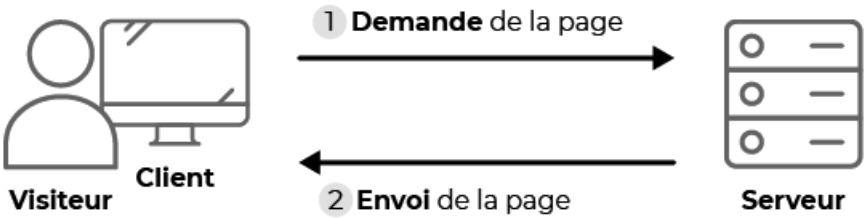
On résume : votre ordinateur est appelé le **client**, tandis que l'ordinateur qui détient le site web est appelé le **serveur**. Comment les deux communiquent-ils ?

C'est justement là que se situe la différence entre un site statique et un site dynamique. Voyons ensemble ce qui change.

## Cas d'un site statique

Lorsque le site est statique, le schéma est très simple. Cela se passe en deux temps.

1. Le client demande au serveur à voir une page web.
2. Le serveur lui répond en lui envoyant la page réclamée.




Transfert avec un site statique

La communication est donc plutôt basique :

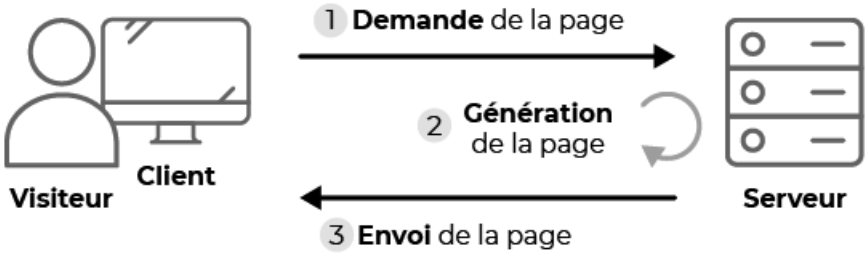
- « Bonjour, je suis le client, je voudrais voir cette page web » ;
- « Tiens, voilà la page que tu m'as demandée ».

Sur un site statique, il ne se passe rien d'autre. Le serveur stocke des pages web et les envoie aux clients qui les demandent sans les modifier.

## Cas d'un site dynamique

Lorsque le site est dynamique, il y a une étape intermédiaire : la page est **générée** :

- Le client demande au serveur à voir une page web.
- Le serveur prépare la page spécialement pour le client.
- Le serveur lui envoie la page qu'il vient de générer.




Transfert avec un site dynamique

La page web est générée à chaque fois qu'un client la réclame. C'est précisément ce qui rend les sites dynamiques « vivants » : le contenu d'une même page peut changer d'un instant à l'autre.

C'est comme cela que certains sites parviennent à afficher par exemple votre pseudonyme sur toutes les pages. Étant donné que le serveur génère une page à chaque fois qu'on lui en demande une, il peut la personnaliser en fonction des goûts et des préférences du visiteur.

## Les langages du Web

Lorsqu'on crée un site web, on est amené à manipuler non pas un mais plusieurs langages. En tant que webmaster, il faut impérativement les connaître.

### Pour un site statique : HTML et CSS

De nombreux langages ont été créés pour produire des sites web. Deux d'entre eux constituent une base incontournable pour tous les webmasters.

- **HTML :** c'est le langage à la base des sites web. Simple à apprendre, il fonctionne à partir de balises. Voici un exemple de code HTML :

```
<p>Bonjour, je suis un <em>paragraphe</em> de texte !</p>
```

- **CSS :** c'est le langage de mise en forme des sites web. Alors que le HTML permet d'écrire le contenu de vos pages web et de les structurer, le langage CSS s'occupe de la mise en forme et de la mise en page. C'est en CSS qu'on choisit notamment la couleur, la taille des menus et bien d'autres choses encore. Voici un code CSS :

```
div.banner {
    text-align: center;
    font-weight: bold;
    font-size: 120%;
}
```

Ces langages sont la base de tous les sites web. Lorsque le serveur envoie la page web au client, il envoie en fait du code en langage HTML et CSS.

Le problème, c'est que lorsqu'on connaît **seulement** HTML et CSS, on ne peut produire que des sites statiques... et non des sites dynamiques ! Pour ces derniers, il est nécessaire de manipuler d'autres langages en plus de HTML et CSS.

## Pour un site dynamique : ajoutez PHP et MySQL

Quel que soit le site web qu'on souhaite créer, HTML et CSS sont donc indispensables. Cependant, ils ne suffisent pas pour réaliser des sites dynamiques. Il faut les compléter avec d'autres langages.

C'est justement tout l'objet du présent ouvrage : vous allez apprendre à manipuler PHP et MySQL pour réaliser un site web dynamique.

- **PHP** est un langage que seuls les serveurs comprennent et qui permet de rendre votre site dynamique. C'est PHP qui « génère » la page web comme on l'a vu sur un des schémas précédents.

Ce sera le premier langage que nous découvrirons ici. Voici un code PHP :

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

Il peut fonctionner seul, mais il ne prend vraiment de l'intérêt que s'il est combiné à un outil tel que MySQL.

- **MySQL** est ce qu'on appelle un SGBD (système de gestion de base de données). Pour faire simple, son rôle est d'enregistrer des données de manière organisée afin de vous aider à les retrouver facilement. C'est grâce à MySQL que vous pourrez enregistrer la liste des membres de votre site, les messages postés sur le forum, etc. Le langage qui permet de communiquer avec la base de données s'appelle le SQL. Voici un code en langage SQL :

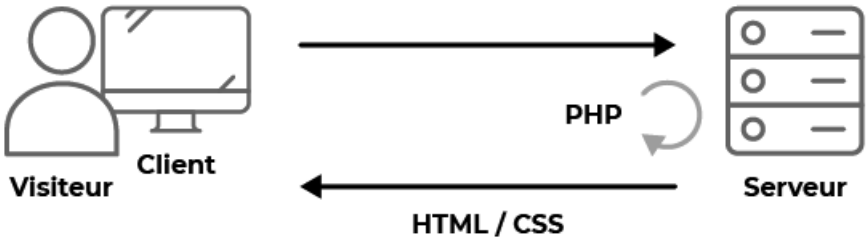
```
SELECT id, auteur, message, datemsg FROM livreor ORDER BY datemsg DESC
LIMIT 0, 10
```

PHP et MySQL sont disponibles gratuitement et sous licence Open Source. Cela signifie une chose essentielle : vous n'aurez pas à débourser un centime pour construire votre site web !

Oublions MySQL pour le moment et concentrons-nous sur PHP.

Les clients sont incapables de comprendre le code PHP : ils ne connaissent que le HTML et le CSS. Seul le serveur peut lire du PHP.

Le rôle de ce langage est justement de générer du code HTML, code qui est ensuite envoyé au client de la même manière qu'un site statique.




PHP décide de ce qui va s'afficher sur la page web envoyée au visiteur.

PHP est un langage de programmation utilisé sur de nombreux serveurs pour prendre des décisions. C'est lui qui décide du code HTML qui sera généré et envoyé au client à chaque fois.

Pour bien comprendre l'intérêt de tout cela, prenons un exemple. On peut écrire en PHP : « **Si le visiteur est membre de mon site et s'il s'appelle Jonathan, affiche** Bienvenue Jonathan **sur la page web. En revanche, si ce n'est pas un membre de mon site, affiche** Bienvenue **à la place et propose au visiteur de s'inscrire** ».

C'est un exemple très basique de site dynamique : selon que vous êtes un membre enregistré ou non, vous ne verrez pas les mêmes choses et n'aurez peut-être pas accès au même contenu.

## En résumé

- Il existe deux types de sites web :
  - les sites statiques : réalisés en HTML et CSS, leur contenu ne peut être mis à jour que par le webmaster ;
  - les sites dynamiques : réalisés avec d'autres outils comme PHP et MySQL en plus de HTML et CSS ; ils permettent aux visiteurs de participer à la vie du site, de poster des messages… bref, de rendre le site vivant !
- Les visiteurs du site sont appelés les clients. Ils demandent au serveur qui héberge le site de leur transmettre les pages web.
- PHP est un langage exécuté par le serveur. Il permet de personnaliser la page en fonction du visiteur, de traiter ses messages, d'effectuer des calculs, etc. Il génère une page HTML.
- MySQL est un système de gestion de bases de données. Il se charge du stockage des informations (liste des messages, des membres).

# 2 Préparer son environnement de travail

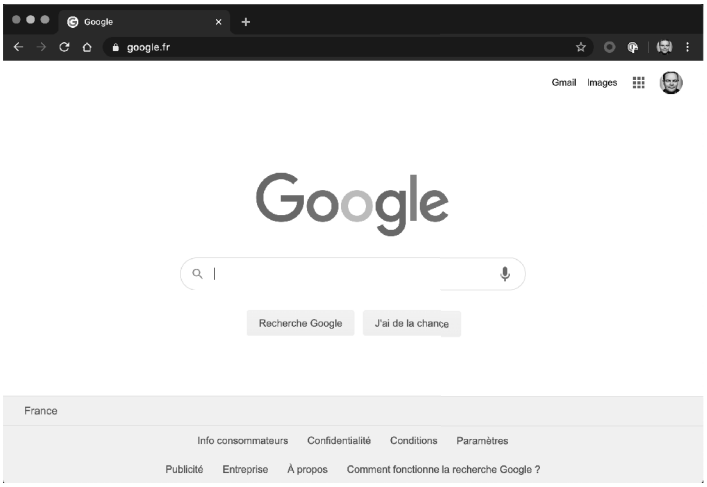
Nous savons désormais que PHP s'exécute sur le serveur et que son rôle est de générer des pages web. Cependant, seul un serveur peut lire du PHP ; or votre ordinateur n'est pas un serveur. Alors, comment créer un site dynamique si PHP ne fonctionne pas chez vous ?

Qu'à cela ne tienne ! Nous allons temporairement transformer votre ordinateur en serveur pour que vous puissiez exécuter du PHP et travailler sur votre site dynamique. Après avoir lu ce chapitre, vous serez fin prêt à programmer !

## Outils de base pour créer un site statique

Les webmasters qui créent des sites statiques avec HTML et CSS ont de la chance, ils ont en général déjà tous les programmes dont ils ont besoin.

- **Un éditeur de texte :** en théorie, un programme tel que le Bloc-notes livré avec Windows suffit, bien qu'il soit recommandé d'utiliser un outil un peu plus évolué comme Visual Studio Code. Nous reparlerons du choix de l'éditeur à la fin de ce chapitre.
- **Un navigateur web :** il permet de tester la page web. Vous pouvez utiliser par exemple Mozilla Firefox, Microsoft Edge, Google Chrome, Opera, Safari, ou tout autre navigateur auquel vous êtes habitué. Il est conseillé de tester son site régulièrement sur différents navigateurs.




Google Chrome

Cependant, pour ceux qui travaillent sur des sites dynamiques, ces outils ne suffisent pas. Il est nécessaire d'installer des programmes supplémentaires.

## Outils pour créer un site dynamique

Pour que votre ordinateur puisse lire du PHP, il faut qu'il se comporte comme un serveur. Rassurez-vous, vous n'avez pas besoin d'acheter une machine spéciale pour cela : il suffit d'installer les mêmes programmes que ceux que l'on trouve sur les serveurs qui délivrent les sites web aux internautes.

- **Apache :** c'est ce qu'on appelle un serveur web. Il s'agit du plus important de tous les programmes, car c'est lui qui est chargé de délivrer les pages web aux visiteurs. Cependant, Apache ne gère que les sites statiques (il ne peut traiter que des pages HTML). Il faut donc le compléter avec d'autres programmes.
- **PHP :** c'est un plug-in pour Apache. En combinant Apache et PHP, notre ordinateur sera capable de lire des pages web en PHP.
- **MySQL :** c'est le logiciel de gestion de bases de données dont je vous parlais précédemment. Il permet d'enregistrer des données de manière organisée (comme la liste des membres de votre site). Nous n'en aurons pas besoin immédiatement, mais autant l'installer sans attendre.

Tous ces éléments sont libres et gratuits. Certes, il en existe d'autres (parfois payants), mais la combinaison Apache + PHP + MySQL est la plus courante sur les serveurs web, à tel point que des « packs » prêts à l'emploi ont été créés. Il est possible de les installer un à un mais cela prend plus de temps et vous n'allez rien y gagner (sauf si vous êtes administrateur de serveur, ce qui ne devrait pas être votre cas).

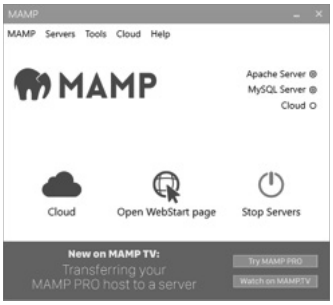
Dans la suite de ce chapitre, nous allons voir comment installer le « pack » correspondant à votre système d'exploitation.

## Sous Windows ou macOS : installer MAMP 4.2

Il existe plusieurs paquetages tout prêts pour Windows et macOS. Je vous propose d'utiliser MAMP, qui fonctionne avec ces deux systèmes. Il a l'avantage d'être régulièrement mis à jour.

Il existe aussi un programme appelé WAMP pour Windows que vous pouvez essayer, mais je le trouve un peu plus compliqué à utiliser que MAMP. Même si vous êtes sous Windows, je vous recommande donc d'essayer MAMP en premier.

1. Commencez par télécharger MAMP 4.2 à l'adresse suivante : *https://www.mamp.info/en/downloads/older-versionm/*.
2. Ensuite, installez-le. Il y a peu d'options. Si on vous propose d'installer MAMP Pro, dites non, car ce programme est payant et nous n'en aurons pas besoin. MAMP seul nous suffit.
3. Lancez MAMP. La fenêtre suivante devrait apparaître.




MAMP une fois démarré

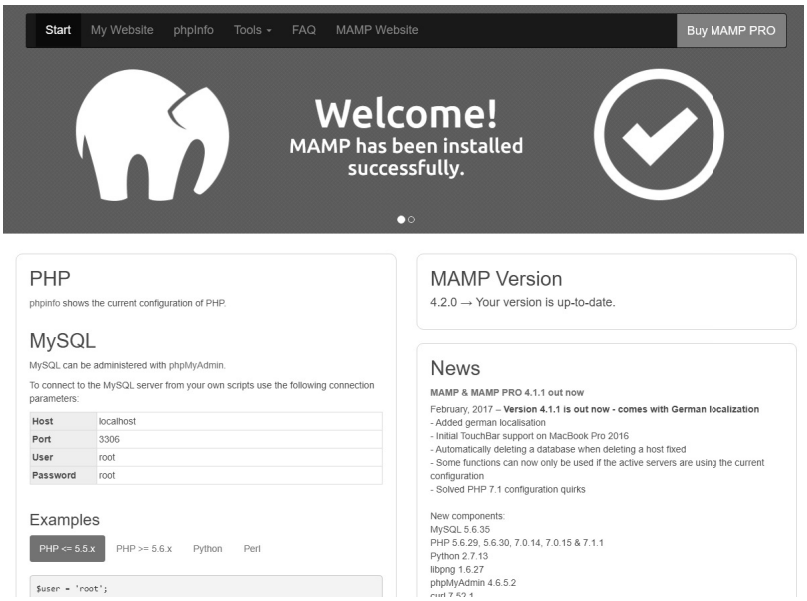
MAMP met régulièrement à jour son logiciel et les captures d'écran et fonctionnalités présentées dans ce cours pourraient ne plus être identiques. Pour bien suivre ce cours, installez la version 4.2 du logiciel, comme recommandé.

Lorsque MAMP démarre, il lance à son tour les deux programmes importants en fond : Apache et MySQL (il faudra peut-être cliquer sur ***Start Servers***). Vous devriez voir les petites diodes s'afficher en vert en haut à droite de la fenêtre. Il faut parfois un petit moment avant que ces programmes ne démarrent.

Si une fenêtre apparaît pour vous indiquer que le pare-feu bloque Apache ou MySQL, cliquez sur ***Autoriser l'accès***. Vous n'avez aucune raison de vous inquiéter, c'est parfaitement normal.

Si MAMP ne se lance pas correctement malgré tout, vérifiez que Skype n'est pas ouvert en même temps. Les deux programmes ne peuvent pas fonctionner simultanément car ils utilisent les mêmes ports de communication sur votre machine. Si c'est le cas, il suffit de fermer Skype pendant que vous utilisez MAMP.

Lorsque MAMP a bien lancé Apache et MySQL, cliquez sur le bouton ***Open WebStart Page***, au milieu, pour ouvrir la page d'accueil de MAMP dans votre navigateur.




Page d'accueil de MAMP dans le navigateur

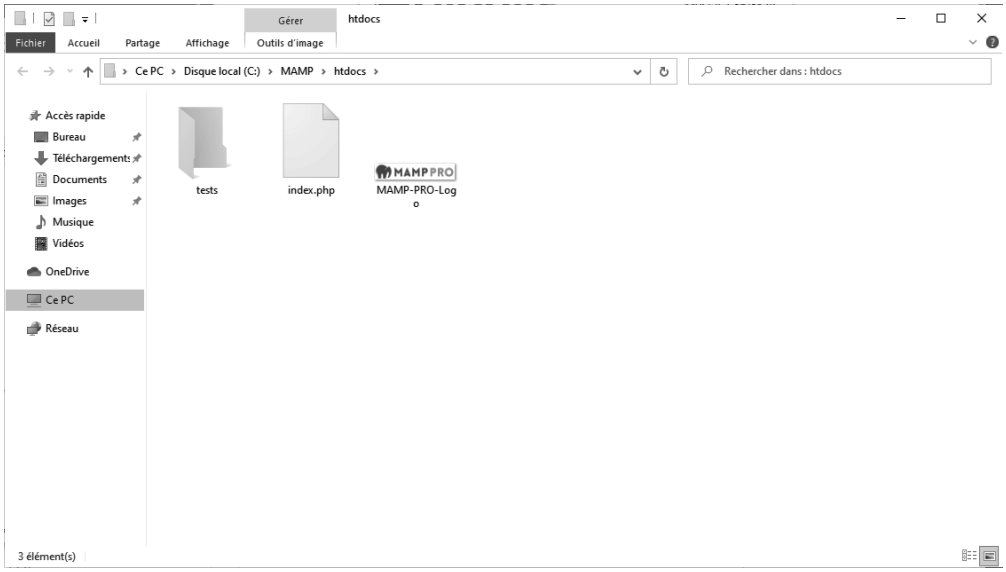
Vous cliquerez sur ***My Website***, en haut à gauche, pour ouvrir le site que vous allez créer. Celui-ci devra être placé dans :

- `c:\MAMP\htdocs` sous Windows ;
- `/Applications/MAMP/htdocs` sous macOS.

Vous pouvez changer ce répertoire par défaut dans les préférences de MAMP, section ***Web Server***, où vous modifierez ***Document root***.

Pour l'instant, si vous ouvrez ***My Website***, il n'y a rien. Créons un projet que nous appellerons `tests` :

1. Ouvrez l'Explorateur Windows (ou le Finder macOS) et rendez-vous dans le dossier racine `htdocs`.
2. Créez le sous-dossier `tests`.




Capture d'écran du dossier « tests » sur l'ordinateur

3. Supprimez le fichier `index.php` (ici, il ne nous servira à rien).
4. Retournez sur la page d'accueil de MAMP et allez dans ***My Website***. Vous devriez voir un dossier `tests` apparaître.


Le dossier « tests » apparaît dans le navigateur.

Au besoin, vous pouvez vous rendre directement dans ce dossier depuis votre navigateur, à l'adresse *http://localhost/tests* ou *http://localhost:8888/tests* (selon votre configuration).

5. Cliquez sur le dossier pour l'ouvrir. Il n'y a rien à l'intérieur pour le moment.


Le dossier « tests » est vide.

Si vous obtenez ce résultat, cela signifie que tout fonctionne. Bravo ! Vous avez installé MAMP avec succès. Vous êtes prêt à programmer en PHP. Vous pouvez passer la section suivante, qui ne concerne que les utilisateurs de Linux.

## Sous Linux : installer XAMPP

Sous Linux, il est courant d'installer Apache, PHP et MySQL séparément. Toutefois, il existe aussi des packs tout prêts comme XAMPP (pour *X, Apache, MySQL, Perl, PHP*), anciennement connu sous le nom de LAMPP.

Ce pack est plus complet que MAMP. Nous n'utiliserons toutefois qu'une partie des éléments installés.

Sur le site officiel de XAMPP (*https://www.apachefriends.org/fr/index.html*), recherchez le lien ***XAMPP pour Linux***.


Téléchargement de XAMPP pour Linux

XAMPP est aussi disponible pour Windows et macOS comme vous pourrez le constater sur le site. La méthode d'installation est sensiblement différente, mais vous pouvez l'essayer si vous avez déjà testé MAMP et s'il ne vous convient pas.

Une fois le téléchargement terminé, ouvrez une console. L'installation et le lancement de XAMPP se font en effet uniquement en console. Allons, allons, pas de chichis, vous n'allez pas me faire avaler que c'est la première fois que vous ouvrez la console !

Rendez-vous dans le dossier où vous avez téléchargé XAMPP. Par exemple, dans mon cas, le fichier se trouve sur le bureau :

```
cd  /Desktop
```

Vous devez passer `root` pour installer et lancer XAMPP.

`root` est le compte administrateur de la machine, qui a notamment le droit d'installer des programmes. Normalement, il suffit de taper `su` et d'entrer le mot de passe `root`. Sous Ubuntu, il faudra taper `sudo su` et taper votre mot de passe habituel.

Si comme moi vous utilisez Ubuntu, tapez donc :

```
sudo su
```

Donnez les droits d'exécution au fichier que vous venez de télécharger :

```
chmod 755 xampp-linux-*-installer.run
```

Puis lancez le programme d'installation :

```
./xampp-linux-*-installer.run
```

Pensez à remplacer le caractère * dans la commande par le numéro de version du fichier téléchargé.

Et voilà ! XAMPP est maintenant installé.

Pour démarrer XAMPP (et donc Apache, PHP et MySQL), tapez la commande suivante :

```
/opt/lampp/lampp start
```

Si vous désirez par la suite arrêter XAMPP, tapez :

```
/opt/lampp/lampp stop
```

N'oubliez pas que vous devez être sous le compte `root` lorsque vous démarrez ou arrêtez XAMPP.

Ce n'est pas bien compliqué, comme vous pouvez le voir !

Vous pouvez maintenant tester XAMPP en ouvrant votre navigateur favori et en tapant l'adresse suivante : *http://localhost*.

Les fichiers PHP devront être placés dans le répertoire `/opt/lampp/htdocs`. Vous pouvez y créer un sous-répertoire `tests` pour vos premiers tests.

```
cd /opt/lampp/htdocs
mkdir tests
```

Une fois le dossier créé, vous pouvez y accéder depuis votre navigateur à l'adresse suivante : *http://localhost/tests*.

## Découvrir le serveur PHP intégré

Nous avons installé des logiciels qui reproduisent le comportement exact d'un serveur tel qu'il serait configuré et installé en ligne. Cependant, pour de petits travaux sur votre ordinateur « en local », PHP fournit un serveur web interne très pratique et qui utilise la ligne de commande pour provoquer l'exécution du script et le rendu de la page.

Par exemple, créons un fichier PHP `index.php` avec le contenu suivant :

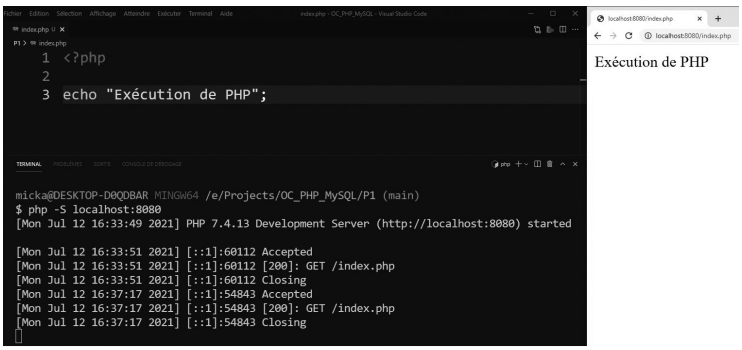
```
<?php
echo "Bonjour";
```

Exécutons-le en ligne de commande **au niveau du dossier où se trouve le fichier** `index.php` :

```
php -S localhost:8080
```

`localhost` est votre « nom de domaine » local. `8080` est un port HTTP quelconque.

Ensuite, en accédant à *http://localhost:8080/index.php*, le retour de l'exécution de ce script PHP sera disponible. Pratique, non ?




Le serveur interne de PHP en action !

Par la suite, nous utiliserons l'une ou l'autre des options proposées.

Vous êtes prêt à travailler en PHP !

## Utiliser un bon éditeur de texte

Comme vous devez déjà le savoir, vous disposez de plusieurs solutions pour éditer le code d'une page web :

- Utiliser un éditeur de texte tout simple, déjà présent sur votre ordinateur, comme le **Bloc-notes**. Ce logiciel suffit normalement à écrire des pages web en HTML et même en PHP, mais…
- le mieux reste d'utiliser un **logiciel spécialisé** qui colore votre code (très pratique) et qui numérote vos lignes (très pratique aussi). Il existe des centaines et des centaines de logiciels gratuits prévus pour les développeurs comme vous. Je vais vous en présenter ici deux : un gratuit (Visual Studio Code) et un payant (PHPStorm).

Je vous propose donc d'installer un logiciel qui va éditer efficacement vos fichiers sources. Vous en avez probablement déjà installé un si vous avez appris à programmer en HTML/CSS mais, comme on n'est jamais trop prudent, je vais rapidement vous en présenter quelques-uns en fonction de votre système d'exploitation.

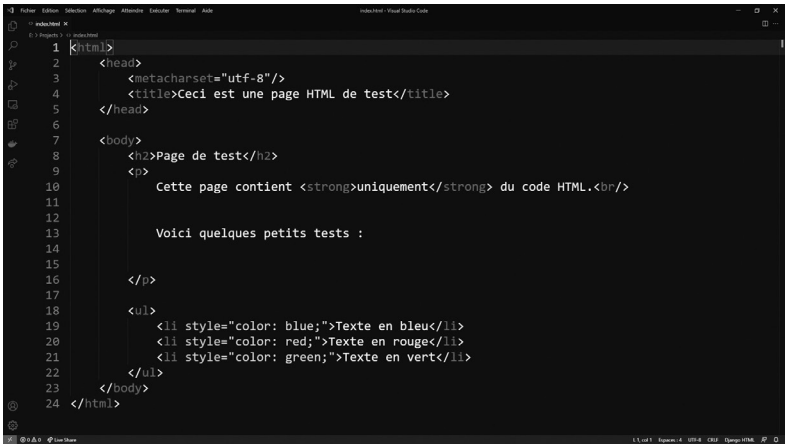
Voici le code source HTML que nous allons utiliser pour commencer en terrain connu. Copiez-collez-le dans l'éditeur de texte que vous allez installer :

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Ceci est une page HTML de test</title>
   </head>
   <body>
      <h2>Page de test</h2>

      <p>
         Cette page contient <strong>uniquement</strong> du code.<br />
Voici quelques petits tests :
      </p>

      <ul>
         <li style="color: blue;">Texte en bleu</li>
         <li style="color: red;">Texte en rouge</li>
         <li style="color: green;">Texte en vert</li>
      </ul>
   </body>
</html>
```

Pour l'instant, le code ne contient pas de PHP afin de commencer en douceur. Nous allons simplement essayer d'enregistrer un fichier HTML avec ce code pour nous échauffer.

## Visual Studio Code

Que vous soyez sous Windows, macOS ou Linux, je vous recommande de commencer par l'éditeur Visual Studio Code (*https://code.visualstudio.com*), qui est suffisamment léger et simple pour des débutants.

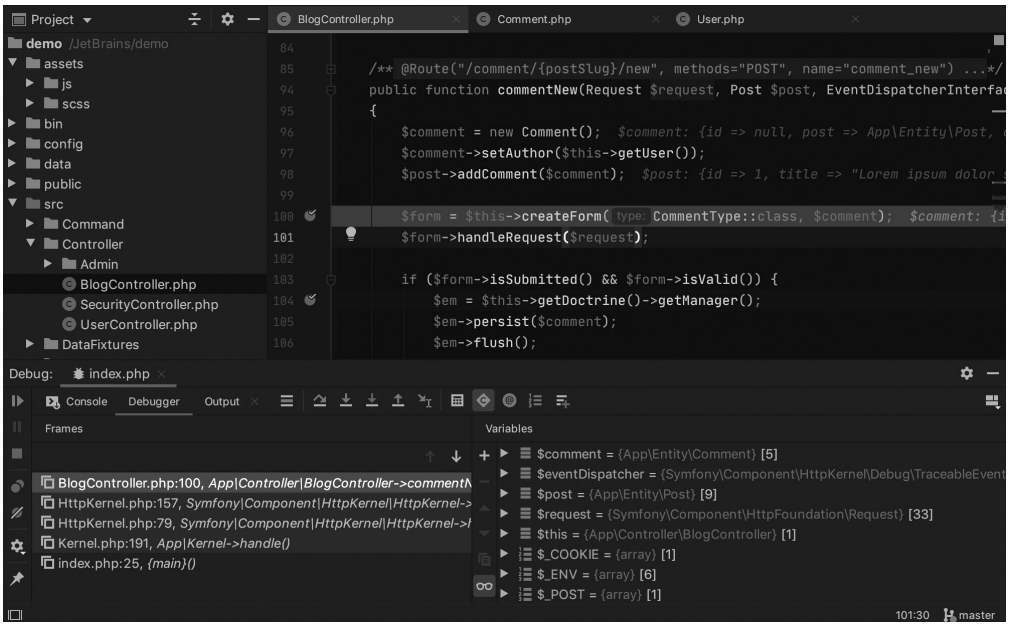



L'éditeur de texte Visual Studio Code

Ne vous fiez pas à son apparente simplicité : Visual Studio Code est en effet rapide et simple à la base, mais il est possible d'étendre ses fonctionnalités avec d'innombrables modules !

Visual Studio Code est un très bon éditeur, utilisé par de nombreux développeurs (y compris des professionnels). Il voit en revanche ses limites sur de gros projets, où certains lui préfèrent PHPStorm.

### PHPStorm




L'éditeur PHPStorm, utilisé par de nombreux professionnels

PHPStorm (*https://www.jetbrains.com/phpstorm/*) ressemble un peu plus à une « machine de guerre ». Et pour cause : c'est un IDE, un environnement de travail de développeur. Il est utilisé par de nombreux développeurs PHP professionnels de ma connaissance.

Vous ne commencerez peut-être pas immédiatement avec PHPStorm, mais rangez-le dans un coin de votre tête car c'est un outil très utilisé que vous essaierez sûrement un jour.

## En résumé

- Pour créer des sites web dynamiques, nous devons installer des outils qui transformeront notre ordinateur en serveur afin de tester notre site.
- Les principaux outils dont nous avons besoin sont :
  - Apache : le serveur web ;
  - PHP : le programme qui permet au serveur web d'exécuter des pages PHP ;
  - MySQL : le logiciel de gestion de bases de données.
- Bien qu'il soit possible d'installer ces outils séparément, il est plus simple pour nous d'installer un paquetage tout prêt : MAMP sous Windows ou macOS ou XAMPP sous Linux.

Il est conseillé d'utiliser un éditeur de texte qui colore le code source, comme Visual Studio Code, pour programmer convenablement en PHP.

Pour les personnes plus expérimentées qui travaillent sur de gros projets, je recommande PHPStorm.

# 3 Écrire son premier script

Dans le premier chapitre, nous avons découvert le principe de fonctionnement du PHP. Ici, nous allons passer à la pratique et réaliser notre toute première page web en PHP.

Ne vous attendez pas à un résultat extraordinaire, mais cela va vous permettre de prendre vos marques. Vous allez en particulier comprendre comment on sépare le code HTML classique du code PHP.

Vous êtes prêt ? Allons-y !

## Les balises PHP

Vous savez donc que le code source d'une page HTML est constitué de **balises** ***(tags)***. Par exemple, `<ul>` est une balise.

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

Le code suivant illustre cela.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Ma page web</title>
  </head>

  <body>
    <h1>Ma page web</h1>

    <p>
```

```
        Bonjour <!-- Insérer le pseudo du visiteur ici --> !
      </p>

    </body>
</html>
```

Comme vous pouvez le voir, on retrouve le code HTML qu'on connaît bien… et on insère en plus des données dynamiques au milieu. Ici, par exemple, c'est le pseudonyme : il change en fonction du visiteur.

OpenClassrooms fait la même chose pour ses membres inscrits. Votre pseudonyme est affiché en haut des pages lorsque vous êtes connecté.

## La forme d'une balise PHP

Pour utiliser du PHP, on va devoir introduire une nouvelle balise… un peu spéciale. Elle commence par `<?php` et se termine par `?>`. Le code PHP sera placé à l'intérieur de cette balise, comme nous allons le voir.

Voici une balise PHP vide :

```
<?php ?>
```

À l'intérieur, on écrira donc du code source PHP :

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

On peut sans problème écrire la balise PHP sur plusieurs lignes. En fait, c'est même indispensable la plupart du temps. Cela donnera quelque chose comme :

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

Il existe d'autres balises pour utiliser du PHP, par exemple `<? ?>`, `<% %>`, `<?= ?>`, etc. Ne soyez donc pas étonné si vous en voyez. Néanmoins, `<?php ?>` est la forme la plus correcte ; vous apprendrez donc ici à vous servir de cette balise et non pas des autres.

Insérer une balise PHP au milieu du code HTML

Comme indiqué, la balise PHP que nous venons de découvrir s'insère au milieu du code HTML. Reprenons l'exemple du chapitre précédent :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ceci est une page de test avec des balises PHP</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <h2>Page de test</h2>

    <p>
      Cette page contient du code HTML avec des balises PHP.<br />
      <?php /* Insérer du code PHP ici */ ?>
      Voici quelques petits tests :
    </p>

    <ul>
      <li style="color: blue;">Texte en bleu</li>
      <li style="color: red;">Texte en rouge</li>
      <li style="color: green;">Texte en vert</li>
    </ul>

    <?php
     /* Encore du PHP
     Toujours du PHP */
    ?>
  </body>
</html>
```

Peut-on placer une balise PHP n'importe où dans le code ?

Oui ! Vraiment n'importe où. Pas seulement dans le corps de la page d'ailleurs : vous pouvez aussi en placer dans l'en-tête :

```
<html>
  <head>
    <title>Ceci est une page de test <?php /* Code PHP */ ?></title>
    <meta charset="utf-8" />
  </head>php
```

Plus fort encore, vous pouvez même insérer une balise PHP au milieu d'une balise HTML, comme le montre la ligne 5 de l'exemple ci-après (bon, ce n'est pas très joli, je vous l'accorde) :

```
<!DOCTYPE html>
<html>
  <head>
```

```
    <title>Ceci est une page de test</title>
    <meta <?php /* Code PHP */ ?> charset="utf-8"/>
</head>
```

Comment ça fonctionne ? À quoi ça peut servir ?

Il faut se rappeler que PHP génère du code HTML. Nous allons mieux comprendre le fonctionnement en apprenant à afficher du texte en PHP.

## Afficher du texte

Grande nouvelle : c'est maintenant que vous allez apprendre votre première instruction en PHP. Ne vous attendez pas à quelque chose d'extraordinaire, votre PC ne va pas se mettre à danser la samba tout seul. ;-)

Vous allez cependant un peu mieux comprendre le fonctionnement de PHP, c'est-à-dire comment il génère du code HTML. Il est indispensable de bien comprendre cela ; soyez donc attentif !

### L'instruction echo

Le PHP est un langage de programmation, ce qui n'était pas le cas du HTML. Dans ce cours, nous partons de zéro, donc je vais supposer que vous n'avez jamais fait de programmation auparavant.

Tout langage de programmation contient ce qu'on appelle des **instructions**. On en écrit une par ligne, en général, et en PHP elles se terminent toutes par un point-virgule. Une instruction commande à l'ordinateur d'effectuer une action précise.

Ici, la première instruction que nous allons découvrir insère du texte dans la page web. Il s'agit de l'instruction `echo`, la plus simple et la plus basique de toutes celles que vous devez connaître.

Voici un exemple d'utilisation de cette instruction :

```
<?php echo "Ceci est du texte"; ?>

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

À l'intérieur de la balise PHP, on écrit l'instruction `echo` suivie du texte à afficher entre guillemets. Ces derniers délimitent le texte, ce qui aide l'ordinateur à se repérer. Enfin, la ligne se termine par un point-virgule, ce qui signifie « fin de l'instruction ».

Notez qu'il existe une instruction identique à `echo` appelée `print`, qui fait la même chose. Cependant, `echo` est plus couramment utilisée.

Il faut savoir qu'on a aussi le droit de demander d'afficher des balises. Par exemple, le code suivant fonctionne :

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

Le mot texte sera affiché en gras grâce à la présence des balises `<strong>` et `</strong>`.

Comment faire pour afficher un guillemet ?

Bonne question. Un guillemet veut dire pour l'ordinateur que le texte à afficher s'arrête là. Vous risquez au mieux de faire « planter » votre beau code et d'avoir une terrible `Parse error`. La solution consiste à faire précéder le guillemet d'une barre oblique inverse `\` :

La solution consiste à faire précéder le guillemet d'un antislash `\` :

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

Vous savez que le code PHP s'insère au milieu du code HTML. Alors allons-y, prenons une page basique en HTML et plaçons-y du code PHP (ligne 12) :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Notre première instruction : echo</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <h2>Affichage de texte avec PHP</h2>

    <p>
      Cette ligne a été écrite entièrement en HTML.<br />
      <?php echo "Celle-ci a été écrite entièrement en PHP."; ?>
    </p>
  </body>
</html>
```

Je vous propose de copier-coller ce code source dans votre éditeur de texte et d'enregistrer la page. Nous allons l'essayer et voir ce qu'elle produit comme résultat.

Mais au fait, vous rappelez-vous comment vous devez enregistrer votre page PHP ?

## Enregistrer une page PHP

Enregistrez la page avec l'extension `.php`, par exemple `affichertexte.php`, dans le dossier `tests` que je vous ai fait créer. Il doit se trouver dans

- `C:\MAMP\htdocs\tests` sous Windows ;
- `/Applications/MAMP/htdocs/tests` sous macOS.

L'essentiel, quel que soit votre système d'exploitation, est que le fichier soit enregistré dans le dossier `htdocs` (ou un de ses sous-dossiers) si vous utilisez MAMP ; sinon le fichier PHP ne pourra pas s'exécuter !

Une fois la page enregistrée, il faut maintenant la tester.

## Tester la page PHP

Pour tester votre page PHP, tout dépend de votre système d'exploitation, mais la manœuvre est la même dans les grandes lignes.

1. Démarrez MAMP (ou XAMPP) si ce n'est pas déjà fait.
2. Allez à l'adresse *http://localhost/tests* ou *http://localhost:8888/tests* (selon votre configuration). Une page web s'ouvre indiquant tous les fichiers qui se trouvent dans le dossier `tests`. Vous devriez voir le fichier `affichertexte.php`.
3. Cliquez dessus : votre ordinateur génère alors le code PHP puis ouvre la page. Vous avez le résultat devant vos yeux.

Le même résultat peut être obtenu dans votre navigateur en allant directement à l'adresse suivante : *http://localhost/tests/affichertexte.php*. La méthode devrait être quasiment la même, que vous soyez sous Windows, macOS X ou Linux.

Si vous utilisez le serveur local de PHP, vous pouvez sauvegarder le fichier où vous le souhaitez mais, en contrepartie, vous devez toujours démarrer le serveur au niveau d'un dossier contenant le fichier à exécuter.

Vous êtes probablement étonné de ce que je vous ai fait faire ; cela semble inutile et ce n'est pas tout à fait faux. Le code PHP a « écrit » une ligne à l'écran, tout simplement.

N'est-ce pas plus simple de l'écrire en HTML ?

Si ! Cependant, vous verrez bientôt l'intérêt de cette fonction. Pour le moment, on constate juste que ça écrit du texte.

En attendant, pour vous amuser et comprendre la force de PHP, essayez juste le code suivant (qu'on expliquera plus tard dans le livre) :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Ma page web</title>
  </head>
  <body>
   <h1>Ma page web</h1>
    <p>Aujourd'hui, nous sommes le <?php echo date('d/m/Y h:i:s'); ?>.</p>
  </body>
</html>
```

Testez ce code et admirez : la date et l'heure s'affichent automatiquement sur la page. Attendez quelques minutes puis actualisez la page : l'heure s'est mise à jour toute seule.

Regardez le code source de la page générée dans le navigateur : vous verrez qu'il n'y a pas de code PHP. L'heure a directement été envoyée dans le code HTML après exécution du code PHP par le serveur.

## Commenter le code

Avant de clore ce chapitre, je tiens à vous parler de quelque chose qui à mes yeux a une très grande importance quand on développe des programmes : les commentaires.

Un **commentaire** est un texte que l'on écrit pour soi (ou pour d'autres développeurs qui vont travailler sur le programme) dans le code PHP. Ce texte est ignoré, c'est-à-dire qu'il disparaît complètement lors de la génération de la page. Il n'y a que vous qui le voyiez.

Cela vous aide à vous y retrouver dans votre code PHP, parce que, si vous n'y touchez pas pendant des semaines avant d'y revenir, vous risquez d'être un peu perdu.

Vous pouvez écrire tout et n'importe quoi, l'essentiel étant de s'en servir à bon escient.

Il existe deux types de commentaires : monolignes ou multilignes.

### Les commentaires monolignes

Pour indiquer que vous écrivez un commentaire sur une seule ligne, vous devez taper deux barres obliques `//`. Ajoutez ensuite votre commentaire.

```
<?php
echo "J'habite en Chine."; // Cette ligne indique où j'habite

// La ligne suivante indique mon âge
echo "J'ai 92 ans.";
?>
```

Dans cet exemple, j'ai mis deux commentaires à des endroits différents :

- le premier est à la fin d'une ligne (mieux pour commenter une ligne précise) ;
- le second prend toute une ligne (mieux pour séparer en blocs fonctionnels).

### Les commentaires multilignes

Ils sont évidemment pensés pour écrire un commentaire sur plusieurs lignes, mais on peut aussi s'en servir pour écrire des commentaires d'une seule ligne. Il faut délimiter le commentaire par `/*` et `*/` :

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

Ici, les commentaires n'ont pas grande utilité, mais vous verrez de quelle façon je les utilise dans les prochains chapitres pour vous décrire le code PHP.

## En résumé

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

# 4 Configurer PHP pour visualiser les erreurs

Avant d'aller plus loin, il est important de faire une pause sur ce que vous allez sans doute beaucoup rencontrer (et ce n'est absolument pas grave !) : les erreurs.

En effet, lorsqu'un script PHP « plante », le comportement par défaut de PHP est de n'afficher qu'une page blanche (une page de navigateur sans contenu).

Pour faciliter notre vie de développeur, il va falloir faire en sorte que les erreurs PHP s'affichent. Sinon, nous aurons de grosses difficultés par la suite pour comprendre pourquoi nos pages ne s'affichent pas correctement.

Nous allons donc changer la configuration de PHP.

## Configurer PHP pour afficher les erreurs

Eh oui, PHP est configurable !

Si les erreurs s'affichent déjà dans votre navigateur, il est inutile de procéder aux manipulations qui vont suivre !

Par défaut, PHP n'affiche pas les erreurs : ainsi, il évite de donner trop d'indications aux utilisateurs, pour des raisons évidentes de sécurité. Retenez ce mantra à vous répéter : « moins l'utilisateur en sait sur mon application, mieux mon application se portera ! ».

La configuration de PHP se fait dans un fichier appelé `php.ini`. Encore faut-il savoir où il se trouve !

## Localiser le fichier de configuration PHP du serveur web

Pour connaître l'ensemble des informations relatives au PHP utilisé par le serveur web, il existe une fonction PHP nommée `phpinfo()` (nous reviendrons sur les fonctions). Nous allons l'utiliser pour localiser le fichier de configuration afin de le modifier.

Je vous invite donc à créer un fichier PHP avec le code qui suit :

```
<?php
phpinfo();
?>
```

Enregistrez-le sous le nom `info.php` dans le dossier qui est lu par votre serveur web (normalement, `htdocs` ou `www`). Et enfin, affichez la page. Vous devriez obtenir le résultat qui s'affiche sur la figure suivante.


| | |
|---|---|
| **System** | Darwin LAPM-201911-01 19.5.0 Darwin Kernel Version 19.5.0: Thu Apr 30 18:25:59 PDT 2020; root:xnu-6153.121.1~7/RELEASE_X86_64 x86_64 |
| **Build Date** | Feb 17 2020 12:49:52 |
| **Configure Command** | './configure' '--with-apxs2=/Applications/MAMP/Library/bin/apxs' '--with-zlib' '--with-zlib-dir=/Applications/MAMP/Library' '--prefix=/Applications/MAMP/bin/php/php7.4.2' '--exec-pr… sysconfdir=/Applications/MAMP/bin/php/php7.4.2/conf' '--with-config-file-path=/Applications/MAMP/bin/php/php7.4.2/conf' '--enable-ftp' '--with-bz2=/Applications/MAMP/Library' '… curl=/Applications/MAMP/Library' '--enable-sockets' '--enable-bcmath' '--enable-soap' '--enable-calendar' '--with-pgsql=shared,/Applications/MAMP/Library/pg' '--enable-exif' '--with… xsl=/Applications/MAMP/Library' '--with-pdo-mysql=mysqlnd' '--with-pdo-pgsql=shared,/Applications/MAMP/Library/pg' '--with-openssl=/Applications/MAMP/Library' '--with-iconv=/… tidy=shared,/Applications/MAMP/Library' '--with-readline' '--with-mhash' '--with-iconv-dir=/Applications/MAMP/Library' '--with-sodium=/Applications/MAMP/Library' '--with-password… kerberos' '--with-pdo-sqlite' '--with-sqlite3' '--with-imap=shared,/Applications/MAMP/Library/lib/imap-2007f/lib' '--with-imap-ssl=/Applications/MAMP/Library' '--disable-phpdbg' '--e… 'KERBEROS_CFLAGS=-I/usr/include' 'KERBEROS_LIBS=-lkrb5' 'SQLITE_CFLAGS= ' 'SQLITE_LIBS=-lsqlite3' 'JPEG_CFLAGS= ' 'JPEG_LIBS=-ljpeg' 'SASL_CFLAGS=-I/usr/include/… |
| **Server API** | Apache 2.0 Handler |
| **Virtual Directory Support** | disabled |
| **Configuration File (php.ini) Path** | /Applications/MAMP/bin/php/php7.4.2/conf |
| **Loaded Configuration File** | /Applications/MAMP/bin/php/php7.4.2/conf/php.ini |

Résultat du script contenant l'instruction phpinfo()

Cette page contient de nombreuses informations, dont la version utilisée (ici, PHP 7.4.2), le type de serveur web (ici Apache) et la localisation du (ou des) fichier(s) de configuration pour PHP.

Notez que vous pouvez aussi accéder au `phpinfo()` depuis le menu ***Tools>php Info*** en haut de la page d'accueil de MAMP.

Retrouvez la ligne `Loaded Configuration File` (ce qui signifie « fichier de configuration chargé ») et regardez la valeur. Dans mon cas, on peut voir ceci :

| | |
|---|---|
| **Loaded Configuration File** | /Applications/MAMP/bin/php/php7.4.2/conf/php.ini |

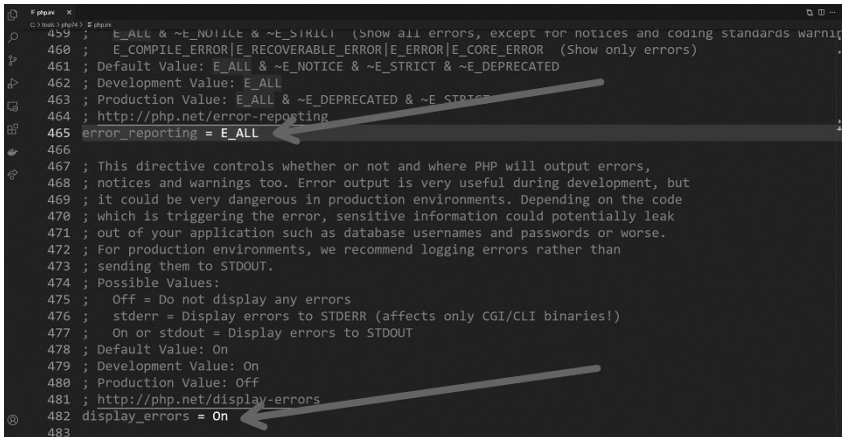
Chemin du fichier de configuration de PHP chargé par le serveur web

## Modifier le fichier de configuration de PHP

Je vais donc ouvrir ce fichier et le modifier. Il faut s'assurer que les clés de configuration `error_reporting` et `display_errors` ont respectivement les valeurs `E_ALL` et `On`.

Allons-y étape par étape.

1. Effectuez une recherche dans le fichier avec le terme `error_reporting`. S'il n'y a pas écrit `error_reporting = E_ALL`, modifiez cette clé pour afficher la valeur correcte.
2. Ensuite, effectuez une nouvelle recherche dans le fichier avec le terme `display_errors`. S'il n'est pas écrit `display_errors = On`, modifiez cette clé pour afficher la valeur correcte.
3. Enregistrez le fichier.
4. Relancez le serveur pour qu'il prenne en compte vos modifications. Il suffit de relancer MAMP par exemple.




Vérifiez que vous avez bien activé les erreurs dans le fichier php.ini

Dans le fichier de configuration, le point-virgule en début de ligne signifie que tout ce qui suit est un commentaire et est donc ignoré. Si l'une de ces lignes (ou les deux) est commentée, il suffit de retirer le point-virgule en début de ligne.

Faites attention à ce que ces lignes de configuration n'existent qu'une seule fois dans le fichier : en effet, ne créez pas ces lignes si elles existaient déjà.

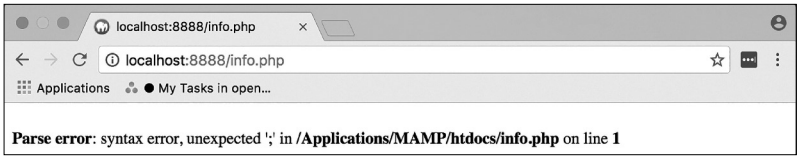
### Tester l'affichage des erreurs

Nous allons maintenant créer une erreur dans un script PHP pour nous assurer que l'erreur s'affiche dans le navigateur : dans le script que nous avons créé pour afficher les informations relatives à PHP pour le serveur web (nous l'avions appelé `info.php`), retirez une parenthèse, puis enregistrez le fichier. Ça devrait donner ceci :

```
<?php
phpinfo(;
?>
```

**Oui je sais, il manque une parenthèse, c'est une erreur, on le fait exprès.**

Maintenant, affichez la page à l'aide de votre navigateur web.




Affichage des erreurs dans le navigateur web

Et voilà ! Si vous voyez bien cette erreur, c'est que PHP est configuré correctement pour travailler. Ouf ! Ça nous fera gagner beaucoup de temps pour comprendre nos problèmes par la suite :)

## En résumé

- PHP dispose d'un mode de débogage pour afficher les erreurs contenues dans vos scripts.
- Pour activer le débogage, on modifie la configuration de PHP en éditant le fichier **`php.ini`**.
- On peut connaître la localisation de ce fichier (et bien d'autres informations) en exécutant la fonction **`phpinfo()`** : `<?php phpinfo() ;?>`.
- Pour activer l'affichage, on change la propriété **`display_errors`** à **`On`** et on filtre le type d'erreur avec **`error_reporting`** (on choisira **`E_ALL`** pour voir toutes les erreurs).

# Deuxième partie

# Les bases de PHP

Passons aux choses sérieuses. Découvrons PHP en douceur dans cette deuxième partie et démarrons le projet fil rouge pour développer notre site web dynamique destiné à partager des recettes de cuisine.

# 5 Les variables

Attention, ce chapitre est fondamental !

Les variables sont un élément indispensable à tout langage de programmation et, en PHP, on n'y échappe pas. Elles servent à retenir temporairement des informations en mémoire. Avec elles, nous allons pouvoir, par exemple, retenir le pseudonyme du visiteur, effectuer des calculs et bien d'autres choses.

Pour développer notre projet de partage, nous aurons besoin de structurer notre application autour des objets qui la composent, ce qu'on appelle les « objets métier ». Pour un site de partage de recettes, c'est simple : des utilisateurs se connectent (ils ont un nom, une adresse e-mail, un mot de passe, un âge...), ils consultent ou créent des recettes (elles ont un titre, un corps, un statut d'activation...) et ainsi de suite pour chacun des objets qui constituent le projet.

## Qu'est-ce qu'une variable ?

Rien qu'avec le nom, vous devez vous dire que c'est quelque chose qui change tout le temps. En effet, le propre d'une variable c'est de pouvoir **varier**. 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.

C'est à vous de créer des variables, dès que vous en avez besoin, pour retenir des informations.

### Un nom et une valeur

Une variable est toujours constituée de deux éléments :

- **son nom** : pour pouvoir la reconnaître, vous devez donner un nom à votre variable, par exemple `age` ;
- **sa valeur** : c'est l'information qu'elle contient et qui peut changer, par exemple `17`.

Ici, je vous ai donné l'exemple d'une variable appelée `age` qui a pour valeur 17.

On peut modifier quand on veut la valeur de cette variable, l'impliquer dans des opérations, etc. Quand on en a besoin, on l'appelle (par son nom) et elle nous dit gentiment la valeur qu'elle contient.

### Les différents types de variables

Les variables sont capables de stocker différents types d'informations. On parle de **types de données**. Voici les principaux types à connaître.

- **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 une valeur logique vraie (`true`) ou fausse (`false`).
- **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 donnée, mais plutôt une **absence** de type.

Cela devrait vous donner une idée de tout ce que PHP peut stocker en mémoire. Ces types suffiront pour la création de notre site.

Maintenant, passons aux choses concrètes. Comment créer une variable et afficher ce qu'elle contient ?

## Affecter une valeur à une variable

### Premières manipulations de variables

Regardez ce code d'exemple :

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

Avec ce code PHP, on vient en fait de créer une variable :

- son nom est `age` ;
- sa valeur est `17`.

On ne peut pas mettre d'espace dans un nom de variable. On utilise donc des initiales majuscules pour « détacher » visuellement les mots et les rendre plus lisibles. C'est ce qu'on appelle la convention ***camelCase*** (cela fait référence aux bosses d'un chameau). Pour le nom, évitez aussi les accents, les cédilles et tout autre symbole : PHP ne les apprécie pas trop… C'est pour cela que j'ai écrit `age` et non `âge`.

Analysons dans le détail le code qu'on vient de voir.

- D'abord, on écrit le symbole « dollar » (`$`) : il précède toujours le nom d'une variable. C'est comme un signe de reconnaissance, qui indique à PHP « j'utilise une variable ».
- Ensuite, il y a le signe « égal » (=) : celui-là, c'est logique, sert à dire que la variable `$age` est égale à…
- À la suite, il y a la valeur de la variable, ici `17`.
- Enfin, il y a l'incontournable point-virgule (`;`) qui termine l'instruction.

Concrètement, le code précédent n'affiche rien du tout. Tant que vous n'utilisez pas `echo`, rien ne s'affiche. Là, le serveur a juste créé la variable temporairement en mémoire, mais il n'a rien fait d'autre.

Supposons maintenant qu'on écrive ceci :

```
<?php
$age = 17; // La variable est créée et vaut 17
$age = 23; // La variable est modifiée et vaut 23
$age = 55; // La variable est modifiée et vaut 55
?>
```

La variable `$age` est créée et prend pour valeur, dans l'ordre : `17`, `23`, puis `55`. Tout cela va très vite : l'ordinateur étant très rapide vous n'aurez pas le temps de dire « ouf » que tout ce code PHP aura été exécuté.

Rien ne s'affiche mais, quelque part dans la mémoire de l'ordinateur, une petite zone nommée `age` vient de prendre successivement les valeurs `17`, `23`, puis `55`.

## Utiliser les types de données

Voici un exemple de variable pour chacun des types présentés précédemment : `string`, `int`, `float`, `bool`, `null`.

### Le type string (chaîne de caractères)

Ce type permet de stocker du texte. Pour cela, vous devez entourer votre texte de guillemets doubles `""` ou de guillemets simples `''` (attention, ce sont des apostrophes).

```
<?php
$nom = "Mathieu Nebra";
$email = 'mathieu.nebra@exemple.com';
?>
```

Attention, petit piège : si vous voulez insérer un guillemet simple alors que le texte est entouré de guillemets simples, il faut l'échapper comme on l'a vu précédemment en insérant devant une barre oblique inverse. Il en va de même pour les guillemets doubles.

```
<?php
$variable = "Mon \"nom\" est Mathieu";
$variable = 'Je m\'appelle Mathieu';
?>
```

En effet, si vous oubliez la barre oblique inverse, PHP va croire que c'est la fin de la chaîne et il ne comprendra pas le texte qui suivra (vous aurez en fait un message `Parse error`).

En revanche, vous pouvez insérer sans problème des guillemets simples au milieu de guillemets doubles et inversement :

```
<?php
$variable = 'Mon "nom" est Mathieu';
$variable = "Je m'appelle Mathieu";
?>
```

La différence est subtile, faites attention. Il y a d'ailleurs une différence plus importante entre les deux types de guillemets dont nous parlerons plus loin.

### Le type int (nombre entier)

Il suffit tout simplement d'écrire le nombre que vous voulez stocker, sans guillemets.

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

### Le type float (nombre décimal)

Vous devez écrire votre nombre avec un point au lieu d'une virgule. C'est la notation anglaise.

```
<?php
$prix = 57.3;
?>
```

### *Le type bool (booléen)*

Pour dire si une variable est vraie ou fausse, vous devez écrire respectivement le mot `true` ou `false` sans guillemets autour (ce n'est pas une chaîne de caractères).

```
<?php
$estUnAuteur = true;
$estUnAdministrateur = false;
?>
```

### *Une variable vide avec NULL*

Si vous voulez créer une variable qui ne contient rien, vous devez lui passer le mot-clé `NULL` (vous pouvez aussi l'écrire en minuscules : `null`).

```
<?php
$aucuneValeur = NULL;
?>
```

Cela sert simplement à indiquer que la variable ne contient rien, tout du moins pour le moment.

## Afficher et concaténer des variables

Nous avons appris à créer des variables et à stocker des informations à l'intérieur. Mais pour le moment, aucun de nos codes sources n'affiche quoi que ce soit.

### Afficher le contenu d'une variable

Rappelez-vous : on peut afficher du texte avec `echo`. On peut aussi se servir de cette instruction pour afficher la valeur d'une variable :

```
<?php
$nom = 'Mathieu Nebra';
echo $nom;
?>
```

Comme vous le voyez, il suffit d'écrire le nom de la variable que vous voulez afficher.

Au fait, on ne doit pas mettre de guillemets après le `echo` ?

Non ; quand il s'agit d'une variable, on ne met pas de guillemets autour.

Créez un fichier PHP avec ce code source pour le tester. Il est inutile d'ajouter tout le code HTML autour ; ce n'est pas une « vraie » page HTML valide, mais c'est bien suffisant pour nos tests. Le contenu de la variable s'affiche dans la page (ici, `'Mathieu Nebra'`).

## Concaténer

Non, ce n'est pas une insulte. Cela signifie **assemblage**. ;-)

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 par exemple : `"Bonjour Mathieu Nebra et bienvenue sur le site !"`. La concaténation est justement un moyen d'assembler du texte et des variables.

Comment faire cela ? Les petits malins auront l'idée d'écrire trois instructions `echo` :

```
<?php
$nom = "Mathieu Nebra";
echo "Bonjour ";
echo $nom;
echo " et bienvenue sur le site !";
?>
```

Vous pouvez tester et vérifier dans le navigateur que ça fonctionne. Cependant, il y a plus malin. On peut tout écrire sur une ligne. Pour cela, il y a deux méthodes et c'est justement maintenant que l'utilisation des guillemets simples ou doubles va faire la différence.

### *Concaténer avec des guillemets doubles*

Avec des guillemets doubles, c'est le plus simple. Vous pouvez écrire le nom de la variable au milieu du texte et il sera remplacé par sa valeur.

```
<?php
$nom = "Mathieu Nebra";
echo "Bonjour $nom et bienvenue sur le site !";
?>
```

Vous obtenez : `Bonjour Mathieu Nebra et bienvenue sur le site !`.

En effet, lorsque vous utilisez des guillemets doubles, les variables qui se trouvent à l'intérieur sont analysées et remplacées par leur vraie valeur. Cette solution présente

l'avantage d'être facile à utiliser, mais je vous recommande plutôt celle qui utilise des guillemets simples.

### *Concaténer avec des guillemets simples*

Écrivez le code précédent avec des guillemets simples.

```
<?php
$nom = 'Mathieu Nebra';
echo 'Bonjour $nom et bienvenue sur le site !'; // Ne fonctionne pas
?>
```

Vous obtenez : `Bonjour $nom et bienvenue sur le site !`.

Miséricorde ! On ne peut pas concaténer du texte avec des guillemets simples ?

Eh bien si ! Cependant, il va falloir écrire la variable en dehors des guillemets et séparer les éléments les uns des autres à l'aide d'un point :

```
<?php
$nom = 'Mathieu Nebra';
echo 'Bonjour ' . $nom . ' et bienvenue sur le site !';
?>
```

Cette fois, vous obtenez bien le résultat souhaité.

Cela semble bien plus compliqué mais, en fait, c'est cette méthode qu'utilisent les programmeurs expérimentés en PHP. En effet, le code est plus lisible et on repère bien la variable alors que, avec les guillemets doubles, elle était comme « noyée » dans le texte. Par ailleurs, votre éditeur de texte devrait colorer la variable, ce qu'il ne faisait pas pour le code précédent.

Cette méthode d'écriture est aussi légèrement plus rapide, car PHP voit tout de suite où se trouve la variable et n'a pas besoin de la chercher au milieu du texte.

Dorénavant, j'écrirai toutes mes chaînes de caractères entre guillemets simples (à de rares exceptions près) et j'utiliserai la seconde méthode de concaténation.

## Faire des calculs simples

On va maintenant faire travailler votre ordinateur et vous allez voir qu'il encaisse les calculs sans problème. Eh oui, PHP sait aussi calculer !

Oh je vous rassure, on ne va pas se lancer dans des calculs tordus, juste des additions, des soustractions, des multiplications et des divisions. C'est du niveau de tout le monde, non ? ;-)

Ici, comme vous vous en doutez, on ne va travailler que sur des variables qui contiennent des nombres.

## Les opérations de base

Les signes à connaître pour effectuer les quatre opérations de base (vous les trouverez sur le pavé numérique de votre clavier, à droite en principe) sont représentés par le tableau suivant. En complément, vous avez l'opération modulo, c'est-à-dire le reste d'une division entière.

**Opérations de base**

| Symbole | Signification |
|---|---|
| + | Addition |
| – | Soustraction |
| * | Multiplication |
| / | Division |
| % | Modulo |

Après, pour vous en servir, cela coule de source. Voici quelques exemples :

```
<?php
$nombre = 2 + 4;         // $nombre prend la valeur 6
$nombre = 5 - 1;         // $nombre prend la valeur 4
$nombre = 3 * 5;         // $nombre prend la valeur 15
$nombre = 10 / 2;        // $nombre prend la valeur 5

// Allez, on ajoute un peu de difficulté
$nombre = 3 * 5 + 1;    // $nombre prend la valeur 16
$nombre = (1 + 2) * 2; // $nombre prend la valeur 6
?>
```

Allons, ne boudez pas, un peu de calcul mental, ça n'a jamais fait de mal à personne. Si vous vérifiez mes calculs, vous verrez qu'il n'y a rien de bien compliqué dans tout ça. Seulement, il ne faut pas avoir peur de « jongler » avec les variables.

Voici des calculs avec plusieurs variables :

```
<?php
$nombre = 10;
$resultat = ($nombre + 5) * $nombre; // $resultat prend la valeur 150
?>
```

C'est de la pure logique, je ne peux rien vous dire de plus. Si vous avez compris ces bouts de code, vous avez tout compris.

### Le modulo

Un autre type d'opération un peu moins connu est le **modulo**, qui représente le reste de la division entière.

Par exemple, `6/3 = 2` et il n'y a pas de reste. En revanche, `7/3 = 2` (car le nombre `3` « rentre » `2` fois dans le nombre `7`) et il reste `1`. Le modulo permet justement de récupérer ce « reste ».

```
<?php
$nombre = 10 % 5; // $nombre prend la valeur 0 car la division tombe juste
$nombre = 10 % 3; // $nombre prend la valeur 1 car il reste 1
?>
```

### Et les autres opérations ?

Je passe sous silence les opérations plus complexes telles que la racine carrée, l'exponentielle, la factorielle, etc. Toutes ces opérations peuvent être réalisées en PHP, mais il faudra passer par ce qu'on appelle des **fonctions**, une notion que l'on découvrira plus tard. Les opérations basiques que nous venons de voir sont amplement suffisantes pour la programmation PHP de tous les jours.

## En résumé

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

# 6 Les conditions

Ce chapitre est lui aussi d'une importance capitale. En effet, vous serez très souvent amené à employer des **conditions** dans vos pages web PHP.

Dans notre projet fil rouge, il nous faudra afficher des informations en fonction du contexte, par exemple autoriser l'auteur d'une recette à la modifier à l'exclusion de tous les autres utilisateurs, ou bien afficher seulement la liste des recettes qui auront été vérifiées par un administrateur...

Vous allez voir que les conditions constituent vraiment la base pour rendre votre site dynamique, c'est-à-dire pour afficher des choses **différentes** en fonction du visiteur, de la date, de l'heure de la journée, etc.

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

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

Celle que je vais vous apprendre à utiliser maintenant est la principale à connaître. Nous en verrons d'autres un peu plus loin.

Nous allons commencer par présenter **les symboles de comparaison** à connaître. Il faut les retenir car ils vous seront utiles pour les conditions.

Ensuite, nous étudierons le fonctionnement de **la structure** `if... else`. Inutile de vous dire qu'il est indispensable de bien comprendre cette partie.

Après, nous compliquerons un peu avec **les conditions multiples**. Vous verrez en effet qu'on peut utiliser plusieurs conditions à la fois.

Enfin, je vous présenterai une **astuce bonus**, parce qu'il y a toujours un bonus pour récompenser ceux qui ont bien suivi jusqu'au bout !

## Les symboles à connaître

Voici les symboles que nous serons amenés à utiliser. Essayez de bien les retenir, car ils vous seront utiles.

**Symboles à connaître et leur signification**

| SYMBOLE | SIGNIFICATION |
|---|---|
| == | Est égal à |
| > | Est supérieur à |
| < | Est inférieur à |
| >= | Est supérieur ou égal à |
| <= | Est inférieur ou égal à |
| != | Est différent de |

Il y a deux symboles « égal » (==) sur la première ligne, à ne pas confondre avec le simple = que nous avons vu 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 (==).

Les symboles « supérieur » (>) et « inférieur » (<) sont situés en bas à gauche de votre clavier.

## La structure if… else

Voici ce qu'on doit écrire, dans l'ordre, pour utiliser cette condition.

1. Pour introduire la condition, on utilise le mot `if`, qui en anglais signifie « si ».
2. On ajoute ensuite, entre parenthèses, la condition (vous allez voir que vous pouvez inventer une infinité de conditions).
3. Enfin, on ouvre des accolades à l'intérieur desquelles on placera les instructions à exécuter si la condition est remplie.

Un exemple vaut toujours mieux qu'un long discours :

```
<?php
$peutEntrer = true; // La condition d'accès

if ($peutEntrer == true) {
   echo "Vous êtes autorisé(e) à accéder au site ✓";
}
?>
```

Ici, on demande à PHP : « si la variable `$peutEntrer` vaut `vrai`, affiche "`Vous êtes autorisé(e) à accéder au site ✓`" ».

Vous remarquerez que, dans la quasi-totalité des cas, c'est sur une variable qu'on exprime la condition.

Ce qui compte ici, c'est qu'il y a deux possibilités :

- soit la condition est remplie et on affiche alors quelque chose ;
- soit elle n'est pas remplie et on saute les instructions entre accolades, on ne fait rien.

Bon, on peut quand même améliorer notre exemple :

```
<?php
$peutEntrer = true;

if ($peutEntrer == true) {
   echo "Vous êtes autorisé(e) à accéder au site ✓";
}
else {
   echo "Accès refusé ✗";
}
?>
```

Tout d'abord, j'ai mis plusieurs instructions entre accolades. Ensuite, vous avez remarqué que j'ai ajouté le mot `else` (« sinon »).

Essayez ce bout de code en modifiant la valeur de `$peutEntrer` (sur la première ligne). Vous allez voir que le message qui s'affiche change en fonction de la valeur que vous indiquez.

Bien entendu, vous mettez les instructions que vous voulez entre accolades. Dans ce qui suit, par exemple, j'ai donné une valeur différente à la variable `$peutEntrer` après avoir affiché un message, valeur qui pourrait nous servir par la suite :

```
<?php
$peutEntrer = "Oui";

// SI on a l'autorisation d'entrer
if ($peutEntrer == "Oui") {
   // instructions à exécuter quand on est autorisé à entrer
} // SINON SI on n'a pas l'autorisation d'entrer
elseif ($peutEntrer == "Non") {
   // instructions à exécuter quand on n'est pas autorisé à entrer
} // SINON (la variable ne contient ni Oui ni Non, on ne peut pas agir)
else {
   echo "Euh, je ne comprends pas ton choix, peux-tu me le rappeler s'il te
plaît ?";
}
?>
```

Ouh là, ça commence à se compliquer un tantinet, n'est-ce pas ?

La principale nouveauté ici, c'est le mot-clé `elseif` qui signifie « sinon si ». Dans l'ordre, PHP rencontre les conditions suivantes.

1. Si `$peutEntrer` est égale à `"Oui"`, tu exécutes ces instructions…
2. Sinon si `$peutEntrer` est égale à `"Non"`, tu exécutes ces autres instructions…
3. Sinon, tu redemandes le choix de l'utilisateur.

Pour vérifier si la variable est vide, vous pouvez taper : `if ($peutEntrer == NULL)`.

## Le cas des booléens

En regardant bien le dernier code source (avec `$peutEntrer`), il serait plus adapté d'utiliser des booléens, ces variables qui valent soit `true` (vrai), soit `false` (faux). Voici comment on teste une variable booléenne :

```
<?php
$peutEntrer = true ;

if ($peutEntrer) {
   echo "Bienvenue petit nouveau. :o)";
}
else {
   echo "Tu n'as pas le droit d'entrer !";
}
?>
```

Un des avantages des booléens est qu'ils sont particulièrement adaptés aux conditions. En effet, vous n'êtes pas obligé d'ajouter `== true`. PHP comprend qu'il doit vérifier si `$peutEntrer` vaut `true`.

Les booléens sont à la fois plus rapides à écrire pour vous et plus compréhensibles. En effet, si vous « lisez » la première ligne, cela donne : « SI on a l'autorisation d'entrer… ».

C'est donc un raccourci à connaître quand on travaille sur des booléens.

Oui mais ta méthode « courte » ne marche pas pour vérifier si le booléen vaut faux. Comment doit-on procéder ?

Il y a un symbole qui permet de vérifier si la variable vaut `false` : le point d'exclamation (`!`). On écrit :

```
<?php
$peutEntrer = true ;

// Si pas autorisé
if (! $peutEntrer) {

}
?>
```

C'est une autre façon de procéder. Si vous préférez écrire `if ($peutEntrer == false)`, c'est tout aussi bien ; toutefois, la méthode « courte » est plus lisible.

## Des conditions multiples

Nous allons essayer de poser plusieurs conditions à la fois. Pour cela, on aura besoin de nouveaux mots-clés. Le tableau suivant reprend les principaux à connaître.

**Mots-clés pour les conditions multiples**

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

Le symbole équivalent pour `OR` est constitué de deux barres verticales. Pour saisir une barre verticale, appuyez simultanément sur les touches ***Alt Gr + 6*** (sur un clavier Azerty français) ou ***Alt Gr + &*** (sur un clavier Azerty belge).

La première colonne contient le mot-clé en anglais, la troisième son équivalent en symbole. Servez-vous de ces mots-clés pour associer plusieurs conditions entre les parenthèses. Voici un premier exemple :

```
<?php
$estActive = true;
$estAuteur = false;

if ($estActive && $estAuteur) {
   echo 'Accès à la recette validé ✓';
} else {
   echo 'Accès à la recette interdit ! ✗';
}
```

C'est tout simple en fait et ça se comprend très bien : si l'utilisateur est actif et s'il est l'auteur, il peut accéder à la recette. Sinon, il verra s'afficher un message de refus.

Bon allez, voici un dernier exemple avec `||` pour que vous l'ayez vu au moins une fois et on arrête là.

```
<?php
$estActive = true;
$estAuteur = false;
$estAdmin = true;

if (($estActive && $estAuteur) || $estAdmin) {
   echo 'Accès à la recette validé ✓';
} else {
   echo 'Accès à la recette interdit ! ✗';
}
```

Nous avons ajouté une condition supplémentaire : soit la condition précédente s'applique, soit l'utilisateur concerné est un administrateur.

## L'astuce bonus

Avec les conditions, il y a une astuce à connaître.

Sachez que les deux codes suivants donnent exactement le même résultat :

```
<?php
$recettesPouletActives = true;

if ($recettesPouletActives) {
   echo '<h1>Liste des recettes à base de poulet</h1>';
}
?>
```

```
<?php $recettesPouletActives = true; ?>

<?php if ($recettesPouletActives): ?> <!-- Ne pas oublier le ":" -->

<h1>Liste des recettes à base de poulet</h1>

<?php endif; ?><!-- Ni le ";" après le endif -->
```

Comme vous le voyez, dans le second cas on n'a pas utilisé l'instruction `echo`. La syntaxe pour utiliser la condition diffère un peu :

- Il n'y a pas d'accolades.
- On ajoute un signe deux-points (`:`) après la parenthèse fermante de l'instruction `if`.
- Et il faut ajouter une instruction `endif ;`.

Nous reviendrons sur cette syntaxe un peu plus loin dans ce cours.

Nous aurons bientôt l'occasion de pratiquer un peu et vous verrez que les conditions sont souvent indispensables.

## Une alternative pratique : switch

En théorie, les structures à base de `if… elseif… else` suffisent pour traiter n'importe quelle condition.

Alors, pourquoi se compliquer la vie avec une autre structure ?

Pour vous faire comprendre l'intérêt de l'instruction `switch`, je vais vous donner un exemple un peu lourd avec les `if` et `elseif` :

```
<?php
$note = 16 ;

if ($note == 0) {
  echo "Tu es vraiment un gros nul !!!";
}

elseif ($note == 5) {
  echo "Tu es très mauvais";
}

elseif ($note == 7) {
  echo "Tu es mauvais";
}

elseif ($note == 10) {
  echo "Tu as pile poil la moyenne, c'est un peu juste…";
}

elseif ($note == 12) {
  echo "Tu es assez bon";
}

elseif ($note == 16) {
  echo "Tu te débrouilles très bien !";
}

elseif ($note == 20) {
  echo "Excellent travail, c'est parfait !";
}

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

Comme vous le voyez, c'est lourd, long et répétitif. Dans ce cas, on peut utiliser une autre structure plus souple : `switch`.

Voici le même exemple avec `switch` (le résultat est le même, mais le code est plus adapté) :

```
<?php
$note = 10;

switch ($note) // On indique sur quelle variable on travaille
{
   case 0: // Dans le cas où $note vaut 0
     echo "Tu es vraiment un gros nul !!!";
   break;

   case 5: // Dans le cas où $note vaut 5
     echo "Tu es très mauvais";
   break;

   case 7: // Dans le cas où $note vaut 7
     echo "Tu es mauvais";
   break;

   case 10: // etc., etc.
     echo "Tu as pile poil la moyenne, c'est un peu juste…";
   break;

   case 12:
     echo "Tu es assez bon";
   break;

   case 16:
     echo "Tu te débrouilles très bien !";
   break;

   case 20:
     echo "Excellent travail, c'est parfait !";
   break;

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

Testez donc ce code ! Essayez de changer la note (dans la première instruction) pour voir comment PHP réagit. Et si vous voulez apporter quelques modifications à ce code (vous allez voir qu'il n'est pas parfait), n'hésitez pas ; cela vous fera de l'entraînement !

Quelles sont les différences ?

- Tout d'abord, il y a beaucoup moins d'accolades (elles marquent seulement le début et la fin du `switch`).
- On indique au début sur quelle variable on travaille (ici, `$note`). On dit à PHP : « Je vais analyser la valeur de `$note`. »
- Après, on utilise des `case` pour analyser chaque cas (`case 0`, `case 10`, etc.). Cela signifie : « Dans le cas où la valeur est 0… Dans le cas où la valeur est 10… »

L'avantage de cela est que l'on n'a plus besoin de mettre le double égal. L'inconvénient est que cela ne fonctionne pas avec les autres symboles (<, >, <=, >=, !=). En clair, le `switch` ne peut tester que l'égalité.

Le mot-clé `default` à la fin est un peu l'équivalent du `else`. C'est le message qui s'affiche par défaut, quelle que soit la valeur de la variable.

Il y a cependant une chose importante à savoir. Supposons dans notre exemple que la note soit de `10`. PHP va lire : `case 0` ? Non. Je saute. `case 5` ? Non plus. Je saute. `case 7` ? Non plus. Je saute. `case 10` ? Oui, j'exécute les instructions. Cependant, contrairement aux `elseif`, PHP ne s'arrête pas là et continue à lire les instructions des `case` qui suivent !

Pour empêcher ce comportement, utilisez l'instruction **`break;`** qui demande à PHP de sortir du `switch`. Dès que PHP tombe sur cette instruction, il sort des accolades et ne lit donc pas les `case` suivants. En pratique, on utilise très souvent un `break` car sinon, PHP lit inutilement des instructions qui ne conviennent pas.

Quand doit-on choisir `if` et quand doit-on choisir `switch` ?

C'est surtout un problème de présentation et de clarté. Pour une condition simple et courte, on utilise le `if` ; quand on a une série de conditions à analyser, on préfère utiliser `switch` pour rendre le code plus clair.

## Les ternaires : des conditions condensées

Il existe une autre forme de condition, beaucoup moins fréquente, mais que je vous présente quand même car vous pourriez un jour ou l'autre tomber dessus. Il s'agit de ce qu'on appelle les **ternaires**.

Un ternaire est une condition condensée qui fait deux choses sur une seule ligne :

- tester la valeur d'une variable dans une condition ;
- affecter une valeur à une variable selon que la condition est vraie ou non.

Prenons l'exemple suivant à base de `if… else` qui met un booléen `$majeur` à vrai ou faux selon l'âge du visiteur :

```
<?php
$age = 24;

if ($age >= 18) {
   $majeur = true;
}
else {
```

```
    $majeur = false;
}
?>
```

On peut écrire la même chose en une seule ligne grâce à une structure ternaire :

```
<?php
$age = 24;

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

// Ou mieux, dans ce cas précis
$majeur = ($age>= 18) ;
?>
```

La condition testée est `$age >= 18`. Si c'est vrai, alors la valeur indiquée après le point d'interrogation (ici, `true`) sera affectée à la variable `$majeur`. Sinon, c'est la valeur qui suit le signe deux-points (`:`) qui sera affectée à `$majeur`.

C'est un peu tordu mais ça marche.

Si vous n'utilisez pas ce type de condition dans vos pages web, je ne vous en voudrai pas. Il faut avouer que les ternaires sont un peu difficiles à lire car ils sont très condensés. Sachez toutefois les reconnaître et les comprendre si vous en rencontrez un jour en lisant le code source de quelqu'un d'autre.

## En résumé

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

# 7 Les boucles

Sur votre site de cuisine, vous aurez sûrement envie de laisser vos utilisateurs commenter les recettes.

Pour cela, nous allons utiliser des tableaux. Il s'agit de structures capables de conserver en mémoire plusieurs éléments. C'est ensuite grâce aux boucles que nous pourrons parcourir les différentes recettes pour les afficher à l'aide du HTML.

Nous reviendrons en détail sur les tableaux dans le chapitre suivant, car c'est l'un des éléments les plus utiles de PHP.

À la fin de ce chapitre, non seulement vous saurez parcourir une liste d'éléments, mais vous aurez également commencé pour de bon à construire votre application.

## Lister des éléments avec un tableau

Reprenons notre projet où nous l'avions laissé. Nous avons des utilisateurs, des recettes et peut-être des commentaires. Avec les connaissances que vous avez pour le moment, voici comment vous pourriez définir deux utilisateurs :

```
<?php
// Premier utilisateur
$nom1 = 'Mickaël Andrieu';
$courriel1 = 'mickael.andrieu@exemple.com';
$motDePasse1 = 'S3cr3t';
$age1 = 34;

// Deuxième utilisatrice
$nom2 = 'Laurène Castor';
```

```
$email2 = 'laurene.castor@exemple.com';
$password2 = 'P4ssW0rD';
$age2 = 28;

// ... et ainsi de suite pour les autres utilisateurs.
?>
```

Pour afficher ces utilisateurs (ou même des recettes), il est inutile de créer des variables pour chacun des éléments qui les caractérisent. En PHP, il existe une structure de type tableau, qui sert à gérer des objets ayant plusieurs propriétés nécessitant d'être rassemblées.

Voici un premier tableau :

```
<?php
$utilisateur1 = ['Mickaël Andrieu', 'mickael.andrieu@exemple.com', 'S3cr3t', 34];

echo $utilisateur1[0]; // "Mickaël Andrieu"
echo $utilisateur1[1]; // "mickael.andrieu@exemple.com"
echo $utilisateur1[3]; // 34
?>
```

Notez pour le moment qu'un tableau se déclare entre crochets (`[]`), qu'il est référencé par des indices **(à partir de `0`**, pas de **`1`)** et qu'on peut accéder à ses éléments à partir de ces indices. Toutefois, la puissance des tableaux ne s'arrête pas là : vous pouvez également construire des tableaux de tableaux.

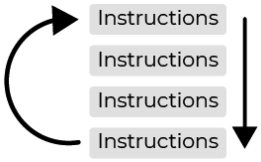
```
<?php
$mickael = ['Mickaël Andrieu', 'mickael.andrieu@exemple.com', 'S3cr3t', 34];
$mathieu = ['Mathieu Nebra', 'mathieu.nebra@exemple.com', 'devine', 33];
$laurene = ['Laurène Castor', 'laurene.castor@exemple.com', 'P4ssw0rD', 28];

$utilisateurs = [$mickael, $mathieu, $laurene];

echo $utilisateurs[1][1]; // "mathieu.nebra@exemple.com"
?>
```

Nous allons maintenant voir comment boucler sur cette liste d'utilisateurs (ou de recettes) pour les afficher.

## Une boucle simple : while

Une boucle est une structure qui fonctionne sur le même principe que les conditions (`if… else`). D'ailleurs, vous allez voir qu'il y a beaucoup de similitudes avec le chapitre précédent.

Concrètement, une boucle sert à répéter plusieurs fois les mêmes instructions. En clair, c'est un gain de temps, c'est très pratique et bien souvent indispensable.




Principe de fonctionnement d'une boucle

Voilà ce qui se passe dans une boucle :

1. Comme d'habitude, les instructions sont d'abord exécutées dans l'ordre, de haut en bas.
2. À la fin des instructions, on retourne à la première.
3. On recommence à lire les instructions dans l'ordre.
4. Et on retourne à la première.
5. Etc.

Le seul hic dans ce schéma, c'est que cela ne s'arrête jamais ! Les instructions seraient exécutées à l'infini.

C'est pour cela que, quel que soit le type de boucle (`while` ou `for`), il faut indiquer une **condition**. Tant que la condition est remplie, les instructions sont réexécutées. Dès que la condition n'est plus remplie, on sort enfin de la boucle.

Voici comment faire avec une boucle simple : `while`.

```
<?php
$continuerBoucle = true;

while ($continuerBoucle == true) {
   // Instructions à exécuter dans la boucle
}
?>
```

`While` peut se traduire par « tant que ». Ici, on demande à PHP : « TANT QUE `$continuerBoucle` est vraie, exécuter ces instructions. »

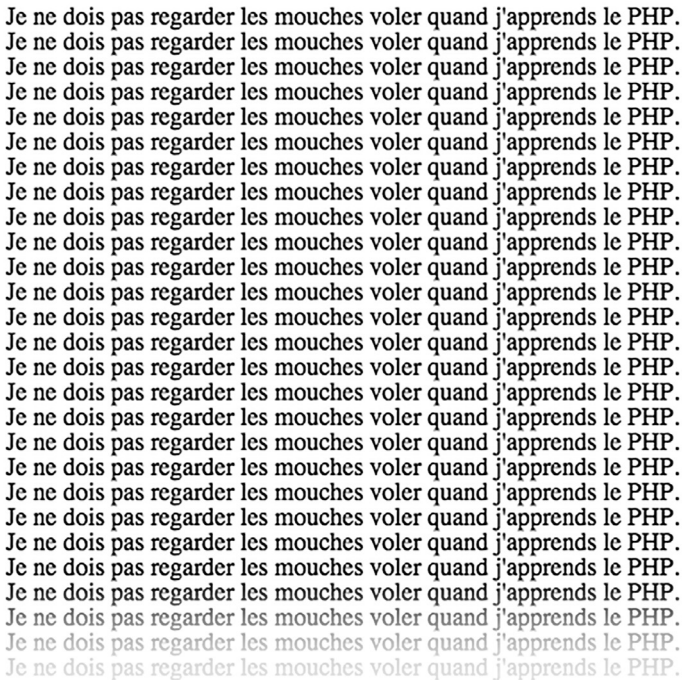
Les instructions qui sont répétées en boucle se trouvent entre les accolades (`{}`), mais je ne vous apprends rien, vous commencez à avoir l'habitude de voir des accolades un peu partout.

Je vais vous montrer quelques exemples d'utilisation de boucles, pour que vous voyiez à quoi ça peut servir…

Pour notre premier exemple, supposons que vous avez été puni et que vous devez copier 100 fois « je ne dois pas regarder les mouches voler quand j'apprends le PHP ». Avant, il fallait prendre son mal en patience et ça durait des heuuuures… Maintenant, avec PHP, on va l'écrire en un clin d'œil :

```
<?php
$ligne = 1;

while ($ligne <= 100) {
   echo 'Je ne dois pas regarder les mouches voler quand j\'apprends le
PHP.<br />';
   $ligne++; // $ligne = $ligne + 1
}
?>
```

Cela affiche… un grand nombre de lignes (voir figure suivante).




Des lignes affichées grâce à une boucle PHP

La boucle pose la condition « TANT QUE `$ligne` est inférieure ou égale à 100 » et contient deux instructions :

- Le `echo` affiche du texte en PHP. Notez qu'il y a une balise HTML `<br />` à la fin, pour aller à la ligne. Chaque phrase sera écrite sur une seule ligne.
- Ensuite vient une instruction bizarre : `$ligne++;` Regardez mon commentaire : c'est exactement la même chose. En fait, c'est une façon plus courte d'ajouter `1` à la variable. On appelle cela **l'incrémentation** (ce nom barbare signifie tout simplement qu'on a ajouté `1` à la variable).

Chaque fois qu'on parcourt la boucle, la valeur de la variable augmente : `1`, `2`, `3`, `4`… `99`, `100`… Dès que la variable atteint `101`, la boucle s'arrête. Et voilà, on a écrit `100` lignes en un clin d'œil.

Si la punition avait été plus grosse, pas de problème ! Il aurait suffi de changer la condition (par exemple, écrire « TANT QUE c'est inférieur ou égal à `500` » pour l'écrire `500` fois).

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

Nous venons donc de voir comment afficher une phrase plusieurs centaines de fois sans effort.

Est-ce vraiment utile ? On n'a pas besoin de faire ça sur un site web !

Pas vraiment, mais comme je vous l'ai dit en introduction, nous apprenons ici des techniques de base qu'on va pouvoir réutiliser dans les prochains chapitres de ce cours. Imaginez que, avec ce système de boucles, vous saurez demander à PHP d'afficher d'une seule traite tous les messages de votre forum. Bien sûr, il vous faudra d'autres connaissances pour y parvenir mais, sans les boucles, vous ne pourriez rien faire.

Voyons un autre exemple. Écrivons toujours une centaine de lignes, mais différentes les unes des autres (on n'est pas obligé d'écrire la même chose à chaque fois), grâce à la concaténation avec la valeur de la variable qui augmente à chaque passage dans la boucle :

```
<?php
$ligne = 1;

while ($ligne <= 100) {
   echo 'Ceci est la ligne n°' . $ligne . '<br />';
   $ligne++;
}
?>

< !--

Ceci est la ligne n°1
Ceci est la ligne n°2
...
-->
```

Cet exemple ressemble beaucoup au précédent. La particularité ici est d'afficher à chaque fois la valeur de `$ligne` (cela vous montre que sa valeur augmente petit à petit).

L'astuce que je vous avais donnée dans le chapitre sur les conditions fonctionne aussi ici : vous pouvez fermer la balise PHP `?>`, écrire du texte en HTML, puis rouvrir la balise PHP `<?php`. Cela vous évite d'utiliser une ou plusieurs instruction(s) `echo` au milieu. On aura l'occasion d'utiliser cette astuce de nombreuses fois dans la suite du cours.

D'accord, mais comment fait-on pour un tableau de tableaux ?

Nous aborderons cette question en détail dans le prochain chapitre. En attendant, voici le code fonctionnel :

```
<?php
$lignes = 3; // nombre d'utilisateurs dans le tableau
$compteur = 0;

while ($compteur < $lignes) {
  echo $utilisateurs[$compteur][0] . ' ' . $utilisateurs[$compteur][1] .
'<br />';
  $compteur++; // Ne surtout pas oublier la condition de sortie !
}
?>
```

En voici le résultat :

Mickaël Andrieu mickael.andrieu@exemple.com
Mathieu Nebra mathieu.nebra@exemple.com
Laurène Castor laurene.castor@exemple.com

Liste des utilisateurs avec une boucle while

## Une boucle plus complexe : for

Mais non, n'ayez pas peur voyons ! Il ne vous arrivera rien de mal : ici, le mot « complexe » ne veut pas dire « compliqué ».

`for` est un autre type de boucle, dans une forme un peu plus condensée et plus commode à écrire, ce qui fait qu'elle est assez fréquemment utilisée.

Cependant, sachez que `for` et `while` donnent le même résultat et servent à la même chose : répéter des instructions en boucle. L'une peut paraître plus adaptée que l'autre dans certains cas, cela dépend aussi des goûts.

Un `for` ressemble beaucoup au `while`, mais c'est la première ligne qui est un peu particulière. Pour que vous compreniez bien la différence avec le `while`, je vais reprendre l'exemple précédent, mais cette fois avec un `for` :

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

Que de choses dans une même ligne ! Après le mot `for`, on trouve des parenthèses qui contiennent trois éléments, séparés par des points-virgules (`;`) :

- Le premier sert à l'**initialisation**. C'est la valeur qu'on donne au départ à la variable (ici, elle vaut `1`).
- Le deuxième est la **condition**. Comme pour le `while`, tant que la condition est remplie, la boucle est réexécutée. Dès que la condition ne l'est plus, on en sort.
- Enfin, le troisième est l'**incrémentation**, qui vous permet d'ajouter `1` à la variable à chaque tour de boucle.

Les deux derniers codes donnent donc exactement le même résultat. Le `for` permet de faire la même chose que le `while`, mais il rassemble sur une seule ligne tout ce qu'il faut savoir sur le fonctionnement de la boucle.

Comment savoir lequel choisir entre un `while` et un `for` ?

La boucle `while` est plus simple et plus flexible : elle permet de réaliser tous les types de boucles, mais on peut oublier d'effectuer certaines étapes comme l'incrémentation de la variable.

En revanche, `for` est bien adapté quand on doit compter le nombre de répétitions des instructions et il évite d'oublier l'incrémentation.

Si vous hésitez entre les deux, il suffit simplement de vous poser la question suivante : « Est-ce que je sais d'avance combien de fois je veux que mes instructions soient répétées ? » Si la réponse est oui, alors la boucle `for` est tout indiquée. Sinon, il vaut mieux utiliser la boucle `while`.

## Afficher des recettes

Reprenons notre description des recettes : un titre, un auteur, un état d'activation et des instructions (la recette à suivre). Le code de votre application à ce stade pourrait être le suivant (avec des recettes d'exemple, bien sûr) :

```php
<?php
// Déclaration du tableau des recettes
$recettes = [
   ['Cassoulet','[...]','mickael.andrieu@exemple.com',true,],
   ['Couscous','[...]','mickael.andrieu@exemple.com',false,],
];
?>


<!DOCTYPE html>
<html>
   <head>
      <title>Affichage des recettes</title>
   </head>
   <body>
      <ul>
         <?php for ($ligne = 0; $ligne <= 1; $ligne++): ?>
            <li><?php echo $recettes[$ligne][0] . ' (' . $recettes[$ligne][2]
. ')'; ?></li>
         <?php endfor; ?>
      </ul>
   </body>
</html>
```

En voici le rendu :

- Cassoulet (mickael.andrieu@exemple.com)
- Couscous (mickael.andrieu@exemple.com)

Affichage de nos recettes (version 0)

Vous noterez que, en HTML, on peut utiliser la boucle `for` comme une boucle `if` en terminant les instructions par `endfor;` au lieu de `endif;`. Par ailleurs, nous avons affiché la recette `"Couscous"` alors que son statut est à `"faux"` ; nous verrons dans le prochain chapitre comment combiner boucles et conditions.

## En résumé

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

# 8 Les tableaux

Nous allons détailler l'une des fonctionnalités présentées dans le chapitre précédent : les tableaux (*arrays*). Il s'agit en fait de variables « composées », que l'on peut imaginer sous la forme de tableaux.

On peut faire énormément de choses avec les tableaux et leur utilisation n'est pas toujours très facile. Cependant, ils vont très rapidement devenir indispensables et vous **devez** bien comprendre leur fonctionnement.

Restons concentrés sur notre projet : nous allons profiter de ce chapitre pour réaliser l'affichage de la liste des recettes.

## Les deux types de tableaux

Un tableau est une variable, mais une variable un peu spéciale.

Reprenons. Jusqu'ici, nous avons travaillé avec des variables toutes simples : elles ont un nom et une valeur. Par exemple :

```
<?php
$titreRecette = 'Cassoulet';
echo 'La recette du ' . $titreRecette;
// Cela affichera : La recette du Cassoulet
?>
```

Cela peut se matérialiser sous la forme suivante :

| Nom | Valeur |
|---|---|
| `$titreRecette` | Cassoulet |

Ici, nous allons voir qu'il est possible d'enregistrer de nombreuses informations dans une seule variable grâce aux tableaux. On en distingue deux types :

- les tableaux numérotés ;
- les tableaux associatifs.

## Les tableaux numérotés

Ils sont très simples à imaginer. Regardez par exemple le tableau suivant, qui reprend le contenu de la variable `$recettes` :

| Clé | Valeur |
|---|---|
| 0 | Cassoulet |
| 1 | Couscous |
| 2 | Escalope milanaise |
| 3 | Salade romaine |
| … | … |

`$recettes` est un **`array`** : c'est ce qu'on appelle une variable « tableau ». Elle n'a pas qu'une seule valeur, mais plusieurs (vous pouvez en mettre autant que vous voulez).

Dans un tableau, les valeurs sont rangées dans des « cases » différentes. Ici, chacune est identifiée par un numéro appelé **clé**.

Un tableau numéroté commence toujours à la case n° 0 ! Ne l'oubliez jamais, ou vous risquez de commettre des erreurs par la suite…

### *Construire un tableau numéroté*

Pour créer un tableau numéroté en PHP, on liste ses valeurs entre crochets `[]` ou on utilise la fonction `array()` :

```
<?php
$recettes = ['Cassoulet', 'Couscous', 'Escalope milanaise', 'Salade romaine'];

// La fonction array permet aussi de créer un tableau
$recettes = array ('Cassoulet', 'Couscous', 'Escalope milanaise', 'Salade
romaine');
?>
```

L'ordre a beaucoup d'importance. Le premier élément (`'Cassoulet'`) aura le n° `0`, `'Couscous'` le n° `1`, etc.

Vous pouvez aussi créer manuellement le tableau case par case :

```
<?php
$recettes[0] = 'Cassoulet';
$recettes[1] = 'Couscous';
$recettes[2] = 'Escalope milanaise';
$recettes[3] = 'Salade romaine';
?>
```

Si vous ne voulez pas avoir à écrire vous-même le numéro de la case que vous créez, laissez PHP le sélectionner automatiquement en laissant les crochets vides :

```
<?php
$recettes[] = 'Cassoulet';            // Créera $recettes[0]
$recettes[] = 'Couscous';             // Créera $recettes[1]
$recettes[] = 'Escalope milanaise'; // Créera $recettes[2]
$recettes[] = 'Salade romaine';      // Créera $recettes[3]
?>
```

### *Afficher un tableau numéroté*

Pour afficher un élément, il faut donner sa position entre crochets après le nom de la variable. Cela revient à dire à PHP, par exemple « Affiche-moi le contenu de la case n° 1 de `$recettes` » :

```
<?php
echo $recettes[1];    // Affichera 'Couscous'
?>
```

Surtout, n'oubliez pas que `'Couscous'` est en deuxième position et qu'il a donc le numéro `1` (étant donné qu'on commence à compter à partir de `0`).

Si vous oubliez les crochets, cela ne fonctionnera pas (et affichera uniquement `'Array'`). Dès que vous travaillez sur des tableaux, vous êtes obligé d'utiliser les crochets pour indiquer dans quelle « case » on doit aller chercher l'information, sinon PHP ne sait pas quoi récupérer.

## Les tableaux associatifs

Les tableaux associatifs fonctionnent sur le même principe, sauf que, au lieu de numéroter les cases, on va les étiqueter en donnant un nom différent à chacune.

Notre objectif ici est d'utiliser un tableau pour décrire une recette. Si le tableau est numéroté, comment savoir que le n° `0` est le titre, le n° `1` la recette, le n° `2` l'auteur… ? C'est là que les tableaux associatifs deviennent utiles.

### Construire un tableau associatif

Pour mieux décrire notre recette, nous allons la stocker sous la forme d'un tableau associatif dans lequel chaque clé est une propriété de la recette :

```
<?php
// Une façon efficace de stocker une recette !
$recette = [
   'titre' => 'Cassoulet',
   'recette' => 'Etape 1 : des flageolets, Etape 2 : ...',
   'auteur' => 'mickael.andrieu@exemple.com',
   'estActive' => true
];
?>
```

Il n'y a ici qu'une seule instruction (un seul point-virgule). J'aurais pu tout écrire sur la même ligne, mais séparer les couples clé/valeur sur plusieurs lignes est plus lisible.

Vous remarquez qu'on écrit une flèche (=>) pour dire « associé à ». Par exemple, on dit que la propriété `'titre'` du tableau `$recette` est associée à la valeur `'Cassoulet'`.

Nous avons créé un tableau qui ressemble à la structure suivante :

| Clé | Valeur |
|---|---|
| titre | 'Cassoulet' |
| recette | 'Etape 1 : des flageolets, Etape 2 : ...' |
| auteur | 'mickael.andrieu@exemple.com' |
| estActive | true |

Il est aussi possible de créer le tableau case par case :

```
<?php
$recette['titre'] = 'Cassoulet';
$recette['recette'] = 'Etape 1 : des flageolets, Etape 2 : ...';
$recette['auteur'] = 'mickael.andrieu@exemple.com';
$recette['estActive'] = true;
?>
```

### Afficher un tableau associatif

Pour afficher un élément, il suffit d'en indiquer le nom entre crochets et entre guillemets simples ou doubles puisque l'étiquette du tableau associatif est un texte. Par exemple, pour extraire le titre de la recette, on devra écrire :

```
<?php
echo $recette['titre'];  // Affiche 'Cassoulet'
?>
```

Comment choisir entre un tableau numéroté et un tableau associatif ?

Comme vous l'avez vu dans les exemples, ces deux types ne servent pas à stocker la même chose.

- Les tableaux numérotés servent à stocker une série d'éléments du même type, comme des recettes. Chaque élément du tableau contiendra alors une recette.
- Les tableaux associatifs permettent de découper une donnée en plusieurs sous-éléments. Par exemple, une recette peut être découpée en titre, descriptif, auteur, statut…

Dans le chapitre précédent, nous avons pourtant utilisé un tableau numéroté. N'aurait-il pas été mieux de choisir un tableau associatif ?

Oui, tout à fait, mais il fallait présenter rapidement les tableaux pour expliquer les boucles. D'ailleurs, nous allons maintenant présenter un autre type de boucle (comme ça, la boucle est bouclée ;) ).

## Parcourir un tableau

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

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

### La boucle for

Il est très simple de parcourir un tableau numéroté avec une boucle `for`. Nous l'avions abordé rapidement dans le chapitre précédent avec un tableau de tableaux :

```
<?php
/**
 * Déclaration du tableau des recettes
 * Chaque élément du tableau est un tableau numéroté (une recette)
 */
$recettes = [
   ['Cassoulet','[...]','mickael.andrieu@exemple.com',true,],
   ['Couscous','[...]','mickael.andrieu@exemple.com',false,],
];
```

```
for ($ligne = 0; $ligne <= 1; $ligne++) {
     echo $recettes[$ligne][0];
}
?>
```

Quand on écrit `$recettes[$ligne]`, la variable `$ligne` est d'abord remplacée par sa valeur. Par exemple, si `$ligne` vaut `1`, alors cela signifie qu'on cherche à obtenir ce que contient `$recettes[1][0]`, c'est-à-dire… `'Couscous'`. Bravo, vous avez compris !

## La boucle foreach

Même si `for` fonctionne, un autre type de boucle est plus adapté aux tableaux : `foreach`.

`foreach` passe en revue chaque ligne du tableau. Lors de chaque passage, elle met la valeur de cette ligne dans une variable temporaire (par exemple, `$element` ou, dans l'exemple qui suit, `$recette`).

```
<?php
// Déclaration du tableau des recettes
$recettes = [
   ['Cassoulet','[...]','mickael.andrieu@exemple.com',true,],
   ['Couscous','[...]','mickael.andrieu@exemple.com',false,],
];

foreach ($recettes as $recette) {
     echo $recette[0]; // Affichera 'Cassoulet', puis 'Couscous'
}
?>
```

C'est le même code que précédemment, mais basé cette fois sur `foreach`. À chaque tour de boucle, la valeur de l'élément suivant est stockée dans la variable `$recette`, qui n'est utilisable qu'à l'intérieur de la boucle.

La boucle `foreach` permet aussi de parcourir les tableaux associatifs :

```
<?php
$recette = [
   'titre' => 'Cassoulet',
   'recette' => 'Etape 1 : des flageolets, Etape 2 : ...',
   'auteur' => 'mickael.andrieu@exemple.com',
   'estActive' => true,
];

foreach ($recette as $valeur) {
   echo $valeur;
}
```

```
/**
 * Affiche
 * 'Cassoulet''Etape 1 : des flageolets, Etape 2 : ...''mickael.andrieu@
exemple.com'true
 */
?>
```

`foreach` va mettre tour à tour dans la variable `$valeur` le titre, le descriptif, l'auteur et le statut contenus dans le tableau `$recette`.

On met donc entre parenthèses d'abord le nom du tableau (ici, `$recette`), puis le mot-clé `as` (qui signifie quelque chose comme « en tant que ») et enfin le nom d'une variable que vous choisissez et qui va contenir tour à tour chacun des éléments (ici, `$valeur`). Entre les accolades, on n'utilise donc que la variable `$valeur`. La boucle s'arrête lorsque l'on a parcouru tous les éléments du tableau.

L'intérêt est encore plus flagrant avec un tableau de tableaux :

```
<?php

$recipes = [
   [
     'titre' => 'Cassoulet',
     'recette' => '',
     'auteur' => 'mickael.andrieu@exemple.com',
     'estActive' => true,
   ],
   [
     'titre' => 'Couscous',
     'recette' => '',
     'auteur' => 'mickael.andrieu@exemple.com',
     'estActive' => false,
   ],
   [
     'titre' => 'Escalope milanaise',
     'recette' => '',
     'auteur' => 'mathieu.nebra@exemple.com',
     'estActive' => true,
   ],
   [
     'titre' => 'Salade romaine',
     'recette' => '',
     'auteur' => 'laurene.castor@exemple.com',
     'estActive' => false,
   ],
];

foreach($recettes as $recette) {
   echo $recette['titre'] . ' proposé(e) par : ' . $recette['auteur'] .
PHP_EOL;
}
?>
```

Ce code produit le résultat suivant :

```
$ php exemple.php
Cassoulet proposé(e) par : mickael.andrieu@exemple.com
Couscous proposé(e) par : mickael.andrieu@exemple.com
Escalope milanaise proposé(e) par : mathieu.nebra@exemple.com
Salade romaine proposé(e) par : laurene.castor@exemple.com
```

Nous n'en avions pas parlé jusque-là puisque, dans ce livre, nous nous concentrons sur la réalisation d'un site web, mais PHP est capable d'exécuter des scripts en ligne de commande, un peu sur le modèle du démarrage du serveur web interne. C'est pratique quand vous souhaitez tester rapidement les exemples du livre.

Toutefois, avec cet exemple, on ne récupère que la valeur. Or, **on peut aussi récupérer la clé de l'élément**. Dans ce cas, on doit écrire `foreach` comme suit :

```
<?php foreach($recette as $cle => $valeur) ?>
```

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

- `$cle`, qui contiendra la clé de l'élément en cours d'analyse (ex. `'titre'`, `'auteur'`) ;
- `$valeur`, qui contiendra la valeur de l'élément en cours (ex. `'Cassoulet'`, `'mickael.andrieu@exemple.com'`).

Testons le fonctionnement avec un exemple :

```
<?php
$recette = [
   'titre' => 'Salade romaine',
   'recette' => 'Etape 1 : Lavez la salade ; Etape 2 : euh...',
   'auteur' => 'laurene.castor@exemple.com',
];

foreach($recette as $cle => $valeur)
{
   echo '[' . $cle . '] vaut ' . $valeur . PHP_EOL;
}
?>
```

Cela donne le résultat suivant :

```
$ php exemple.php
[titre] vaut Salade romaine
[recette] vaut Etape 1 : Lavez la salade ; Etape 2 : euh ...
[auteur] vaut laurene.castor@exemple.com
```

Grâce à cette façon de procéder, la boucle contient maintenant la clé ET la valeur. Croyez-moi, `foreach` est une boucle vraiment pratique. On s'en sert régulièrement.

## Afficher rapidement un tableau avec print_r

Parfois, en codant votre site en PHP, vous voudrez savoir ce que contient un tableau, juste pour information. Pour cela, vous pourriez utiliser une boucle `for` ou, mieux, une boucle `foreach`. Cependant, si vous n'avez pas besoin d'une mise en forme spéciale, la fonction `print_r` vous sera utile. C'est une sorte de `echo` spécialisée dans les tableaux.

Cette commande a toutefois un défaut : elle ne renvoie pas de code HTML comme `<br />` pour les retours à la ligne. Pour bien les voir, il faut donc utiliser la balise HTML `<pre>` qui donne un affichage plus correct.

```
<?php
$recettes = [
   [
      'titre' => 'Cassoulet',
      'recette' => '',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estActive' => true,
   ],
   [
      'titre' => 'Couscous',
      'recette' => '',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estActive' => false,
   ],
];

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

Cela donne le résultat suivant :

```
Array
(
   [0] => Array
          (
             [titre] => Cassoulet
             [recette] =>
             [auteur] => 'mickael.andrieu@exemple.com'
             [estActive] => true
          )
   [1] => Array
          (
             [titre] => Couscous
             [recette] =>
             [auteur] => 'mickael.andrieu@exemple.com'
             [estActive] => false
          )
)
```

Voilà, c'est facile à utiliser du moment que l'on n'oublie pas la balise `<pre>` pour obtenir un affichage correct.

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

## Rechercher dans un tableau

Nous allons maintenant lancer des recherches dans des tableaux. Elles sont de trois types, basés sur des fonctions PHP :

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

### Vérifier si une clé existe dans le tableau : array_key_exists

Voici notre problème : on a un tableau, mais on ne sait pas si la clé que l'on recherche s'y trouve.

Pour vérifier cela, on va utiliser la fonction `array_key_exists` en lui donnant d'abord le nom de la clé à rechercher, puis le nom du tableau :

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

La fonction renvoie un booléen, c'est-à-dire `true` (vrai) si la clé est dans le tableau et `false` (faux) sinon. Cela s'insère facilement dans un test avec un `if` :

```
<?php
$recette = [
   'titre' => 'Salade romaine',
   'recette' => 'Etape 1 : Lavez la salade ; Etape 2 : euh ...',
   'auteur' => 'laurene.castor@exemple.com',
];

if (array_key_exists('titre', $recette)) {
   echo 'La clé "titre" se trouve dans la recette !';
}

if (array_key_exists('commentaires', $recette)) {
   echo 'La clé "commentaires" se trouve dans la recette !';
}
?>
```

Seule la clé `titre` est trouvée et affichée :

```
$ php exemple.php
La clé "titre" se trouve dans la recette !
```

## Vérifier si une valeur existe dans le tableau : in_array

Le principe est le même mais, cette fois, on recherche dans les valeurs. `in_array` renvoie `true` si la valeur se trouve dans le tableau, `false` sinon.

Pour changer un peu de notre tableau `$recettes`, je vais en créer un autre qui contient le nom des utilisateurs du site.

```
<?php
$utilisateurs = [
   'Mathieu Nebra',
   'Mickaël Andrieu',
   'Laurène Castor',
];

if (in_array('Mathieu Nebra', $utilisateurs)) {
   echo 'Mathieu fait bien partie des utilisateurs enregistrés !';
}

if (in_array('Arlette Chabot', $utilisateurs)) {
   echo 'Arlette fait bien partie des utilisateurs enregistrés !';
}
?>
```

On ne voit que le message pour Mathieu, car Arlette n'est pas enregistrée parmi les utilisateurs :

```
$ php exemple.php
Mathieu fait bien partie des utilisateurs enregistrés !
```

## Récupérer la clé d'une valeur dans un tableau : array_search

Comme `in_array`, `array_search` travaille sur les valeurs d'un tableau.

- Si la valeur cherchée est trouvée, `array_search` renvoie la clé (ou le numéro) correspondante.
- Sinon, `array_search` renvoie `false`.

Reprenons le tableau des utilisateurs :

```
<?php
$utilisateurs = [
   'Mathieu Nebra',
   'Mickaël Andrieu',
   'Laurène Castor',
];

$positionMathieu = array_search('Mathieu Nebra', $utilisateurs);
echo '"Mathieu" se trouve en position ' . $positionMathieu . PHP_EOL;

$positionLaurène = array_search('Laurène Castor', $utilisateurs);
echo '"Laurène" se trouve en position ' . $positionLaurène . PHP_EOL;
?>
```

```
$ php exemple.php
"Mathieu" se trouve en position 0
"Laurène" se trouve en position 2
```

Je sais que je me répète, mais n'oubliez pas qu'un tableau numéroté commence à l'indice 0 !

Il existe d'autres fonctions pour effectuer une recherche dans un tableau, mais vous avez là les principales à connaître.

## Afficher des recettes (version 2)

Avec tout ce que nous avons appris dans ce chapitre, nous pouvons améliorer le code d'affichage des recettes : nous allons passer les recettes en tableau associatif, contrôler la clé `estActive` et afficher les recettes seulement si cette clé vaut `true`.

```
<?php
$recettes = [
   [
      'titre' => 'Cassoulet',
      'recette' => 'Etape 1 : des flageolets !',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estActive' => true,
   ],
   [
      'titre' => 'Couscous',
      'recette' => 'Etape 1 : de la semoule',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estActive' => false,
   ],
   [
      'titre' => 'Escalope milanaise',
      'recette' => 'Etape 1 : prenez une belle escalope',
```

```
    'auteur' => 'mathieu.nebra@exemple.com',
    'estActive' => true,
  ],
];
?>


<!DOCTYPE html>
<html>
  <head>
    <title>Affichage des recettes</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/
bootstrap.min.css"
      rel="stylesheet"
    >
  </head>
  <body>
    <div class="container">
      <h1>Affichage des recettes</h1>
      <!-- Boucle sur les recettes -->
      <?php foreach($recettes as $recette) : ?>
        <!-- si la clé existe et a pour valeur "vrai", on affiche -->
        <?php if (array_key_exists('estValide', $recette)
                  && $recette['estValide'] == true): ?>
          <article>
            <h3><?php echo $recette['titre']; ?></h3>
            <div><?php echo $recette['recette']; ?></div>
            <i><?php echo $recette['auteur']; ?></i>
          </article>
        <?php endif; ?>
      <?php endforeach ?>
    </div>
  </body>
</html>
```


Affichage des recettes en fonction de leur statut

## En résumé

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

# 9 Les fonctions

Comme les boucles, les fonctions évitent de répéter du code PHP qu'on utilise souvent. Toutefois, alors que les boucles sont de bêtes machines tout juste capables de répéter deux cents fois la même chose, les fonctions sont des robots « intelligents » qui s'adaptent en fonction de ce que vous voulez faire et qui automatisent grandement la plupart des tâches courantes.

Nous profiterons de ces nouvelles connaissances pour améliorer et simplifier l'affichage des recettes en recoupant la liste des utilisateurs et celle des recettes.

## Qu'est-ce qu'une fonction ?

Une fonction est une série d'instructions qui effectuent des actions et qui retournent une valeur. En général, dès que vous avez besoin de réaliser 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 le fait pour vous. Et, si la fonction n'existe pas, vous avez la possibilité de la créer.

Imaginez que les fonctions sont des robots.

Une fonction est comme un robot.

Vous ne savez pas ce qui se passe à l'intérieur de ce robot, mais vous pouvez appuyer sur un bouton pour lui demander de faire quelque chose de précis. Avec les fonctions, c'est le même principe !

## Dialoguer avec une fonction

Voici le genre de dialogue qu'on peut avoir avec une fonction :

- « Toi, la fonction `RecetteOK`, dis-moi si j'ai le droit d'afficher cette recette. »

La fonction effectue les calculs demandés puis répond :

- « Oui tu peux ».

En **entrée**, on donne à la fonction un **paramètre** (ici, une recette sous forme de tableau) sur lequel elle va réaliser des opérations, avant de nous retourner en **sortie** le résultat (ici, un booléen).

```
<?php
$recette = [
   'titre' => 'Escalope milanaise',
   'recette' => '',
   'auteur' => 'mathieu.nebra@exemple.com',
   'estValide' => true,
];

recetteOK($recette); // retourne le booléen true
?>
```

Grâce à la fonction, vous n'avez plus besoin de vous rappeler comment on décide si une recette doit ou non être affichée. Bon, ici c'était assez simple (il suffisait de vérifier la valeur de la clé `estValide`), mais vous serez souvent amené à programmer des opérations plus complexes. Les fonctions sont pratiques quand on a plusieurs fois besoin de l'information et qu'il y a plusieurs conditions à respecter.

Imaginons par exemple que les conditions d'affichage des recettes évoluent : la clé `estValide` vaut `true`, l'utilisateur est connecté et administrateur, l'utilisateur est majeur... Nous n'allons pas recopier et implémenter toutes ces instructions à chaque fois ; nous utiliserons une fonction dont le rôle sera de nous retourner `true` ou `false`.

En quoi est-ce différent des boucles qui répètent aussi le même code plusieurs fois ?

Une fonction est capable de s'adapter selon les informations que vous lui envoyez (les paramètres). Par exemple dans notre cas, il suffit de lui transmettre le tableau `$recette`.

Concrètement, les fonctions sont capables de récupérer des informations comme la date et l'heure actuelles, de chiffrer des données, d'envoyer des courriels, de lancer des recherches dans du texte et bien d'autres choses encore !

## Les fonctions en PHP

Nous avons jusqu'ici imaginé un cas d'utilisation très simple que nous avions résolu précédemment avec une boucle et une condition, ce qui n'est pas très intéressant. Revenons aux choses sérieuses et concrètes : explorons les fonctions puis codons-en une pour afficher le nom de l'auteur plutôt que son adresse électronique lors de l'affichage d'une recette.

### *Appeler une fonction*

En PHP, comment appelle-t-on une fonction ? Par son nom, pardi ! Voici un exemple :

```
<?php
recetteOK();
?>
```

La fonction `recetteOK` est une fonction imaginaire : elle n'existe pas (à moins de la créer nous-même). Par conséquent, n'essayez pas d'exécuter ce code PHP chez vous, car il ne fonctionnera pas. Lisez simplement pour bien comprendre le fonctionnement, vous aurez l'occasion de pratiquer plus loin dans ce chapitre.

Comme vous le voyez, j'ai simplement écrit le nom de la fonction, suivi de parenthèses vides, puis de l'inévitable point-virgule. Ainsi, j'appelle la fonction `recetteOK`, mais je ne lui envoie aucune information, aucun paramètre.

Certaines fonctions peuvent agir sans paramètres, mais elles sont assez rares. Dans le cas de `recetteOK`, par exemple, cela n'a pas de sens de l'appeler sans lui donner la recette nécessaire pour décider !

Pour transmettre un paramètre (un nombre, une chaîne de caractères, un booléen…), il faut l'écrire entre les parenthèses :

```
<?php
/**
 * Il n'est pas nécessaire de déclarer une variable $recette
 * pour passer l'information en tant que paramètre d'une fonction.
 */
recetteOK([
   'titre' => 'Escalope milanaise',
   'recette' => '',
   'auteur' => 'mathieu.nebra@exemple.com',
   'estValide' => true,
]);
?>
```

Souvent, les fonctions acceptent plusieurs paramètres. Vous devez dans ce cas les séparer par des virgules :

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

Cette fonction recevra quatre paramètres : `17`, le texte `Vert`, le booléen `true` et le nombre `41,7`.

### *Récupérer la valeur de retour de la fonction*

Maintenant que nous savons appeler une fonction et même lui envoyer plusieurs paramètres, il faut récupérer ce qu'elle nous retourne, si toutefois elle retourne quelque chose. Il y a en effet deux types de fonctions :

- Certaines ne retournent aucune valeur (ça ne les empêche pas d'effectuer des actions). Il n'y a rien de plus à faire que dans les codes précédents : la fonction est appelée, fait son travail et on ne lui demande rien de plus.
- D'autres (comme `recetteOK`) retournent une valeur. Il nous faut récupérer cette dernière dans une variable, comme suit :

```
<?php
$OK = recetteOK([
   'titre' => 'Escalope milanaise',
   'recette' => '',
   'auteur' => 'mathieu.nebra@exemple.com',
   'estValide' => true,
]);

if ($OK) {
   echo 'La recette doit être affichée !';
} else {
   echo 'La recette doit être cachée !';
}
?>
```

Dans cet exemple, il se passe en fait deux choses :

1. La fonction `recetteOK` est appelée avec un tableau en paramètre.
2. Le résultat renvoyé par la fonction (lorsqu'elle a terminé) est stocké dans la variable `$OK`. Cette dernière aura donc pour valeur `true` après l'exécution de cette ligne de code.

Comme on l'a vu, il est possible d'envoyer en entrée plusieurs paramètres à une fonction ; en revanche, cette dernière ne peut retourner qu'une seule valeur. Il existe un moyen de contourner cette limitation en combinant des variables au sein d'un tableau.

## Les fonctions prêtes à l'emploi

PHP propose des centaines de fonctions prêtes à l'emploi. Sur le site officiel, la documentation du langage les répertorie toutes, classées par catégories (*http://fr.php.net/manual/fr/funcref.php*).

Ces fonctions sont très pratiques. En fait, c'est en partie sur ce point que réside la force de PHP : ses fonctions sont vraiment excellentes car elles couvrent la quasi-totalité de nos besoins. À chaque fois, ou presque, que je m'apprêtais à écrire une fonction, j'ai constaté qu'elle existait déjà.

Voici un petit aperçu des fonctions proposées, pour vous mettre l'eau à la bouche :

- `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 *(thumbnails)* ;
- `mail` pour envoyer un e-mail avec PHP (très pratique pour créer une **newsletter**) ;
- de nombreuses options pour modifier des images, y écrire du texte, tracer des lignes, des rectangles ;
- `crypt` pour chiffrer des mots de passe ;
- `date` pour renvoyer l'heure, la date…
- etc.

Dans la plupart des cas, il faudra passer des paramètres à la fonction pour qu'elle sache sur quoi travailler.

Nous allons ici découvrir rapidement quelques fonctions pour vous habituer à les utiliser. Nous ne pourrons jamais toutes les passer en revue mais, avec l'expérience de ces premières fonctions et la documentation de PHP, vous n'aurez aucun mal à aller plus loin par vous-même. Nous allons voir trois fonctions qui modifient des chaînes de caractères et une qui récupère la date.

### Traiter des chaînes de caractères

De nombreuses fonctions servent à manipuler le texte. En voici trois couramment utilisées.

#### *Calculer la longueur d'une chaîne : strlen*

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

```
<?php
$recette = 'Etape 1 : des flageolets ! Etape 2 : de la saucisse toulousaine';
$longueur = strlen($recette);

echo 'La phrase ci-dessous comporte ' . $longueur . ' caractères :' . PHP_EOL . $recette;
?>
```

```
$ php exemple.php
La phrase ci-dessous comporte 63 caractères :
Etape 1 : des flageolets ! Etape 2 : de la saucisse toulousaine
```

Dans le même ordre d'idée, la fonction `count` compte le nombre d'éléments dans un tableau car, en PHP, une chaîne de caractères est… un tableau de caractères !

### *Rechercher et remplacer une chaîne : str_replace*

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

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

```
$ php exemple.php
le Cassoulet, C'est très bon
```

On a besoin d'indiquer trois paramètres :

- la chaîne qu'on recherche (ici, les `'c'`, mais on aurait pu rechercher un mot ou plus) ;
- la chaîne qu'on veut mettre à la place (ici, on remplace les `'c'` par des `'C'`) ;
- la chaîne dans laquelle on doit mener la recherche.

### *Formater une chaîne : sprintf*

Cette fonction de formatage est très pratique lorsque nous avons besoin de passer plusieurs variables et peut remplacer la concaténation pour améliorer la lisibilité du code. Elle est très puissante ; n'hésitez pas à consulter la documentation officielle.

```
<?php
$recette = [
   'titre' => 'Salade romaine',
   'recette' => 'Etape 1 : Lavez la salade ; Etape 2 : euh...',
   'auteur' => 'laurene.castor@exemple.com',
];

echo sprintf(
   '%s par "%s" : %s',
   $recette['titre'],
   $recette['auteur'],
   $recette['recette']
);
?>
```

```
$ php exemple.php
Salade romaine par "laurene.castor@exemple.com" : Etape 1 : Lavez la salade
; Etape 2 : euh...
```

## Récupérer la date et l'heure

Cette fonction peut donner beaucoup d'informations. Voici les principaux paramètres à connaître :

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

Lors de votre saisie, respectez la casse : les majuscules et les minuscules sont importantes !

Si vous voulez afficher l'année, il faut donc envoyer le paramètre `Y` à la fonction :

```
<?php
$annee = date('Y');
echo $annee;
?>
```

On peut bien entendu faire mieux ; voici la date complète et l'heure :

```
<?php
// Enregistrons les informations de date dans des variables

$jour = date('d');
$mois = date('m');
$annee = date('Y');

$heure = date('H');
$minute = date('i');

// Maintenant, on peut afficher ce qu'on a recueilli
echo 'Bonjour ! Nous sommes le ' . $jour . '/' . $mois . '/' . $annee .
'et il est ' . $heure. ' h ' . $minute;
?>
```

Et voilà le travail ! On a pu afficher la date et l'heure en un clin d'œil.

Si l'heure n'était pas bonne, sachez que c'est le serveur qui la donne. Selon l'endroit où vous vous trouvez, le fuseau horaire du serveur peut ne pas correspondre au vôtre.

## Créer ses propres fonctions

Bien que PHP propose des centaines de fonctions, parfois il n'y aura pas ce que vous cherchez et il faudra écrire vous-même la fonction. C'est une façon pratique d'étendre les possibilités offertes par PHP.

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 :

- vérifier si la recette est valide ;
- récupérer des recettes à afficher ;
- récupérer le nom d'un utilisateur en fonction de l'adresse électronique associée à une recette.

### Vérifier la validité d'une recette

Cette fonction très simple retourne `true` si la recette est valide et `false` sinon. Précédemment, nous avions employé une condition `if` pour vérifier la propriété `estValide` de la recette.

```
<?php
$recette = [
   'titre' => 'Salade romaine',
   'recette' => 'Etape 1 : Lavez la salade ; Etape 2 : euh...',
   'auteur' => 'laurene.castor@exemple.com',
   'estValide' => true,
];

// au minimum
if ($recette['estValide']) {
   return true;
} else {
   return false;
}

// mieux
$OK = $recette['estValide'];

// encore mieux !
if (array_key_exists('estValide', $recette)) {
   $OK = $recette['estValide'];
} else {
   $OK = false;
}
?>
```

Nous n'allons tout de même pas recopier tout ce code pour chaque recette à vérifier ! Voici le code de la fonction correspondante :

```
<?php
function recetteOK(array $recette) : bool {
   if (array_key_exists('estValide', $recette)) {
      $OK = $recette['estValide'];
   } else {
      $OK = false;
   }

   return $OK;
}
?>
```

Pour créer une fonction, vous devez taper le mot-clé `function`, puis lui donner un nom (celle de notre exemple s'appelle `recetteOK()`).

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, une recette). Nous pouvons (et c'est une bonne pratique) définir le type de la variable attendue (ici, `array`). Notre fonction doit forcément être appelée avec un paramètre (une recette) sans quoi elle ne pourra pas travailler.

Vous avez peut-être remarqué que cette ligne est la seule à ne pas se terminer par un point-virgule. C'est normal, il ne s'agit pas d'une instruction mais juste d'une « carte d'identité » de la fonction (son nom, ses paramètres).

Notre fonction peut aussi – et c'est une autre bonne pratique – préciser le type de la valeur qu'elle retourne (ici, un booléen). Ensuite, vous repérez des accolades ; elles délimitent son contenu.

Voilà, la fonction est créée, vous n'avez plus besoin d'y toucher. Pour l'appeler par la suite, il suffit d'indiquer son nom et de préciser ses paramètres entre parenthèses. Enfin, il ne faut pas oublier le fameux point-virgule (`;`) car il s'agit d'une instruction.

```
<?php
// 2 exemples
$saladeRomaine = [
   'titre' => 'Salade romaine',
   'recette' => 'Etape 1 : Lavez la salade ; Etape 2 : euh...',
   'auteur' => 'laurene.castor@exemple.com',
   'estValide' => true,
];

$sushis = [
   'titre' => 'Sushis',
   'recette' => 'Etape 1 : du saumon ; Etape 2 : du riz',
   'auteur' => 'laurene.castor@exemple.com',
   'estValide' => false,
];


// Répond true
$saladeRomaineOK = recetteOK($saladeRomaine);
```

```
// Répond false
$sushisOK = recetteOK($sushis);
?>
```

Un conseil pour vous entraîner sur les fonctions : basez-vous sur mes exemples et essayez de les retoucher petit à petit pour voir ce que cela donne. Certaines fonctions sont très simples, d'autres plus compliquées, alors allez-y prudemment.

## Récupérer les recettes valides

Nous voulons afficher la liste des recettes valides. Or, nous disposons maintenant d'une fonction qui vérifie la validité d'une recette. Il nous reste à boucler sur l'ensemble.

```
<?php
$recettes = [...]; // Les recettes

// AVANT
foreach ($recettes as $recette) {
  if ($recette['estValide']) {
    // echo $recette['titre'] ...
  }
}

// APRES
function trouverRecettes(array $recettes) : array {
  $recettesValides = [];

  foreach($recettes as $recette) {
    if (recetteOK($recette)) {
      $recettesValides[] = $recette;
    }
  }

  return $recettesValides;
}

// construire l'affichage HTML des recettes
foreach(trouverRecettes($recettes) as $recette) {
  // echo $recette['titre'] ...
}
?>
```

Comme précédemment, on boucle et on ne conserve que les recettes valides identifiées grâce à la fonction `recetteOK()`.

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

Afficher le nom de l'auteur

Allez, on passe à la vitesse supérieure. Nous allons créer une fonction pour améliorer l'affichage, ce qui nous donne à nouveau l'occasion de manipuler des tableaux.

Cette fois, la problématique est de relier l'adresse associée à un compte utilisateur à celle utilisée pour la création d'une recette.

Si on découpe le problème en étapes, vous avez déjà toutes les connaissances nécessaires :

1. Boucler sur les recettes valides.
2. Prendre l'adresse e-mail.
3. Boucler sur les utilisateurs de la plate-forme.
4. Si les adresses correspondent, prendre le nom.
5. Sinon, continuer à parcourir la liste des utilisateurs.

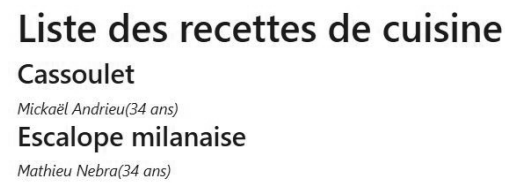
Une solution tout à fait valide serait donc la suivante (avec une boucle dans une boucle) :

```
<?php
$utilisateurs = [
   [
      'nom' => 'Mickaël Andrieu',
      'courriel' => 'mickael.andrieu@exemple.com',
      'age' => 34,
   ],
   [
      'nom' => 'Mathieu Nebra',
      'courriel' => 'mathieu.nebra@exemple.com',
      'age' => 34,
   ],
   [
      'nom' => 'Laurène Castor',
      'courriel' => 'laurene.castor@exemple.com',
      'age' => 28,
   ],
];

$recettes = [
   [
      'titre' => 'Cassoulet',
      'recette' => '',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estValide' => true,
   ],
   [
      'titre' => 'Couscous',
      'recette' => '',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estValide' => false,
   ],
   [
      'titre' => 'Escalope milanaise',
      'recette' => '',
      'auteur' => 'mathieu.nebra@exemple.com',
```

```
    'estValide' => true,
  ],
  [
    'titre' => 'Salade romaine',
    'recette' => '',
    'auteur' => 'laurene.castor@exemple.com',
    'estValide' => false,
  ],
];

function trouverAuteur(string $courrielAuteur, array $utilisateurs) :
string {
  for ($i = 0; $i < count($utilisateurs); $i++) {
    $auteur = $utilisateurs[$i];
    if ($courrielAuteur === $auteur['courriel']) {
      return $auteur['nom'] . '(' . $auteur['age'] . ' ans)';
    }
  }
}

function recetteOK(array $recette) : bool {
  if (array_key_exists('estValide', $recette)) {
    $OK = $recette['estValide'];
  } else {
    $OK = false;
  }

  return $OK;
}

function trouverRecettes(array $recettes) : array
{
  $recettesValides = [];

  foreach($recettes as $recette) {
    if (recetteOK($recette)) {
      $recettesValides[] = $recette;
    }
  }

  return $recettesValides;
}
?>


<!DOCTYPE html>
<html lang="en">
<head>
  <title>Recettes de cuisine</title>
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.
min.css"
    rel="stylesheet"
  >
</head>
<body>
  <div class="container">
    <h1>Liste des recettes de cuisine</h1>
```

```
    <?php foreach(trouverRecettes($recettes) as $recette) : ?>
      <article>
        <h3><?php echo $recette['titre']; ?></h3>
        <div><?php echo $recette['recette']; ?></div>
        <i><?php echo trouverAuteur($recette['auteur'], $utilisateurs);
?></i>
      </article>
    <?php endforeach ?>
  </div>
</body>
</html>
```

En voici le résultat :




L'affichage du nom des utilisateurs, c'est fait !

## En résumé

- Les fonctions sont des blocs de code qui exécutent des instructions selon certains paramètres.
- Les fonctions ont généralement une entrée et une sortie. Elles peuvent également être typées : `string`, `int`, `bool`, `array`...
- PHP propose des centaines de fonctions prêtes à l'emploi pour tout type de tâches : envoyer un e-mail, récupérer l'heure, chiffrer des mots de passe...
- Si PHP ne propose pas la fonction dont on a besoin, il est possible de la créer avec le mot-clé `function`. On définira alors le type des paramètres et celui de la valeur retournée.

# 10 Au secours ! Mon script plante !

Alors comme ça votre script ne fonctionne pas et PHP vous affiche des erreurs incompréhensibles ?

Vous n'avez pas de souci à vous faire, c'est tout à fait normal. On ne réussit jamais un script du premier coup (en tout cas, moi, jamais).

Des milliers de messages d'erreur différents peuvent survenir (OK, jusque-là rien de très rassurant) et je n'ai pas vraiment la possibilité de vous en dresser une liste complète… mais je vais vous présenter les erreurs les plus courantes, ce qui devrait résoudre la grande majorité de vos problèmes. ;-)

## Les erreurs les plus courantes

Il est facile de parler d'erreurs « courantes », car vous verrez que certaines reviennent plus souvent que d'autres :

- `parse error` ;
- `undefined function` ;
- `wrong parameter count`.

### Parse error

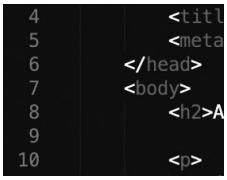
Si on devait dire qu'il existe UNE erreur de base, ce serait très certainement celle-ci. Il est impossible de programmer en PHP sans y avoir droit un jour.

Le message d'erreur que vous obtenez ressemble à celui-ci :

```
Parse error: parse error in fichier.php on line 15
```

Ce message vous indique une erreur dans `fichier.php` à la ligne 15. Généralement, cela veut dire que votre problème se situe à la ligne 15, mais ce n'est pas toujours le cas (ce serait trop facile, sinon). Parfois c'est la précédente qui a un problème : pensez donc à regarder autour de la ligne indiquée.

Avec un éditeur de texte spécialisé comme Visual Studio Code, les numéros de ligne sont indiqués.




Numérotation des lignes dans Visual Studio Code

Concrètement, qu'est-ce qu'une `parse error` ? Il s'agit en fait d'une instruction PHP mal formulée. Il peut y avoir plusieurs causes à cela.

- Vous avez oublié le **point-virgule à la fin** de l'instruction. Par exemple :

```
$valeur = 5
```

  Si vous ajoutez le point-virgule à la fin, tout rentrera dans l'ordre !

```
$valeur = 5;
```

- Vous avez oublié de **fermer des guillemets** (ou une parenthèse). Par exemple :

```
echo "Bonjour !;
```

  Il suffit de fermer correctement les guillemets (ou la parenthèse) pour supprimer le problème :

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

- Vous vous êtes trompé dans la **concaténation** et avez peut-être oublié un point par exemple :

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

  Une fois l'erreur corrigée, cela donne :

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

- Il peut aussi s'agir d'une **accolade mal fermée** (pour un `if`, par exemple). Vérifiez que vous avez correctement fermé toutes vos accolades. Si vous oubliez d'en fermer une, il est probable que la `parse error` vous indique que l'erreur se trouve à la dernière ligne du fichier.

## undefined function

La fonction inconnue est une autre erreur assez classique. Vous obtenez un message similaire au suivant :

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

Là, il faut comprendre que vous avez utilisé une fonction qui n'existe pas. Deux possibilités s'offrent à vous :

- La **fonction n'existe vraiment pas**. Vous avez probablement fait une faute de frappe ; vérifiez si une fonction à l'orthographe similaire existe.
- La fonction existe, 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 GD 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.

Une dernière chose : il se peut aussi que vous essayiez d'utiliser une fonction qui n'est pas disponible dans la version de PHP que vous utilisez. Vérifiez ce point dans le manuel.

## wrong parameter count

Si vous utilisez mal une fonction, vous obtiendrez une erreur comme la suivante :

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

Cela signifie que vous avez oublié des paramètres pour la fonction, ou bien que vous en avez trop mis. Consultez le mode d'emploi de la fonction pour savoir combien de paramètres elle prend et quels sont ceux qui sont facultatifs.

Par exemple, la fonction `fopen()` requiert au minimum deux paramètres : le nom du fichier à ouvrir et le mode d'ouverture. Si vous ne précisez que le premier :

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

… vous obtiendrez l'erreur `Wrong parameter count`. Pensez donc à ajouter le paramètre qui manque :

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

## Quelques erreurs plus rares

Les erreurs PHP sont très variées. N'espérez donc pas que je vous dresse ici la liste des 3 946 erreurs possibles.

Je vais vous présenter ici quelques erreurs un peu plus rares que `parse error`, mais que vous rencontrerez probablement un jour :

- `headers already sent by…` ;
- `l'image contient des erreurs` ;
- `maximum execution time exceeded`.

### headers already sent by…

Voici une erreur classique quand on travaille avec les sessions ou avec les *cookies* :

```
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 disent « Ce que tu vas recevoir est une page HTML », « Ce que tu vas recevoir est une image PNG » ou encore « Inscris un cookie ».

Tout cela doit être effectué avant que le moindre code HTML ne soit envoyé. En PHP, la fonction qui envoie des informations d'en-tête s'appelle `header()`. D'autres en envoient également toutes seules, par exemple `session_start()` et `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 cela provoquera l'erreur `Headers already sent by…`.

Voici un exemple de code qui provoque l'erreur :

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

Ici, j'ai eu le malheur de placer un peu de code HTML avant `session_start()`, ce qui a provoqué l'erreur. Inversez les deux instructions et vous n'aurez plus de problème :

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

## L'image contient des erreurs (utilisation de la bibliothèque GD)

C'est le navigateur qui vous envoie ce message d'erreur et non pas PHP. Cela survient lorsque **vous travaillez avec la bibliothèque GD**. Si vous avez commis une erreur dans votre code (par exemple, une banale `parse error`), elle sera **inscrite dans l'image**. Ainsi, cette dernière ne sera pas valide et le navigateur ne pourra pas l'afficher.

Comment faire « apparaître » l'erreur ?

Deux possibilités s'offrent à vous :

- supprimer la ligne suivante dans votre code (l'erreur apparaîtra à la place du message `L'image contient des erreurs`) :

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

- 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 ; il ne vous restera plus qu'à corriger le bogue !

## maximum execution time exceeded

Cette erreur est le plus souvent provoquée par une boucle infinie :

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

Imaginez que vous programmiez une boucle `while` sans prévoir la condition d'arrêt : votre script PHP va tourner en boucle sans jamais s'arrêter.

Heureusement, PHP limite le temps d'exécution d'une page PHP à 30 secondes par défaut (en général, un serveur met moins de 50 millisecondes à charger une page PHP). Si une page prend plus de temps pour se générer, PHP arrête tout en signalant que c'est trop long, parce que sinon cela pourrait ralentir tout le serveur et rendre votre site inaccessible.

Voici un exemple de boucle `while` qui ne s'arrêtera jamais :

```
<?php
$compteur = 5;
while ($compteur == 5)
{
   echo 'Zéro ';
}
?>
```

Un tel code PHP ne s'arrêtera jamais parce que `$compteur` vaut TOUJOURS `5`…

## Déboguer un premier script

Rien ne vaut la pratique pour mettre en application ce que vous venez d'apprendre. Je vous propose donc de copier le code suivant et de le corriger :

```
<?php
$utilisateurs = [
   [
      'nom' => 'Mickaël Andrieu',
      'courriel' => 'mickael.andrieu@exemple.com'
      'age' => 34,
   ],
   [
      'nom' => 'Mathieu Nebra',
      'courriel' => 'mathieu.nebra@exemple.com',
      'age' => 34,
   ],
   [
      'nom' => 'Laurène Castor',
      'courriel' => 'laurene.castor@exemple.com',
      'age' => 28,
   ],
];

$recettes = [
   [
      'titre' => 'Cassoulet',
      'recette' => '',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estValide' => true,
   ],
   [
      'titre' => 'Couscous',
      'recette' => '',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estValide' => false,
   ],
   [
      'titre' => 'Escalope milanaise',
      'recette' => '',
      'auteur' => ' mathieu.nebra@exemple.com',
      'estValide' => true,
   ],
   [
      'titre' => 'Salade Romaine',
      'recette' => '',
      'auteur' => 'laurene.castor@exemple.com',
      'estValude' => false,
   ],
];
```

```
function afficherRecette(array $recette) : string
{
  $contenuRecette = '';
  if ($recette['estValide']) {
    $contenuRecette = '<article>';
    $contenuRecette .= '<h3>' . $recette['titre'] . '</h3>';
    $contenuRecette .= '<div>' . $recette['recette'] . '</div>';
    $contenuRecette .= '<i>' . $recette['auteur'] . '</i>';
    $contenuRecette .= '</article>';
  }
  return $recette;
}


function afficherAuteur(string $courrielAuteur, array $utilisateurs) :
string
{
  for ($i = 0; $i < count($utilisateurs); $i++) {
    $auteur = $utilisateurs[$i];
    if ($courrielAuteur === $auteur['courriel']) {
      return $auteur['nom'] . '(' . $auteur['age'] . ' ans)';
    }
  }
}


function trouverRecettes(array $recettes) : array
{
  $recettesValides = [];
  foreach($recettes as $recette) {
    if ($recette['estValide']) {
      $recettesValides[] = $recette;
    }
  }
  return $recettesValides;
}
?>



<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Les recettes mais page blanche :(</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/
bootstrap.min.css"
      rel="stylesheet"
    >
  </head>
  <body>
    <div class="container">
      <h1>Liste des recettes</h1>
      <!-- Plus facile à lire -->
      <?php foreach(trouverRecettes($recettes) as $recette) : ?>
        <article>
          <h3><?php echo($recette['titre']); ?></h3>
          <div><?php echo($recette['recette']); ?></div>
```

```
            <i><?php echo(afficherAuteur($recette['auteur'],
$utilisateurs)); ?></i>
          </article>
       <?php endforeach ?>
     </div>
   </body>
</html>
```

Pour rappel :

- Soit vous placez ce fichier dans un dossier accessible par votre serveur, soit vous exécutez le serveur local de PHP dans le dossier qui contient ce fichier.
- Il faut avoir activé le débogage et configuré la propriété `display_error`.
- Il y a plusieurs erreurs à corriger : procédez par étape.

Alors, avez-vous commis des erreurs en corrigeant des erreurs ?

La première erreur affichée par PHP est la suivante :

```
Parse error: syntax error, unexpected ''age'' (T_CONSTANT_ENCAPSED_STRING),
expecting ']' in exemple.php on line 6
```

Il y a donc une erreur à la ligne 6, ou autour... vous aurez peut être remarqué qu'il manque une virgule en ligne 5 juste après l'affectation de la propriété `courriel` :

```
<?php
// avant
[
...
   'courriel' => 'mickael.andrieu@exemple.com' // OUPS
   'age' => 34,
]


// après
[
...
   'courriel' => 'mickael.andrieu@exemple.com',
   'age' => 34,
]
?>
```

La deuxième erreur est plus discrète puisque le site semble s'afficher correctement sous vos yeux. Pourtant, il y a un message sous le titre principal :

```
Notice: Undefined index: estValide in exemple.php on line 88
```

La clé pour le tableau `$recette` ne semble pas définie (l'erreur n'aurait pas été levée si on avait utilisé la fonction `array_key_exists`). Il faut vérifier le tableau des recettes

et compléter ou corriger. Il y a une toute petite coquille : `estVali**d**e` avait été remplacé par `estVal**u**de`.

```
<?php
// avant
[
  'titre' => 'Salade romaine',
  'recette' => '',
  'auteur' => 'laurene.castor@exemple.com',
  'estVal**u**de' => false, // OUPS
]

// après
[
  'titre' => 'Salade romaine',
  'recette' => '',
  'auteur' => 'laurene.castor@exemple.com',
  'estVali**d**e' => false,
]
?>
```

La troisième et dernière erreur est beaucoup plus bavarde :

```
**Fatal error**: Uncaught TypeError: Return value of afficherAuteur() must be
of the type string, none returned in exemple.php:71
Stack trace: #0 exemple.php(107): afficherAuteur() #1 {main} thrown in
exemple.**php** on line **81**
```

Cette fois, l'erreur provient de la fonction `afficherAuteur()`, qui doit retourner une chaîne de caractères (le nom de l'utilisateur) et qui ne retourne... rien (`None`).

Que s'est-il passé ?

```
<?php
function afficherAuteur(string $courrielAuteur, array $utilisateurs) : string
{
  for ($i = 0; $i < count($utilisateurs); $i++) {
    $auteur = $utilisateurs[$i];
    if ($courrielAuteur === $auteur['courriel']) {
      return $auteur['nom'] . '(' . $auteur['age'] . ' ans)';
    }
  }
}
?>
```

D'après le code de la fonction, on note que, si l'adresse est trouvée, alors le nom et l'âge de l'utilisateur sont retournés. En revanche, que se passe-t-il si l'adresse n'est pas trouvée ? Eh bien, il ne se passe rien car le cas n'a pas été prévu. Vous avez deux options pour tenter de corriger ce problème :

- retourner une chaîne de caractères même si l'adresse n'est pas trouvée (`"utilisateur inconnu"`) ;
- vérifier la liste : il y a peut-être une coquille dans l'une des adresses...

## En résumé

- Vous rencontrerez des bogues dans vos projets, c'est inéluctable.
- Ne paniquez pas ; lisez et essayez de comprendre le message d'erreur.
- Vous pouvez aussi copier le message d'erreur et le rechercher sur Internet. Il y a de fortes chances que d'autres personnes aient déjà rencontré ce problème.

# 11 Organiser les pages en blocs fonctionnels

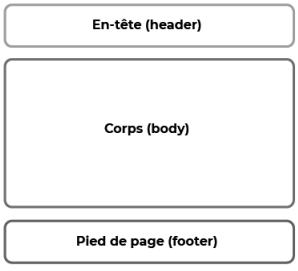
Jusque là, nous avons travaillé sur la page d'accueil de notre site, qui contient la liste des articles. Cependant, notre projet va être constitué de plusieurs autres pages : les recettes, une page d'édition/création de recette et un formulaire de contact. Pour naviguer entre elles, quelques liens HTML regroupés dans un menu principal seront nécessaires.

Vous est-il déjà arrivé de vouloir modifier le menu de votre site et de devoir pour cela corriger le code HTML de chacune de vos pages web ? Il est en effet très souvent recopié sur chacune d'elles. Cela fonctionne, mais ce n'est pas très pratique…

Une des fonctionnalités les plus simples et les plus utiles de PHP est l'**inclusion de pages**. On peut très facilement inclure tout ou partie d'une page à l'intérieur d'une autre. Cela évite de copier plusieurs fois le même code HTML.

## Le découpage d'une page web

Les sites web sont généralement découpés selon le schéma suivant.




Découpage usuel d'une page web

Jusqu'ici, vous étiez condamné à copier à l'identique sur chaque page :

- l'en-tête ;
- le menu ;
- le pied de page.

Habituellement, seul le contenu du corps change.

Regardez le code d'exemple suivant qui décrit une page web (appelons-la `index.php`) avec en-tête, menu et pied de page :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Mon super site</title>
  </head>

  <body>
    <!-- L'en-tête -->
    <header>
    </header>

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

    <!-- Le corps -->
    <div id="corps">
      <h1>Mon super site</h1>
      <p>
        Bienvenue sur mon super site !<br />
        Vous allez adorer ici, c'est un site génial qui va parler de...
euh... Je cherche encore un peu le thème de mon site. :-D
      </p>
    </div>

    <!-- Le pied de page -->
    <footer id="pied_page">
      <p>Copyright moi, tous droits réservés</p>
    </footer>
  </body>
</html>
```

## Penser le contenu en blocs fonctionnels

En PHP, nous pouvons facilement insérer à l'intérieur d'une page tout ou partie d'autres pages. Le principe de fonctionnement des **inclusions** est plutôt simple à comprendre. Vous écrivez les parties de code concernées dans autant de fichiers, que vous incluez ensuite sur toutes les pages. Ainsi, vous n'aurez à corriger qu'un seul fichier pour mettre à jour toutes les pages.

Prenons l'exemple du menu. Créez un fichier `entete.php` et insérez-y uniquement le code HTML correspondant :

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

Procédez de même pour le pied de page et tout autre page utile en fonction des besoins de votre site.

Mais… la page `entete.php` ne contient pas le moindre code PHP…

Une page dont l'extension est `.php` peut très bien ne contenir aucune balise PHP (même si c'est plutôt rare). Dans ce cas, cela redevient une page HTML classique qui n'est pas modifiée avant l'envoi. Ici, en théorie, vous pourriez très bien l'enregistrer avec l'extension `.html`, soit `entete.html`. Néanmoins, afin d'éviter de mélanger des pages de natures différentes sur votre site, je vous recommande de travailler uniquement avec l'extension `.php` à partir de maintenant.

Maintenant que vos « morceaux » sont prêts, reprenez les pages de votre site, par exemple la page d'accueil nommée `index.php`. Remplacez le menu par le code PHP suivant :

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

Cette instruction ordonne à l'ordinateur : « Insère ici le contenu de la page `entete.php` ».

Reprenons le code présenté au début du chapitre et remplaçons chaque code répétitif par un `include` :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Mon super site</title>
  </head>

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

    <!-- Le corps -->
    <div id="corps">
      <h1>Mon super site</h1>
      <p>
        Bienvenue sur mon super site !<br />
        Vous allez adorer ici, c'est un site génial qui va parler de...
euh... Je cherche encore un peu le thème de mon site. :-D
      </p>
    </div>

    <!-- Le pied de page -->
    <?php include("pied_page.php"); ?>
  </body>
</html>
```

Ce code suppose que votre page `index.php` et celles qui sont incluses (comme `entete.php`) sont dans le même dossier. Si le menu était dans un sous-dossier appelé `pourInclure`, il aurait fallu écrire :

```
<?php include("pourInclure/entete.php"); ?>
```

C'est le même principe que pour les liens relatifs, que vous connaissez déjà dans le langage HTML.

La page finale que reçoit le visiteur est identique à celle du début du chapitre… mais vous, vous avez énormément gagné en flexibilité puisque votre code n'est plus recopié 150 fois inutilement.

Le nombre d'`include` par page n'est pas limité ; par conséquent, vous pouvez découper votre code en autant de sous-parties que vous le souhaitez.

## Organiser le site de recettes

Maintenant, appliquons ce que nous venons d'apprendre à notre projet.

Nous avons besoin de fonctions et de variables pour afficher la liste de nos articles et, puisque nous aurons peut-être besoin de les réutiliser (surtout les fonctions), nous allons également extraire ces concepts dans leurs propres fichiers, respectivement `variables.php` et `fonctions.php`.

```
<?php
// variables.php

$utilisateurs = [
   [
      'nom' => 'Mickaël Andrieu',
      'courriel' => 'mickael.andrieu@exemple.com',
      'age' => 34,
   ],
   [
      'nom' => 'Mathieu Nebra',
      'courriel' => 'mathieu.nebra@exemple.com',
      'age' => 34,
   ],
   [
      'nom' => 'Laurène Castor',
      'courriel' => 'laurene.castor@exemple.com',
      'age' => 28,
   ],
];


$recettes = [
   [
      'titre' => 'Cassoulet',
      'recette' => '',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estValide' => true,
   ],
   [
      'titre' => 'Couscous',
      'recette' => '',
      'auteur' => 'mickael.andrieu@exemple.com',
      'estValide' => false,
   ],
   [
      'titre' => 'Escalope milanaise',
      'recette' => '',
      'auteur' => 'mathieu.nebra@exemple.com',
      'estValide' => true,
   ],
   [
      'titre' => 'Salade romaine',
      'recette' => '',
      'auteur' => 'laurene.castor@exemple.com',
      'estValide' => false,
   ],
];
?>
```

```
<?php
// fonctions.php

function recetteOK(array $recette) : bool
{
   if (array_key_exists('estValide', $recette)) {
      $OK = $recette['estValide'];
   } else {
      $OK = false;
   }
   return $OK;
}


function afficherAuteur(string $courrielAuteur, array $utilisateurs) : string
{
   for ($i = 0; $i < count($utilisateurs); $i++) {
      $auteur = $utilisateurs[$i];
      if ($courrielAuteur === $auteur['courriel']) {
         return $auteur['nom'] . '(' . $auteur['age'] . ' ans)';
      }
   }
}


function trouverRecettes(array $recettes) : array
{
   $recettesValides = [];
   foreach($recettes as $recette) {
      if (recetteOK($recette)) {
         $recettesValides[] = $recette;
      }
   }
      return $recettesValides;
}
?>
```

## En-tête

Ensuite, nous aurons évidement un en-tête de site avec un menu et un titre :

```
<!-- entete.php -->
<nav class="navbar navbar-expand-lg navbar-light bg-light">
   <div class="container-fluid">
      <a class="navbar-brand" href="index.php">Site de recettes</a>
      <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
         <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarSupportedContent">
         <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
```

```
            <a class="nav-link active" aria-current="page" href="index.
php">Accueil</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="contact.php">Contact</a>
          </li>
        </ul>
      </div>
    </div>
</nav>
```

Le menu est basique pour l'instant, mais il vous permet de passer de la page d'accueil à celle de contact, par exemple.

Ce code HTML est un peu compliqué à comprendre. La partie visuelle est assurée par le *framework* CSS Bootstrap 5 : il n'est pas nécessaire de le connaître pour suivre ce cours. Gardez juste en tête que chacune des classes ajoutées va permettre de concevoir la page correctement. Si cela vous ennuie, supprimez toutes les classes et le projet restera fonctionnel, c'est promis !

## Pied de page

Cette partie du site reste « ferrée » en bas de page et contient quelques liens et surtout les conditions d'utilisation du site (les fameux *copyrights*) :

```
<!-- pied_page.php -->
<footer class="bg-light text-center text-lg-start mt-auto">
  <div class="text-center p-3">
    © 2021 Copyright:
    <a class="text-dark" href="https://openclassrooms.com/">OpenClassrooms</a>
  </div>
</footer>
```

## Corps de la page d'accueil

Maintenant, nous pouvons reprendre ce que nous avions construit ensemble dans les chapitres précédents. Nous devons inclure le menu, les variables, les fonctions et le pied de page :

```
<!-- index.php -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Site de recettes - Page d'accueil</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/
bootstrap.min.css"
      rel="stylesheet"
    >
  </head>

  <body class="d-flex flex-column min-vh-100">
    <div class="container">

      <!--inclusion de l'en-tête du site -->
      <?php include_once('entete.php'); ?>
      <h1>Site de recettes</h1>

      <!-- inclusion des variables et fonctions -->
      <?php
        include_once('variables.php');
        include_once('fonctions.php');
      ?>

      <?php foreach(trouverRecettes($recettes) as $recette) : ?>
        <article>
          <h3><?php echo $recette['titre']; ?></h3>
          <div><?php echo $recette['recette']; ?></div>
          <i><?php echo afficherAuteur($recette['auteur'], $utilisateurs);
?></i>
        </article>
      <?php endforeach ?>
    </div>

    <!-- inclusion du bas de page du site -->
    <?php include_once('pied_page.php'); ?>
  </body>
</html>
```

Pressé(e) de voir le résultat ?


Affichage du site web avec l'en-tête et le pied de page

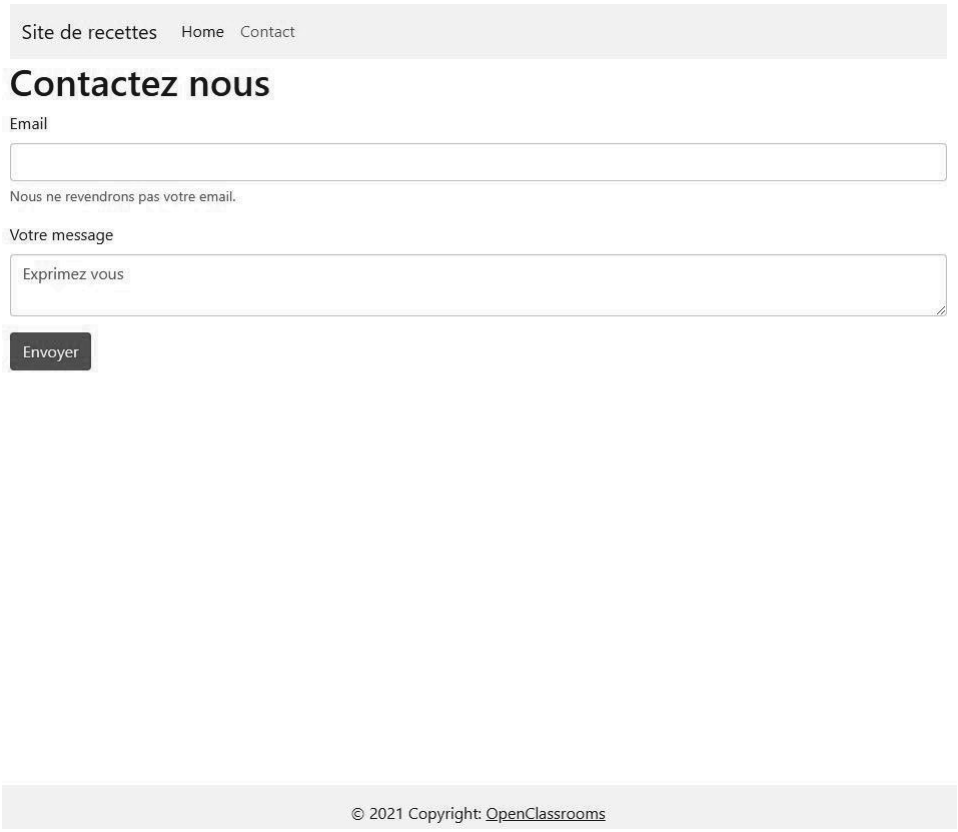
## Corps de la page de contact

En bonus, nous allons ajouter un simple formulaire HTML. Nous verrons dans les prochains chapitres comment le traiter avec PHP de manière à avoir une navigation fonctionnelle.

Voici la page de contact à ce stade du projet :

```
<!-- contact.php -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Site de recettes - Formulaire de Contact</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/
bootstrap.min.css"
      rel="stylesheet"
    >
  </head>

  <body class="d-flex flex-column min-vh-100">
    <div class="container">
      <?php include_once('entete.php'); ?>
      <h1>Contactez nous</h1>
      <form>
        <div class="mb-3">
          <label for="courriel" class="form-label">Courriel</label>
          <input type="email" class="form-control" id="courriel"
name="courriel" aria-describedby="aideCourriel">
          <div id="aideCourriel" class="form-text">Nous ne revendrons pas
votre email.</div>
        </div>
        <div class="mb-3">
          <label for="message" class="form-label">Votre message</label>
          <textarea class="form-control" placeholder="Exprimez-vous"
id="message" name="message"></textarea>
        </div>
        <button type="submit" class="btn btn-primary">Envoyer</button>
      </form>
      <br />
    </div>

    <?php include_once('pied_page.php'); ?>
  </body>
</html>
```

En voici le résultat visuel.




Le formulaire de contact

## En résumé

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

# Troisième partie

# Transmettre des données de page en page

Maintenant que vous avez acquis les bases de PHP, nous pouvons nous intéresser à du concret.

Le langage PHP a été conçu pour que vous puissiez transmettre des informations de page en page, au fil de la navigation d'un visiteur sur votre site. C'est notamment ce qui vous permet de retenir son pseudonyme tout au long de sa visite, mais aussi de récupérer et de traiter les informations qu'il renseigne sur votre site, notamment dans des formulaires.

Grâce à cette partie, vous allez pouvoir commencer à vraiment communiquer avec vos visiteurs !

# 12 Transmettre des données avec l'URL

Grâce aux URL et à PHP, nous allons traiter le formulaire soumis par l'utilisateur et retourner une page indiquant que la demande a bien été prise en compte.

URL signifie *Uniform Resource Locator*. C'est une adresse sur le Web. Toutes les adresses que vous voyez en haut de votre navigateur, comme *http://www.openclassrooms.com*, sont des URL.

Lorsque vous lancez une recherche sur Google en tapant le mot `OpenClassrooms`, la barre d'adresse contient une URL un peu longue qui ressemble à ceci :

*http://www.google.fr/search?rlz=1C1GFR343&q=openclassrooms*

Les informations situées après le point d'interrogation sont des données que l'on fait transiter d'une page à une autre. Nous allons découvrir dans ce chapitre comment cela fonctionne.

## Envoyer des paramètres dans l'URL

Les URL servent à transmettre des informations. Comment cela fonctionne-t-il exactement ?

### Former une URL pour envoyer des paramètres

Imaginons que votre site s'appelle `monsite.com` et a une page PHP intitulée `bonjour.php`. Pour accéder à cette dernière, vous devez aller à l'URL suivante :

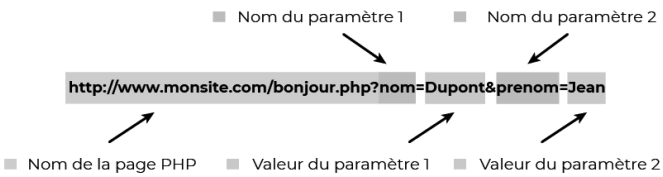
*http://www.monsite.com/bonjour.php*

Pour **envoyer** des informations à la page `bonjour.php`, on va ajouter des informations à la fin de l'URL :

*http://www.monsite.com/bonjour.php?nom=Dupont&prenom=Jean*

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

La figure suivante montre comment est composée cette URL.




Structure d'une URL

Le point d'interrogation sépare le nom de la page PHP des paramètres. Ensuite, ces derniers s'enchaînent selon la forme `nom=valeur` et sont séparés les uns des autres par le symbole `&`.

Peut-on écrire autant de paramètres qu'on veut ?

En théorie, oui. Il suffit de les séparer par le symbole `&`. On peut donc voir une URL de la forme suivante :

*page.php?param1=valeur1&param2=valeur2&param3=valeur3&param4=valeur4…*

La seule limite est la longueur de l'URL. En général, il n'est pas conseillé de dépasser les 256 caractères, mais les navigateurs arrivent parfois à gérer des URL plus longues.

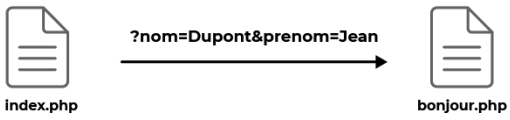
## Créer un lien avec des paramètres

Maintenant, nous pouvons créer des liens en HTML qui transmettent des paramètres d'une page vers une autre.

Imaginons que vous ayez deux fichiers sur votre site :

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

Nous voulons créer un lien de `index.php` vers `bonjour.php` qui transmet des informations dans l'URL, comme le schématise la figure suivante.




Lien entre index.php et bonjour.php

Pour cela, ouvrez `index.php` (puisque c'est ce fichier qui contiendra le lien) et insérez-y par exemple le code suivant :

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

Le symbole `&` a été remplacé par `&` dans le lien. Il s'agit de la convention HTML. Si vous ne l'écrivez pas ainsi, le code source ne passera pas la validation W3C.

Ce lien appelle la page `bonjour.php` et lui envoie deux paramètres :

- `nom`, avec la valeur `Dupont` ;
- `prenom`, avec la valeur `Jean`.

### Créer un formulaire avec la méthode HTTP GET

La deuxième solution pour faire passer des informations dans l'URL est de proposer à l'utilisateur de soumettre un formulaire avec la méthode HTTP `GET`. Cela se code avec une balise `form` dont l'attribut `method` a la valeur `GET`.

```
<form action="bonjour.php" method="GET">
  <!-- données à faire passer à l'aide de champs input -->
  <input name="nom">
  <input name="prenom">
</form>
```

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

Voyons maintenant comment récupérer ces informations.

## Récupérer les paramètres en PHP

Intéressons-nous maintenant à la page qui reçoit les paramètres.

Dans notre exemple, il s'agit de `bonjour.php`. Elle va automatiquement créer une variable dite superglobale, au nom un peu spécial : `$_GET`. Il s'agit d'un tableau associatif dont les clés correspondent aux noms des paramètres envoyés dans l'URL.

Nous avons fait un lien vers *bonjour.php?nom=Dupont&prenom=Jean*. Cela signifie que nous aurons accès aux variables suivantes :

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

On peut récupérer ces informations, les traiter, les afficher… autrement dit, on est libre de faire ce qu'on veut avec. Pour commencer, essayons tout simplement de les afficher.

Créez le fichier `bonjour.php` et placez-y le code suivant :

Bien sûr, pour une vraie page web, il faudrait écrire toutes les informations d'en-tête nécessaires en HTML : le `doctype`, la balise `<head>`, etc. Ici, pour faire simple, je n'écris pas tout ce code. La page ne sera pas très belle (ni valide W3C d'ailleurs) mais, pour l'instant, nous procédons juste à des tests.

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

Ici, nous affichons le prénom qui a été passé dans l'URL. Bien entendu, nous avons aussi accès au nom. Nous pouvons afficher le nom et le prénom sans problème :

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

Comme vous le voyez, j'en ai profité pour faire une petite concaténation. Ce code affiche le contenu de deux cases du tableau `$_GET`.

Pour notre site de recettes, nous aurons un formulaire, `contact.php`, qui affichera un message de bonne réception via la page `accuse_reception.php`.

Pour rappel, voici le formulaire de contact (simplifié) que nous avions dans le chapitre précédent :

```
<form action="accuse_reception.php" method="GET">
   <div>
      <label for="courriel">Courriel</label>
      <input type="email" name="courriel">
   </div>
   <div>
      <label for="message">Votre message</label>
      <textarea placeholder="Exprimez vous" name="message"></textarea>
   </div>
   <button type="submit">Envoyer</button>
</form>
```

Par exemple, si l'utilisateur soumet l'adresse *utilisateur@exemple.com* et le message `"Bonjour"`, le formulaire est alors converti en lien vers :

*accuse_reception.php?courriel=utilisateur%40exemple.com&message=Bonjour*

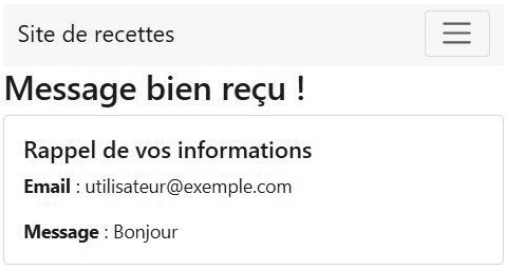
Lors de la soumission, la variable superglobale `$_GET` contient les données envoyées.

| Nom | Valeur |
|---|---|
| `$_GET['courriel']` | `utilisateur@exemple.com` |
| `$_GET['message']` | `Bonjour` |

Ces informations sont récupérées dans le fichier `accuse_reception.php` :

```
<h1>Message bien reçu !</h1>
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Rappel de vos informations</h5>
    <p class="card-text"><b>email</b> : <?php echo $_GET['courriel']; ?></
p>
    <p class="card-text"><b>Message</b> : <?php echo $_GET['message']; ?></
p>
  </div>
</div>
```

Vous obtiendrez le résultat suivant :




Message de bonne réception du formulaire de contact

## Ne jamais faire confiance aux données reçues !

Il ne faut JAMAIS faire confiance aux variables qui transitent de page en page, comme `$_GET` que nous étudions ici.

Oui je suis alarmiste et oui je veux vous faire – un peu – peur, mais rassurez-vous, je vais vous expliquer tout ce qu'il faut savoir pour que ça se passe bien et que vous ne risquiez rien.

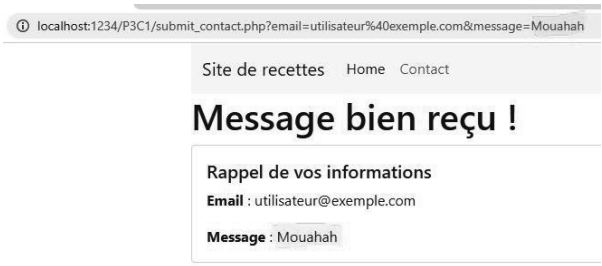
## N'importe quel visiteur peut trafiquer les URL

Si vous testez les codes précédents chez vous, vous devriez tomber sur une URL de la forme :

*http://localhost/accuse_reception.php?courriel=utilisateur%40exemple.com&message=Bonjour*

Si vous êtes un peu bricoleur, vous pouvez vous amuser à changer les paramètres directement dans la barre d'adresse, comme dans la figure suivante.

*http://localhost/accuse_reception.php?courriel=utilisateur%40exemple.com&message=Mouahah*




On peut facilement modifier les paramètres dans le navigateur.

Comme vous le voyez, ça fonctionne ! N'importe qui peut facilement modifier les URL dans la barre d'adresse du navigateur et y écrire ce qu'il veut.

Et alors, quel est le problème ? Ce n'est pas plus mal, si le visiteur veut adapter l'URL pour modifier son message ou son adresse…

Peut-être, mais cela montre une chose : **on ne peut pas faire confiance aux données qu'on reçoit** puisque n'importe qui peut les changer.

## Tester 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/accuse_reception.php*

Or, cette page va afficher quelque chose comme :

```
1. Notice: Undefined index: courriel in accuse_reception.php on line 15
ou
Warning: Undefined array key "courriel" in accuse_reception.php on line 15
```

Que s'est-il passé ? On a essayé d'afficher la valeur de `$_GET['courriel']` et celle de `$_GET['message']` mais, comme on vient de les supprimer de l'URL, ces variables n'ont pas été créées ! PHP nous avertit qu'on essaie d'utiliser des variables qui n'existent pas.

Pour résoudre ce problème, on peut faire appel à une fonction un peu spéciale, `isset()`, qui teste si une variable existe. Nous allons nous en servir pour afficher un message spécifique si une information est absente.

```
<?php
if (!isset($_GET['courriel']) || !isset($_GET['message'])) {
   echo('Il faut une adresse et un message pour soumettre le formulaire.');
   // Arrête l'exécution de PHP
   return;
}
?>
```

Ce code teste si les variables `$_GET['courriel']` et `$_GET['message']` existent. Si c'est le cas, on affiche la confirmation de réception du message. Sinon, on affiche un message d'erreur. Essayez maintenant d'accéder à la page `accuse_reception.php` sans les paramètres. Vous allez voir qu'on gère bien le cas où le visiteur aurait retiré les paramètres de l'URL.

Est-ce vraiment la peine de prévoir ce cas ? Moi je ne m'amuse jamais à modifier mes URL et mes visiteurs non plus, je pense !

Oui, oui, **trois fois oui** ! Il faut anticiper le cas où des paramètres que vous attendiez viendraient à manquer.

Il ne faut jamais faire confiance à l'utilisateur. Tôt ou tard, vous tomberez sur une personne malintentionnée qui essaiera de trafiquer l'URL pour mettre n'importe quoi dans les paramètres. Il faut que votre site soit prêt à faire face à cette configuration.

Dans notre exemple, si on ne prévoyait pas le cas, ce n'était pas bien grave (on affichait juste des messages d'erreur). Toutefois, lorsque votre site web deviendra plus complexe, cela pourrait avoir des conséquences plus ennuyeuses.

## Contrôler la valeur des paramètres

Une fois les valeurs soumises et correctement récupérées, il faut vérifier qu'elles correspondent à ce qui est attendu. Imaginez par exemple que l'utilisateur se soit trompé et que l'adresse envoyée ne soit pas valide : vous ne pourrez donc pas le recontacter pour répondre à son besoin. Il va donc falloir, d'une part, contrôler si l'adresse transmise est bien valide à l'aide de la fonction `filter_var` et, d'autre part, vérifier que le message n'est pas vide à l'aide de la fonction `empty`.

Voici donc le code final sécurisé, qui prévoit tous les cas pour éviter d'être pris au dépourvu par un utilisateur mal intentionné :

```
<?php
if (
   (!isset($_GET['courriel']) || !filter_var($_GET['courriel'],
FILTER_VALIDATE_EMAIL))
   || (!isset($_GET['message']) || empty($_GET['message']))
   )
{
   echo('Il faut une adresse et un message valides pour soumettre le
formulaire.');
   return;
}
```

Cela fait beaucoup de conditions pour quelque chose qui était à la base assez simple, mais c'est nécessaire. Vous devrez toujours faire très attention et prévoir tous les cas les plus tordus, comme nous venons de le faire : vérifier que tous les paramètres que vous attendiez sont bien là et qu'ils contiennent des valeurs correctes, comprises dans des intervalles raisonnables. Si vous gardez cela en tête, vous n'aurez pas de problèmes !

Nous n'avons pas détaillé le mécanisme des fonctions `filter_var` et `empty`. N'hésitez pas à consulter tous les exemples de la documentation PHP pour bien comprendre.

## En résumé

- Une URL représente l'adresse d'une page web (commençant généralement par *http://*).
- Lorsqu'on code un lien vers une page, il est possible d'ajouter des paramètres sous la forme `fichier.php?param1=valeur1&param2=valeur2` qui seront transmis.
- La page de destination reçoit ces paramètres dans un tableau nommé `$_GET`.
- Cette technique est très pratique pour transmettre des valeurs à une page, mais il faut prendre garde au fait que le visiteur peut les modifier très facilement.
- La fonction `isset()` vérifie si une variable est définie ou non.
- D'autres fonctions, comme `filter_var()` ou `empty()`, sont utiles pour valider les champs.

# 13 Administrer des formulaires de façon sécurisée

Les formulaires constituent le principal moyen pour vos visiteurs d'entrer des informations sur votre site. Dans le chapitre précédent, nous avions abordé un premier formulaire de contact assez basique pour envoyer une URL avec des paramètres. Maintenant, nous allons apprendre à créer des formulaires plus sécurisés.

Ce chapitre est particulièrement important ; nous allons réutiliser tout ce que nous avons appris ici. Soyez attentifs !

## Créer la base du formulaire

Retenez bien que vous travaillez normalement sur deux pages différentes : celle qui contient le formulaire (`contact.php` dans notre exemple) et celle qui reçoit les données du formulaire pour les traiter (`accuse_reception.php`).

En HTML, pour insérer un formulaire, on se sert de la balise `<form>`. On l'utilise de la manière suivante :

```
< !-- contact.php -->
<form method="post" action="accuse_reception.php">
  <p>
    On insérera ici les éléments de notre formulaire.
  </p>
</form>
```

Il faut connaître deux attributs très importants pour la balise `<form>` : la méthode (`method`) et la cible (`action`). Il est impératif que vous compreniez à quoi ils servent.

### Privilégier la méthode POST pour envoyer les données du formulaire

Retenez deux méthodes pour envoyer les données du formulaire :

- `get` : les données transiteront par l'URL comme on l'a appris précédemment. On pourra les récupérer grâce au tableau `$_GET`. Cette méthode est assez peu utilisée car on ne peut pas envoyer beaucoup d'informations dans l'URL.
- `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 qu'on veut et est le plus souvent privilégiée. 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. **On ne doit pas plus faire confiance aux formulaires qu'aux URL**.

### Choisir la page cible du formulaire

L'attribut `action` sert à définir la page appelée par le formulaire. C'est cette dernière 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 : `$_GET` ou `$_POST`.

### Ajouter des champs input avec leur attribut name

La variable `$_GET` sera complétée avec la valeur de l'attribut `name` en tant que clé et ce qui est soumis par l'utilisateur en tant que valeur.

```
<input type="text" name="nom" value="Mateo21" />

<?php
// Après soumission du formulaire
echo $_GET['nom']; // "Mateo21"

// OU
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.

### Faire attention avec les champs cachés

Les champs cachés constituent une catégorie à part. Il s'agit de codes dans votre formulaire qui n'apparaîtront pas aux yeux du visiteur, mais qui vont quand même créer une variable avec une valeur. On peut s'en servir pour transmettre des informations fixes.

Supposons que vous ayez besoin de « retenir » que le pseudo du visiteur est `"Mateo21"`. Vous allez saisir le code suivant :

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

À l'écran, sur la page web, on ne verra rien. Pourtant, dans la page cible, une variable `$_POST['pseudo']` sera créée avec la valeur `"Mateo21"` !

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

On croit souvent à tort que, parce que ces champs sont cachés, le visiteur ne peut pas les voir. C'est faux ! En effet, n'importe quel visiteur peut afficher le code source de la page pour voir et modifier les champs cachés en lisant le code HTML.

## Ne faites jamais confiance aux données reçues : la faille XSS

Les mises en garde du chapitre précédent ne concernaient pas que les paramètres qui transitent par l'URL : tout cela vaut aussi pour les formulaires !

Même si je vois comment on peut modifier l'URL, je ne comprends pas comment un visiteur peut modifier le formulaire de mon site et trafiquer les données !

### Attention au code HTML que vous recevez !

La faille XSS (pour *cross-site scripting*) est vieille comme le monde (euh… 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.

Reprenons notre page de récapitulatif avec la méthode `post` :

```
<h5>Rappel de vos informations</h5>
<p><b>Courriel</b> : <?php echo $_POST['courriel']; ?></p>
<p><b>Message</b> : <?php echo $_POST['message']; ?></p>
```

Si le visiteur décide d'écrire du code HTML dans la zone de message, cela fonctionnera très bien ! Par exemple, imaginons qu'il envoie le code `<strong>Badaboum </strong>`. Le code source HTML généré par PHP sera le suivant :

```
<p><b>Message</b> : <strong>Badaboum</strong></p>
```

Et alors ? S'il veut mettre son message en gras, c'est son problème, non ?

Outre le fait qu'il peut insérer n'importe quel code HTML (et rendre votre page invalide), ce qui n'est pas le plus grave, il peut aussi ouvrir des balises de type `<script>` pour faire exécuter du code JavaScript au visiteur qui visualisera la page.

```
<p><b>Message</b> : <script>alert('Badaboum')</script></p>
```

Tous les visiteurs qui arriveront sur cette page verront une boîte de dialogue JavaScript s'afficher. C'est plutôt gênant.


Exécution de JavaScript par la faille XSS

## Sécuriser en bloquant l'exécution de code JavaScript

Pour ignorer le code HTML, il suffit d'utiliser la fonction `htmlspecialchars`, qui va transformer les chevrons des balises HTML `<>` en `&lt;` et `&gt;` respectivement. Cela provoquera l'affichage de la balise plutôt que son exécution.

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

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

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

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

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

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

Voici donc une nouvelle version du code de réception du formulaire (avec `post`) sécurisée, qui prévoit tous les cas pour éviter d'être pris au dépourvu par un utilisateur mal intentionné :

```
<?php
// accuse_reception.php
if (!isset($_POST['courriel']) || !isset($_POST['message']))
{
   echo('Il faut une adresse et un message pour soumettre le formulaire.');
   return;
}

$courriel = $_POST['courriel'];
$message = $_POST['message'];
?>


<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Site de Recettes - Demande de contact reçue</title>
      <link
         href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/
bootstrap.min.css"
         rel="stylesheet"
      >
   </head>
   <body>
      <div class="container">
      <?php include_once('entete.php'); ?>
         <h1>Message bien reçu !</h1>
         <div class="card">
            <div class="card-body">
               <h5 class="card-title">Rappel de vos informations</h5>
               <p class="card-text"><b>email</b> : <?php echo($courriel); ?>
</p>
               <p class="card-text"><b>Message</b> : <?php echo strip_
tags($message); ?></p>
            </div>
         </div>
      </div>
   </body>
</html>
```

## En résumé

- Les formulaires sont le moyen le plus pratique pour le visiteur de transmettre des informations à votre site. PHP est capable de récupérer les données saisies et de les traiter.
- Les données envoyées via un formulaire se retrouvent dans un tableau `$_POST`.
- De la même manière que pour les URL, il ne faut pas accorder sa confiance absolue aux données que vous envoie l'utilisateur. Traitez les données reçues avec vigilance.
- Que ce soit pour des données issues de l'URL (`$_GET`) ou d'un formulaire (`$_POST`), il faut s'assurer qu'aucun texte qui vous est envoyé ne contient du HTML s'il est destiné à être affiché sur une page. Sinon, vous ouvrez une faille appelée XSS qui peut être néfaste pour la sécurité de votre site.
- Pour éviter la faille XSS, il suffit d'appliquer la fonction `htmlspecialchars` ou `strip_tags` sur tous les textes envoyés par vos visiteurs.

# 14 Activer le partage de fichiers

Imaginez qu'une personne essaie de vous contacter car elle est victime d'un bogue d'utilisation sur votre site ; elle souhaiterait peut-être vous partager une capture d'écran, ce qui faciliterait beaucoup votre travail de débogage.

Pour cela, vous devez proposer la soumission de fichiers dans votre formulaire de contact.

Vous avez découvert les superglobales `$_GET` et `$_POST` dans les chapitres précédents ; c'est maintenant à `$_FILES` de faire son entrée dans votre projet.

## Envoyer des fichiers

Il est possible d'envoyer des fichiers grâce aux formulaires. Là encore, cela se déroule en deux temps :

1. Le visiteur arrive sur votre formulaire et le remplit (en indiquant le fichier à envoyer).
2. PHP reçoit les données du formulaire et, s'il y a des fichiers dedans, il les « enregistre » dans l'un des dossiers du serveur.

L'envoi du fichier peut être un peu long si celui-ci est gros. Il faudra afficher un message invitant le visiteur à patienter pendant l'envoi.

On va commencer par adapter le formulaire de contact.

## Paramétrer le formulaire d'envoi de fichier

Dès l'instant où votre formulaire propose aux visiteurs d'envoyer un fichier, il faut ajouter l'attribut `enctype="multipart/form-data"` à la balise `<form>`.

```
<form action="accuse_reception.php" method="post" enctype="multipart/
form-data">
   < !-- champs du formulaire -->
</form>
```

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

Maintenant que c'est fait, nous pouvons ajouter à l'intérieur du formulaire une balise permettant d'envoyer un fichier. Elle est de type `<input type="file" />`. Il faut donner un nom à ce champ de formulaire (grâce à l'attribut `name`) pour que PHP puisse le reconnaître par la suite.

```
<form action="accuse_reception.php" method="POST" enctype="multipart/
form-data">
   <!-- Ajout des champs email et message -->
   [...]
   <!-- Ajout champ de téléversement -->
   <div class="mb-3">
      <label for="capture" class="form-label">Votre capture d'écran</label>
      <input type="file" class="form-control" id="capture" name="capture" />
   </div>
   <!-- Fin ajout du champ -->
   <button type="submit" class="btn btn-primary">Envoyer</button>
</form>
```

Voilà, c'est suffisant.

## Traiter l'envoi en PHP

C'est maintenant que cela devient important. Il faut qu'on écrive le code de la page `accuse_reception.php` pour traiter l'envoi du fichier.

« Traiter l'envoi du fichier » ? C'est-à-dire ?

Si le fichier a été envoyé sur le serveur c'est bon, non ? Qu'est-ce que PHP aurait besoin de faire ?

En fait, au moment où la page PHP s'exécute, le fichier a été envoyé sur le serveur mais il est stocké dans un **dossier temporaire**. C'est à vous de décider si vous l'acceptez définitivement ou non. Par exemple, vous pouvez vérifier si le fichier a la bonne extension (si vous demandiez une image et qu'on vous envoie un `.txt`, vous devrez refuser le fichier).

Si le fichier est bon, vous l'accepterez définitivement grâce à la fonction `move_uploaded_file`.

Comment savoir si le fichier est valide ?

Pour chaque fichier envoyé, une variable `$_FILES['nom_du_champ']` est créée. Dans notre cas, la variable s'appellera `$_FILES['capture']`.

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

| Variable | Signification |
|---|---|
| `$_FILES['capture']['name']` | Contient le nom du fichier envoyé par le visiteur. |
| `$_FILES['capture']['type']` | Indique le type du fichier envoyé. Si c'est une image `.gif`, par exemple, le type sera `image/gif`. |
| `$_FILES['capture']['size']` | Indique la taille en octets du fichier envoyé. Il faut environ 1 000 octets pour faire 1 Ko, et 1 000 000 d'octets pour faire 1 Mo.<br>Attention : la taille de l'envoi est limitée par PHP. Par défaut, il est impossible de transmettre des fichiers de plus de 8 Mo. |
| `$_FILES['capture']['tmp_name']` | Juste après l'envoi, le fichier est placé dans un répertoire temporaire sur le serveur en attendant que votre script PHP décide si oui ou non il accepte de le stocker pour de bon. Cette variable contient l'emplacement temporaire du fichier (c'est PHP qui gère cela). |
| `$_FILES['capture']['error']` | Contient un code d'erreur indiquant 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 inséré un second champ d'envoi de fichier dans votre formulaire, il y aura une seconde variable `$_FILES['nom_de_votre_autre_champ']['size']` contiendra donc la taille du second fichier et ainsi de suite.

Je vous propose de procéder aux vérifications suivantes pour décider si l'on accepte ou non le fichier.

1. Vérifier tout d'abord si le visiteur a bien envoyé un fichier (en testant la variable `$_FILES['capture']` avec `isset()`) et s'il n'y a pas eu d'erreur d'envoi (grâce à `$_FILES['capture']['error']`).
2. Vérifier si la taille du fichier ne dépasse pas 1 Mo, par exemple (environ 1 000 000 d'octets) grâce à `$_FILES['capture']['size']`.
3. Vérifier si l'extension du fichier est autorisée (il faut interdire à tout prix que les gens puissent envoyer des fichiers PHP, sinon ils pourraient exécuter des scripts sur votre serveur). Dans notre cas, nous autoriserons seulement les images (fichiers `.png`, `.jpg`, `.jpeg` et `.gif`). Nous analyserons pour cela la variable `$_FILES['capture']['name']`.

Nous allons donc coder une série de tests dans notre page `accuse_reception.php`.

## Tester si le fichier a bien été envoyé

On commence par vérifier qu'un fichier a été envoyé. Pour cela, on va tester si la variable `$_FILES['capture']` existe avec `isset()`.

On vérifie dans le même temps s'il n'y a pas d'erreur d'envoi.

```
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas d'erreur
if (isset($_FILES['capture']) AND $_FILES['capture']['error'] == 0) {

}
?>
```

## Vérifier la taille du fichier

On veut interdire les fichiers de plus de 1 Mo, soit environ 1 000 000 octets (j'arrondis pour simplifier). On doit donc tester `$_FILES['capture']['size']` :

```
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas d'erreur
if (isset($_FILES['capture']) AND $_FILES['capture']['error'] == 0) {
   // Testons si le fichier n'est pas trop gros
   if ($_FILES['capture']['size'] <= 1000000)
      {

   }
}
?>
```

## Vérifier l'extension du fichier

On peut récupérer l'extension du fichier dans une variable grâce au code suivant :

```
<?php
$infoFichier = pathinfo($_FILES['capture']['name']);
$extension = $infoFichier['extension'];
?>
```

La fonction `pathinfo` renvoie un tableau contenant entre autres l'extension du fichier dans `$infoFichier['extension']`. On stocke cette information dans une variable `$extension`. On vérifie ensuite si elle fait bien partie des valeurs autorisées à l'aide de la fonction `in_array()`. On obtient finalement le code suivant :

```
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas d'erreur
if (isset($_FILES['capture']) AND $_FILES['capture']['error'] == 0) {
   // Testons si le fichier n'est pas trop gros
   if ($_FILES['capture']['size'] <= 1000000) {
      // Testons si l'extension est autorisée
      $infoFichier = pathinfo($_FILES['capture']['name']);
      $extension = $infoFichier['extension'];
      $extensionsAutorisees = ['jpg', 'jpeg', 'gif', 'png'];
      if (in_array($extension, $extensionsAutorisees)) {

      }
   }
}
?>
```

## Valider le téléversement du fichier

Si tout est bon, on accepte le fichier en appelant `move_uploaded_file()`. Cette fonction prend deux paramètres :

- le nom temporaire du fichier (récupéré avec `$_FILES['capture']['tmp_name']`) ;
- le chemin et le nom sous lesquels sera stocké le fichier de façon définitive. On peut utiliser le nom d'origine du fichier `$_FILES['capture']['name']` ou générer un nom au hasard.

Je propose de placer le fichier dans un sous-dossier `televersements` et de garder le nom d'origine. Comme `$_FILES['capture']['name']` contient le chemin entier vers le fichier d'origine (`C:\dossier\fichier.png`, par exemple), il nous faudra extraire le nom du fichier. On peut utiliser pour cela la fonction `basename` qui renverra juste `fichier.png`.

```
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas d'erreur
if (isset($_FILES['capture']) AND $_FILES['capture']['error'] == 0) {
   // Testons si le fichier n'est pas trop gros
   if ($_FILES['capture']['size'] <= 1000000) {
      // Testons si l'extension est autorisée
      $infoFichier = pathinfo($_FILES['capture']['name']);
      $extension = $infoFichier['extension'];
      $extensionsAutorisees = ['jpg', 'jpeg', 'gif', 'png'];
      if (in_array($extension, $extensionsAutorisees)) {
         // On peut valider le fichier et le stocker définitivement
         move_uploaded_file($_FILES['capture']['tmp_name'], 'televersements/'
. basename($_FILES['capture']['name']));
         echo "L'envoi a bien été effectué !";
      }
   }
}
?>
```

Lorsque vous enverrez le script sur Internet à l'aide d'un logiciel FTP, vérifiez que le dossier `televersements` existe sur le serveur et qu'il a les droits d'écriture. Pour ce faire, sous FileZilla par exemple, cliquez droit sur le dossier et choisissez ***Attributs du fichier***.

Cela affiche les droits du dossier (on parle de *CHMOD*). Mettez-les à `733` ; ainsi, PHP pourra placer les fichiers téléversés dans ce dossier.

Ce script est un début, mais en pratique il vous faudra sûrement encore l'améliorer. Par exemple, si le nom du fichier contient des espaces ou des accents, cela posera un problème une fois envoyé sur le Web. En outre, si quelqu'un envoie un fichier qui a le même nom que celui d'une autre personne, l'ancien sera écrasé.

La solution consiste en général à « choisir » nous-même le nom du fichier stocké sur le serveur plutôt que de se servir du nom d'origine. Vous pouvez créer un compteur qui s'incrémente : `1.png`, `2.png`, `3.jpg`, etc.

Soyez toujours très vigilant sur la sécurité, vous devez éviter que quelqu'un puisse envoyer des fichiers PHP sur votre serveur.

## En résumé

- Les formulaires permettent d'envoyer des fichiers. On retrouve les informations sur ces derniers dans un tableau `$_FILES`. Leur traitement est cependant complexe.
- Il faudra toujours contrôler les fichiers reçus :
  – leur existence ;
  – leur taille (limitée par la configuration de PHP) ;

  – leur extension (à l'aide de la fonction `pathinfo()`) et surtout refuser tout fichier PHP qui pourrait par la suite s'exécuter sur votre serveur.

- À l'aide de la fonction `move_uploaded_files()`, vous pouvez conserver les fichiers téléversés sur votre serveur, mais vérifiez au préalable que vous avez les droits d'écriture sur le dossier concerné.

# 15 Protéger une page par mot de passe

Croyez-le ou non, vous avez déjà le niveau pour protéger le contenu d'une page par mot de passe !

Voici la liste des connaissances que vous allez devoir mobiliser pour cela : afficher du texte avec `echo`, utiliser des variables, transmettre des variables via une zone de texte dans un formulaire, utiliser des conditions simples (`if... else`) et inclure des fichiers avec `include` ou `include_once`.

Le but est de parvenir à assembler toutes vos connaissances pour répondre à un problème précis.

Vous voulez que les contributeurs et contributrices de votre site de recettes puissent se connecter et y être reconnus. Pour simplifier, il y aura un formulaire de connexion avec adresse électronique et mot de passe et, une fois la personne connectée, nous afficherons un message de bienvenue :

```
"Bonjour utilisateur@exemple.com et bienvenue sur le site !"
```

Nous ajouterons une contrainte : la liste des recettes ne sera affichée que si l'utilisateur est connecté.

Les utilisateurs seront déjà disponibles sous la forme d'un tableau associatif PHP. Ils ont chacun une clé `'motPasse'` avec un mot de passe et une clé `'courriel'` avec leur adresse e-mail.

Pour ne pas avoir à réécrire tout le site, vous démarrerez avec une archive du projet sans système de connexion, c'est-à-dire le projet fil rouge auquel nous sommes parvenus ensemble jusque-là (https://s3.eu-west-1.amazonaws.com/course.oc-static.com/courses/918836/TP+cours+PHP.zip). Vous pouvez donc concentrer votre attention sur la nouvelle fonctionnalité à implémenter.

Travailler d'abord au brouillon

Pour coder correctement, je recommande toujours de travailler d'abord au brouillon (vous savez, avec un stylo et une feuille de papier !). Cela semble bien souvent une perte de temps, mais c'est tout à fait le contraire. Si vous vous mettez à écrire des lignes de code au fur et à mesure, ça va être le bazar à coup sûr. À l'inverse, si vous prenez cinq minutes pour y réfléchir devant une feuille de papier, votre code sera mieux structuré et vous éviterez de nombreuses erreurs (qui font perdre du temps).

À quoi doit-on réfléchir sur notre brouillon ?

1. Au problème que vous vous posez (qu'est-ce que je veux réussir à faire ?).
2. Au schéma du code : vous allez commencer par le découper en plusieurs morceaux, eux-mêmes découpés en petits morceaux (c'est plus facile à avaler).
3. Aux fonctions et aux connaissances en PHP dont vous allez avoir besoin (pour être sûr que vous les utilisez convenablement).

## Poser le problème

On doit soumettre une adresse et un mot de passe dans un formulaire de connexion. Si le formulaire est valide, nous affichons un message de succès, sinon un message d'erreur. La liste de recettes n'est affichée qu'à un utilisateur qui s'est connecté avec succès.

## Schématiser 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 se présenter :

- Vous n'êtes **pas connecté** : le formulaire de contact s'affiche, mais pas la liste des recettes.
- Vous avez soumis le formulaire avec le **bon mot de passe** : le message de succès et la liste des recettes s'affichent, mais pas le formulaire de connexion.
- Vous avez soumis le formulaire avec le **mauvais mot de passe** : le message d'erreur et le formulaire de connexion s'affichent, mais pas la liste des recettes.

Vous devez donc créer une page `identification.php`, qui contiendra le formulaire, et adapter la page d'accueil `index.php`, qui doit maintenant inclure un formulaire de connexion et une condition sur l'affichage des recettes.

## Mobiliser les connaissances requises

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

On a préparé le terrain ensemble ; maintenant, vous savez tout ce qu'il faut pour réaliser le script. Vous êtes normalement capable de rédiger le code vous-même. Il ne fonctionnera probablement pas du premier coup, mais ne vous en faites pas : c'est le métier qui rentre !

## Correction

### La page identification.php

```
<?php
// Validation du formulaire
if (isset($_POST['courriel']) &&  isset($_POST['motPasse'])) {
  foreach ($utilisateurs as $utilisateur) {
    if ($utilisateur['courriel'] === $_POST['courriel'] &&
        $utilisateur['motPasse'] === $_POST['motPasse']) {
      $utilisateurConnecte = ['courriel' => $utilisateur['courriel']];
    } else {
      $messageErreur = sprintf('Les informations envoyées ne permettent
pas de vous identifier : (%s/%s)',
            $_POST['courriel'],
            $_POST['motPasse'] );
    }
  }
}
?>

<!--
   Si utilisateur/trice est non identifié(e), on affiche le formulaire
-->
<?php if(!isset($utilisateurConnecte)): ?>
<form action="index.php" method="post">
  <!-- si message d'erreur on l'affiche -->
  <?php if(isset($messageErreur)) : ?>
    <div class="alert alert-danger" role="alert">
      <?php echo $messageErreur; ?>
    </div>
  <?php endif; ?>
  <div class="mb-3">
    <label for="courriel" class="form-label">Email</label>
    <input type="email" class="form-control" id="courriel" name="courriel"
aria-describedby="aideCourriel" placeholder="vous@exemple.com">
    <div id="aideCourriel" class="form-text">L'adresse utilisée lors de la
création de compte.</div>
  </div>
  <div class="mb-3">
    <label for="motPasse" class="form-label">Mot de passe</label>
```

```
      <input type="password" class="form-control" id="motPasse"
name="motPasse">
    </div>
    <button type="submit" class="btn btn-primary">Envoyer</button>
</form>
<!--
      Si utilisateur/trice bien connecté(e) on affiche un message de succès
-->
<?php else: ?>
    <div class="alert alert-success" role="alert">
      Bonjour <?php echo $utilisateurConnecte['courriel']; ?> et bienvenue
sur le site !
    </div>
<?php endif; ?>
```

J'ai choisi un champ de type `password` puisqu'il s'agit d'un mot de passe. En dehors de ce point, il n'y a rien de particulier à noter.

## Coder la page index.php

Cette page va inclure le formulaire et limiter l'accès aux recettes.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Site de Recettes - Page d'accueil</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.
min.css"
      rel="stylesheet"
    >
</head>
    <body class="d-flex flex-column min-vh-100">
      <div class="container">

      <!-- Navigation -->
      <?php include_once('entete.php'); ?>

      <!-- Inclusion du formulaire de connexion -->
      <?php include_once('identification.php'); ?>
        <h1>Site de Recettes !</h1>

        <!-- Si l'utilisateur existe, on affiche les recettes -->
        <?php if(isset($utilisateurConnecte)): ?>
          <?php foreach(trouverRecettes($recettes) as $recette) : ?>
            <article>
              <h3><?php echo $recette['titre']; ?></h3>
              <div><?php echo $recette['recette']; ?></div>
              <i><?php echo afficherAuteur($recette['auteur'],
$utilisateurs); ?></i>
            </article>
```

```
          <?php endforeach ?>
        <?php endif; ?>
      </div>

      <?php include_once('pied_page.php'); ?>
    </body>
</html>
```


Le formulaire de connexion

Ce formulaire est-il si sécurisé que cela ?

Oui, honnêtement il l'est... du moins tant que vos utilisateurs ne choisissent pas de mots de passe trop simples à deviner. Un bon mot de passe 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 ».

## Aller plus loin

Pour l'instant, ce système de connexion n'est pas « persistant », c'est-à-dire que, si vous rechargez la page d'accueil ou revenez sur votre site plus tard, l'information de la connexion n'aura pas été conservée.

Nous verrons dans les prochains chapitres comment procéder pour conserver les informations de connexion d'un utilisateur.

## En résumé

- Il est recommandé de passer des informations en `POST` lorsqu'on conçoit un formulaire de connexion.
- Soit l'utilisateur est identifiable, c'est-à-dire que l'on retrouve une correspondance avec les identifiants fournis, auquel cas on autorise l'affichage du contenu.
- Soit l'utilisateur n'est pas identifiable et alors on affiche un message d'erreur.

# 16 Sessions et cookies

Précédemment, vous avez mis en place un système de connexion. Certes, il fonctionne, mais il ne conserve pas l'information.

Dans ce chapitre, vous allez apprendre à manipuler deux notions qui servent à conserver l'information entre deux pages PHP pour une durée plus ou moins longue. Bienvenue dans le monde des données persistantes, un sujet qui va nous occuper jusqu'à la fin de ce chapitre, vous disposerez alors d'un système de connexion entièrement fonctionnel.

## Les sessions

Jusqu'ici, nous étions parvenus à passer des variables de page en page à l'aide d'URL ou de formulaires. On sait ainsi envoyer d'une page à une autre les identifiants du visiteur mais, dès qu'on charge une autre page, ces informations sont « oubliées ». C'est là qu'interviennent les sessions : elles servent à conserver des variables sur toutes les pages de votre site.

### Fonctionnement

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

1. Un visiteur arrive sur votre site. On demande à créer une session pour lui. PHP génère alors un numéro unique. Celui-ci est souvent très long et écrit en hexadécimal, par exemple : `a02bbffc6198e6e0cc2715047bc3766f`. Ce numéro sert d'identifiant et est appelé « id de session » (ou `PHPSESSID`). PHP le transmet automatiquement à toutes les pages en utilisant généralement un *cookie*.
2. Une fois la session générée, on peut créer une infinité de variables de session pour nos besoins, que l'on stockera dans la supervariable `$_SESSION` ; par exemple,

`$_SESSION['nom']` qui contient le nom du visiteur, `$_SESSION['prenom']` qui contient le prénom, etc. Le serveur conserve ces variables même lorsque la page PHP a fini d'être générée ; quelle que soit la page de votre site, vous pourrez par exemple récupérer le nom et le prénom du visiteur via la superglobale `$_SESSION`.

3. Lorsque le visiteur se déconnecte de votre site, la session est fermée et PHP « oublie » alors toutes les variables de session que vous avez créées. Il est en fait difficile de savoir précisément quand un visiteur quitte votre site. En effet, lorsqu'il ferme son navigateur ou va sur un autre site, le vôtre n'en est pas informé. Soit le visiteur clique sur un bouton ***Déconnexion*** (que vous aurez créé) avant de s'en aller, soit on attend quelques minutes d'inactivité pour le déconnecter automatiquement (cas le plus fréquent) : on parle alors de *timeout*.

Pour activer ou détruire une session, vous devez connaître deux fonctions :

- `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.
- `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 *(timeout)*, mais vous pouvez aussi créer une page de 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 (avant même la balise `<!DOCTYPE>`). Si vous oubliez de lancer `session _ start()`, vous ne pourrez pas accéder à la variable superglobale `$ _ SESSION`.

## Mise en place d'une session

Je vous propose d'étudier un exemple concret et de mettre en place une session pour conserver l'information sur l'utilisateur connecté.

```
<?php
// identification.php

// On démarre la session AVANT d'écrire du code HTML
session_start();

// Validation du formulaire
if (isset($_POST['courriel']) && isset($_POST['motPasse'])) {
   foreach ($utilisateurs as $utilisateur) {
      //utilisateur/trice trouvé(e)
      if ($utilisateur['courriel'] === $_POST['courriel'] &&
           $utilisateur['motPasse'] === $_POST['motPasse']) {
        // Enregistrement de l'adresse de l'utilisateur en session
        $_SESSION['UTILISATEUR_CONNECTE'] = $utilisateur['courriel'] ;
      } else {
        $messageErreur = sprintf('Ces informations ne permettent pas de vous
identifier : (%s/%s)', $_POST['courriel'], $_POST['motPasse'] );
      }
```

```
    }
}
?>

<!--
    Si utilisateur/trice est non identifié(e), on affiche le formulaire
-->
<?php if(!isset($_SESSION['UTILISATEUR_CONNECTE'])): ?>
<form action="index.php" method="post">
  <!-- si message d'erreur on l'affiche -->
  <?php if(isset($messageErreur)) : ?>
    <div class="alert alert-danger" role="alert">
      <?php echo $messageErreur; ?>
    </div>
  <?php endif; ?>
  <div class="mb-3">
    <label for="courriel" class="form-label">Email</label>
    <input type="email" class="form-control" id="courriel" name="courriel"
aria-describedby="aideCourriel" placeholder="vous@exemple.com">
    <div id="aideCourriel" class="form-text">L'adresse utilisée lors de la
création de compte.</div>
  </div>
  <div class="mb-3">
    <label for="motPasse" class="form-label">Mot de passe</label>
    <input type="password" class="form-control" id="motPasse"
name="motPasse">
  </div>
  <button type="submit" class="btn btn-primary">Envoyer</button>
</form>

<!--
    Si utilisateur/trice bien connecté(e) on affiche un message de succès
-->
<?php else: ?>
  <div class="alert alert-success" role="alert">
    Bonjour <?php echo $_SESSION['UTILISATEUR_CONNECTE']; ?> et bienvenue
sur le site !
  </div>
<?php endif; ?>
```

En résumé, on peut créer des variables de session comme on crée des variables classiques, à condition de les écrire dans la superglobale `$_SESSION` et d'avoir lancé le système de sessions avec `session_start()`. Ces variables sont ainsi conservées de page en page pendant toute la durée de la présence de votre visiteur.

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

Concrètement, les sessions peuvent servir dans de nombreux cas sur votre site (pas seulement pour retenir un nom et un prénom). Voici quelques 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.

## Les cookies

Travailler avec des cookies revient à peu près à la même chose qu'avec des sessions, à quelques petites différences près. Voici ce que nous allons faire pour les découvrir :

1. Nous allons expliquer ce qu'est exactement un cookie parce que, si ça se trouve, il y en a qui croient en ce moment même que je vais parler de recettes de cuisine !
2. Ensuite, nous verrons comment **écrire un cookie** : c'est facile, si on respecte quelques règles.
3. Enfin, nous verrons comment **récupérer le contenu d'un cookie** : ce sera le plus simple.

### Qu'est-ce qu'un cookie ?

Un cookie, c'est un petit fichier qu'on enregistre sur l'ordinateur du visiteur. Il contient du texte et permet de « retenir » des informations. Par exemple, vous pouvez enregistrer dans un cookie le pseudo du visiteur pour le relire la prochaine fois qu'il viendra sur votre site.

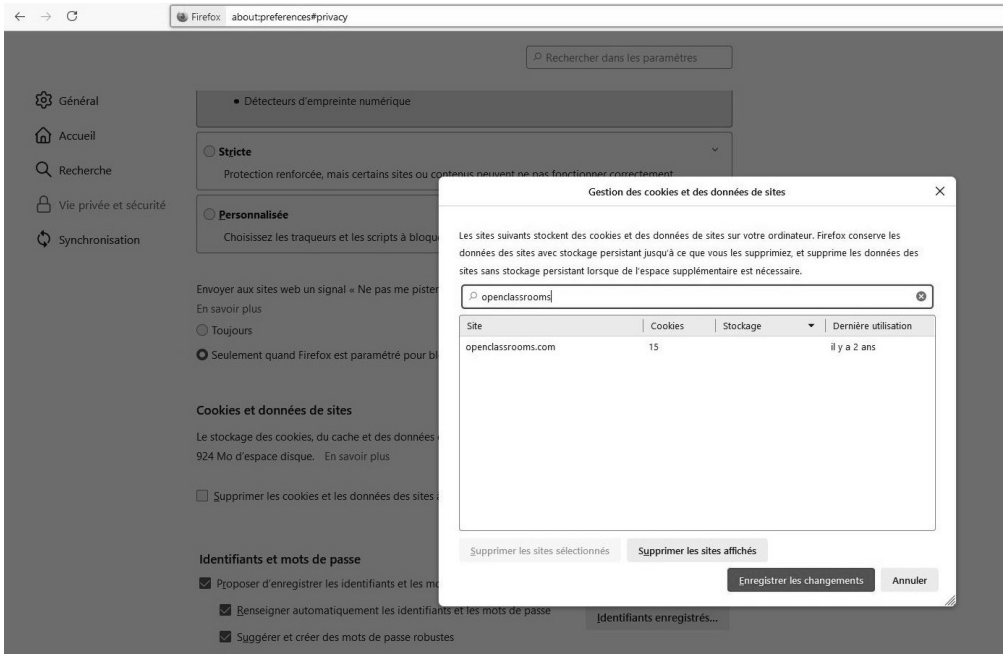
Parfois les cookies ont une mauvaise image. On fait souvent l'erreur de penser que les cookies sont « dangereux ». Non, ce ne sont pas des virus, juste de petits fichiers de texte. Au pire, un site marchand peut retenir que vous aimez les appareils photos numériques et adapter ses publicités en conséquence,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

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

Où sont stockés les cookies sur mon disque dur ?

Cela dépend de votre navigateur web. Généralement, on ne touche pas directement à ces fichiers, mais on peut afficher à l'intérieur du navigateur la liste des cookies qui sont stockés. On peut choisir de les supprimer à tout moment.

Si vous utilisez Mozilla Firefox, allez dans le menu ***Paramètres>Vie privée et sécurité>Cookies et données de site*** et cliquez sur ***Gérer les données***. Vous obtenez alors la liste et la valeur de tous les cookies stockés, comme sur la figure suivante.




Cookies sous Mozilla Firefox

Les cookies sont classés par site web. Chaque site web peut écrire plusieurs cookies. Ces derniers sont des informations temporaires qu'on stocke sur l'ordinateur des visiteurs. La taille est limitée à quelques kilo-octets : vous ne pouvez pas stocker beaucoup d'informations à la fois, mais c'est en général suffisant.

## Écrire un cookie

Comme une variable, un cookie a un nom et une valeur. Par exemple, le cookie utilisateur s'appellerait chez moi `UTILISATEUR_CONNECTE`.

Pour écrire un cookie, on utilise la fonction PHP `setcookie()`. On lui donne en général trois paramètres, dans l'ordre suivant :

1. le nom du cookie (ex : `UTILISATEUR_CONNECTE`) ;
2. sa valeur (ex : `utilisateur@exemple.com`) ;
3. sa date d'expiration, sous forme de *timestamp* (ex : `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 sa valeur actuelle, 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.

Si vous voulez supprimer le cookie dans un an, il vous faudra donc écrire : `time()+365*24*3600`.

## Sécuriser un cookie avec les propriétés httpOnly et secure

Configurons les options `httpOnly` et `secure` sur le cookie. Sans entrer dans les détails, cela le rendra inaccessible en JavaScript sur tous les navigateurs qui supportent cette option (c'est le cas de tous les navigateurs récents.). Cette option réduit drastiquement les risques de faille XSS sur votre site, au cas où vous auriez oublié d'utiliser `htmlspecialchars` à un moment.

Voici comment créer un cookie **sécurisé** :

```
<?php
// retenir l'adresse de la personne connectée pendant 1 an
setcookie(
   'UTILISATEUR_CONNECTE',
   'utilisateur@exemple.com',
   [
      'expires' => time() + 365*24*3600,
      'secure' => true,
      'httpOnly' => true,
   ]
);
?>
```

En procédant de cette façon, vous diminuez le risque que l'un de vos utilisateurs se fasse voler le contenu d'un cookie à cause d'une faille XSS.

Ne placez JAMAIS le moindre code HTML avant d'utiliser `setcookie()` ! Les gens qui ont des problèmes avec cette fonction ont la plupart du temps commis cette erreur.

## Afficher et récupérer 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 de tableau.

De ce fait, pour ressortir l'adresse du visiteur inscrite dans un cookie, il suffit d'écrire : `$_COOKIE['UTILISATEUR_CONNECTE']`. Cela nous donne un code PHP tout simple pour afficher de nouveau cette adresse :

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

Comme vous le voyez encore une fois, le gros avantage est que les superglobales sont accessibles partout. Notez toutefois que, si le cookie n'existe pas, la variable superglobale n'existe pas ; il faut donc faire appel à `isset`.

Les cookies viennent du visiteur. Comme toute information qui vient du visiteur, elle n'est pas sûre. N'importe qui peut créer des cookies et envoyer ainsi de fausses informations à votre site.

## Modifier un cookie existant

Vous vous demandez peut-être comment modifier un cookie déjà existant ? Il faut refaire appel à `setcookie()` **en gardant le même nom de cookie**, ce qui « écrasera » l'ancien.

Par exemple, si c'est Laurène Castor qui se connecte au site, j'écrirai :

```
<?php
setcookie(
   'UTILISATEUR_CONNECTE',
   'laurene.castor@exemple.com',
   [
      'expires' => time() + 365*24*3600,
      'secure' => true,
      'httponly' => true,
   ]
);
?>
```

Notez que, dans ce cas, le temps d'expiration du cookie est remis à zéro pour un an.

## En résumé

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

# Quatrième partie

# Stocker des informations dans une base de données

En PHP, on peut difficilement se passer d'une base de données. Cet outil incontournable sert à enregistrer des données de façon efficace et organisée.

Tout ce que vous voulez **enregistrer** sur votre site va se retrouver stocké dans une base de données : liste des membres, liste des recettes, commentaires, etc.

# 17 Travailler avec des bases de données

Pour l'instant, vous avez découvert le fonctionnement du langage PHP, mais vous ne vous sentez probablement pas encore capable de créer de vrais sites web avec ce que vous avez appris. C'est parfaitement normal, car il vous manque un élément crucial : la **base de données**.

Une base de données permet d'enregistrer des données de façon organisée et hiérarchisée. Certes, vous connaissez les variables, mais celles-ci restent en mémoire seulement le temps de générer la page. Vous pourriez aussi écrire dans des fichiers, mais cela devient vite très compliqué dès que vous avez beaucoup de données à enregistrer.

Or, il va bien falloir stocker quelque part la liste de vos membres, les recettes, les commentaires sous les recettes… Les bases de données constituent le meilleur moyen de réaliser cela de façon simple et propre. Nous allons les étudier durant toute cette partie du cours.

## Le langage SQL et les bases de données

La base de données (BDD) est un fichier où sont enregistrées des informations. Ce n'est pas comme un fichier texte. Ce qui est très important ici, c'est que ces informations sont toujours **classées**. Et c'est cela qui fait que la BDD est si pratique : c'est un moyen simple de ranger des informations.

Et si je préfère rester désordonné ? Si je n'ai pas envie de classer mes informations ? Est-on obligé de classer chaque information qu'on enregistre ?

C'est un peu ce que je me disais au début… Classer certaines choses, d'accord, mais il me semblait que je n'en aurais besoin que très rarement. Grave erreur ! Comme vous

allez le constater, 99 % du temps, on range ses informations dans une base de données. Pour le reste, on peut les enregistrer dans un fichier… mais, quand on a goûté aux bases de données, on peut difficilement s'en passer ensuite.

Imaginez par exemple une armoire, dans laquelle chaque dossier est à sa place. Quand tout est rangé, il est beaucoup plus facile de retrouver un objet, n'est-ce pas ? Eh bien là, c'est pareil : en classant les informations que vous collectez (concernant vos visiteurs, par exemple), il vous sera très facile de récupérer plus tard ce que vous cherchez.

Nous allons utiliser le SGBD (système de gestion de bases de données) 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 possible de choses à apprendre si vous choisissez de changer de SGBD.

## Donner des ordres au SGBD en langage SQL

Vous allez devoir communiquer avec le SGBD pour lui donner l'ordre de récupérer ou d'enregistrer des données. Pour lui « parler », on utilise le langage SQL.

La bonne nouvelle, c'est que ce langage est un standard, c'est-à-dire que, quel que soit le SGBD que vous utilisez, vous vous servirez du langage SQL. La mauvaise, c'est qu'il existe quelques petites variantes d'un SGBD à l'autre, mais cela concerne généralement les commandes les plus avancées.

Comme vous vous en doutez, il va falloir apprendre le langage SQL pour travailler avec les bases de données. Ce langage n'a rien à voir avec le PHP, mais il est nécessaire.

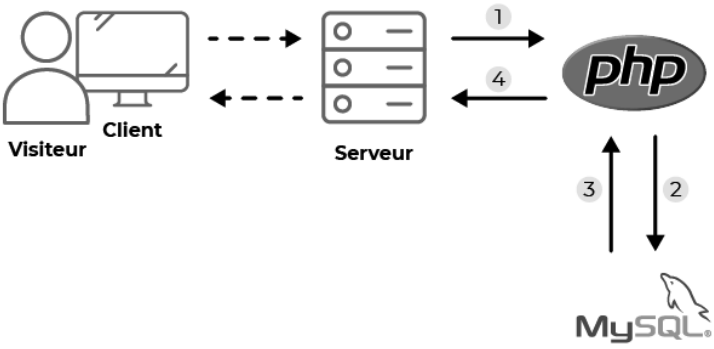
Voici un exemple de commande en langage SQL, pour vous donner une idée :

```
SELECT id, auteur, message, datemsg FROM livreor ORDER BY datemsg DESC
```

Le principal objectif de cette partie du cours sera d'apprendre les instructions nécessaires à écrire en PHP pour réaliser des requêtes en base de données, ainsi que les bases du SQL.

## PHP fait la jonction entre vous et MySQL

Pour compliquer un petit peu l'affaire (sinon, ce n'est pas rigolo), on ne va pas parler directement à MySQL. Seul PHP peut le faire. C'est donc ce dernier qui va faire l'intermédiaire entre vous et MySQL. On devra demander à PHP : « dis à MySQL de faire ceci. »




Communication entre PHP et MySQL

Voici ce qui peut se passer lorsque le serveur reçoit une demande d'un client qui veut poster un message :

1. Le serveur utilise toujours PHP et lui fait donc passer le message.
2. PHP exécute les actions demandées et se rend compte qu'il a besoin de MySQL. En effet, le code PHP contient à un endroit l'instruction : « va demander à MySQL d'enregistrer ce message ». Il fait donc passer le message à MySQL.
3. MySQL réalise 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.

Maintenant que nous avons fait les présentations, nous allons découvrir comment est organisée une base de données. Bien en comprendre l'organisation est en effet absolument indispensable.

## Structure d'une base de données

Avec les bases de données, il faut utiliser un vocabulaire **précis**. Heureusement, vous ne devriez pas avoir trop de mal à vous en souvenir, vu qu'on va se servir d'une image : celle d'une armoire. Je vous demande d'imaginer 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. Par exemple, on peut imaginer un tiroir qui contient les pseudonymes et informations concernant vos visiteurs, un autre qui contient les recettes…
- C'est dans une table que 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**.

Par exemple, le tableau suivant vous montre à quoi peut ressembler le contenu d'une table appelée `utilisateurs`.

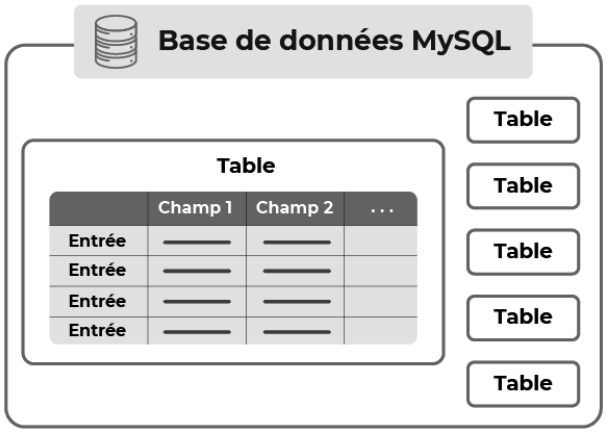
**Contenu de la table utilisateurs**

| ID | NOM | COURRIEL | AGE | MOTPASSE |
|---|---|---|---|---|
| 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 |
| … | … | … | … | |

Chaque ligne est une entrée. Ici, il n'y en a que quatre, mais une table peut très bien en contenir 100, 1 000 ou même 100 000.

Très souvent, on crée un champ `Numero` ou `ID` (identifiant). Même si ce n'est pas obligatoire, nous verrons plus loin qu'il est très pratique de numéroter ses entrées.

Pour finir, voici l'indispensable schéma pour que tout soit clair.




Organisation d'une base de données MySQL

La base de données contient plusieurs tables (on peut en ajouter autant qu'on veut). Pour vous donner des exemples concrets, voici le nom des tables que vous allez créer pour compléter votre site de partage de recettes :

- `utilisateurs` : tous les comptes utilisateurs ;
- `recettes` : toutes les recettes ;

• `commentaires` : tous les commentaires liés aux recettes.

Si quelque chose ne vous paraît pas clair, si vous avez l'impression de mélanger un peu bases, tables, champs ou entrées, relisez de nouveau cette partie. Il faut que vous soyez capable de reproduire le schéma tout seul sur un bout de papier.

## Enregistrer les données

Avant de terminer le chapitre, voici une question qu'on se pose fréquemment.

Ils sont bien jolis ces tableaux et ces schémas, ces bases, ces champs... Mais concrètement, où MySQL enregistre-t-il les données ?

En fait, tout ce que je viens de vous montrer, c'est une façon de « visualiser » la chose. Il faut que vous imaginiez que la base de données gère les informations sous forme de tableaux, parce que c'est la meilleure représentation qu'on peut s'en faire. Concrètement toutefois, quand MySQL enregistre des informations, il les écrit **dans des fichiers**.

Ces fichiers sont quelque part sur votre disque dur. Par exemple, avec MySQL sous Windows, si vous utilisez WAMP/MAMP, vous devriez les trouver dans `C:\wamp\mysql\data`. Toutefois, **il ne faut jamais les ouvrir et encore moins les modifier directement**. Il faut toujours parler avec MySQL qui va se charger d'en extraire et modifier les informations.

C'est justement le gros avantage de la base de données : le rangement des informations n'est pas contraignant. Vous demandez à MySQL de vous sortir tous les commentaires d'une recette enregistrés de février à juillet : il va lire dans ses fichiers et il vous fournira les réponses. Vous vous contentez de « dialoguer » avec MySQL. Il se charge quant à lui du « sale boulot », c'est-à-dire de ranger vos données dans ses fichiers.

## En résumé

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

# 18 phpMyAdmin

Dans ce chapitre, vous allez créer et mettre en place la base de données pour votre site de recettes.

Il existe plusieurs façons de communiquer avec MySQL : de l'invite de commande jusqu'à des logiciels accessibles par le navigateur. Ici, vous allez utiliser **phpMyAdmin**, un des outils les plus connus pour manipuler une base de données MySQL.

phpMyAdmin est livré avec MAMP et XAMP, vous allez donc pouvoir vous en servir tout de suite.

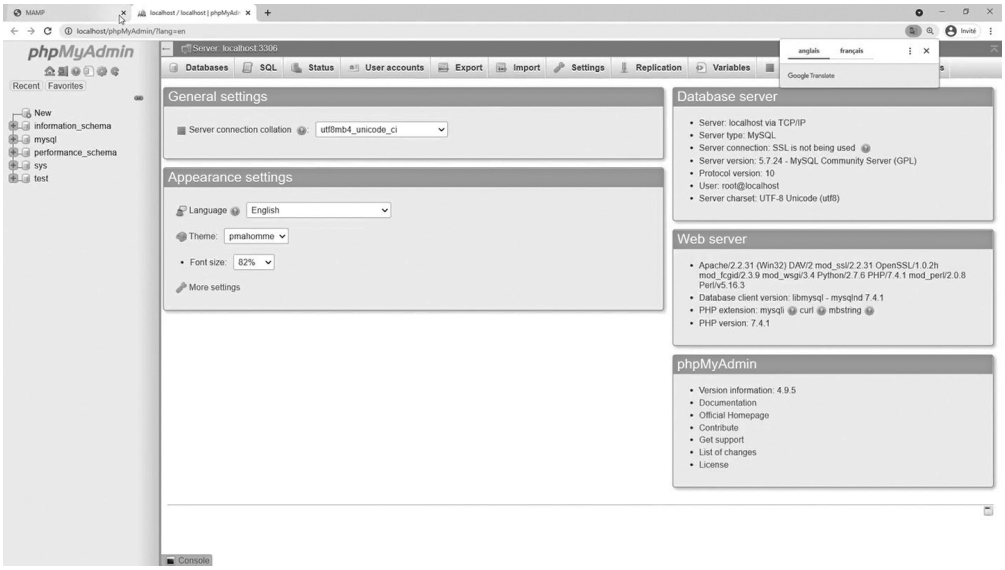
## Créer une table pour les recettes

Pour créer cette table, nous allons associer chacune des propriétés de nos recettes à un champ de la table et lui associer un type de données. Ensuite, nous devons définir un champ qui servira d'identifiant unique (un peu comme un numéro de Sécurité Sociale), de sorte à pouvoir retrouver chaque recette.

Tout d'abord, démarrez MAMP puis, sur la page d'accueil, cliquez sur ***Tools>phpMyAdmin***.

phpMyAdmin n'est pas un programme mais un ensemble de pages PHP toutes prêtes dont on se sert pour gagner du temps.

L'accueil de phpMyAdmin ressemble à la figure suivante.



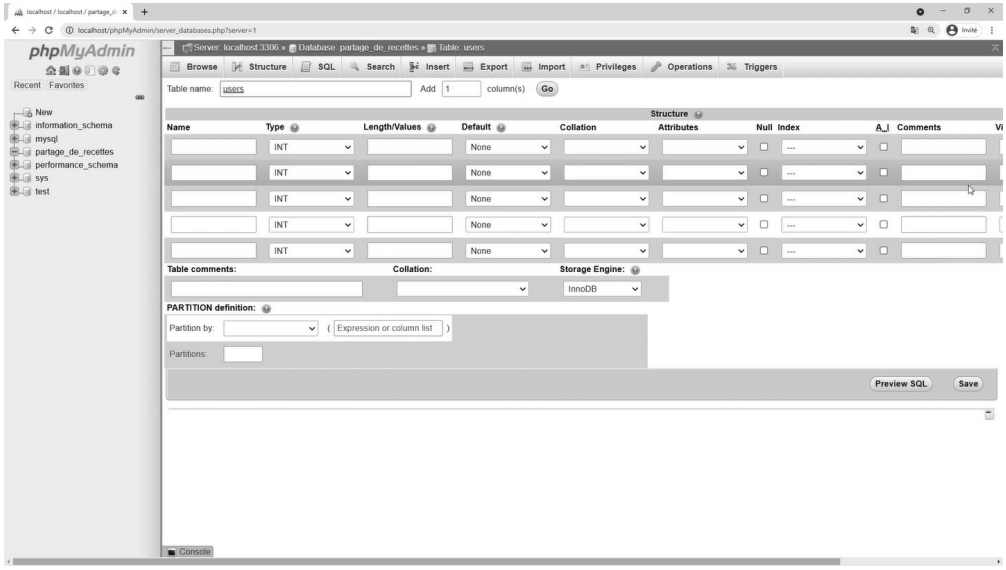

Accueil de phpMyAdmin

Cliquez sur ***New*** (menu de gauche) pour créer la base de données `partage_de_recettes`, puis validez en cliquant sur ***Create***.


La base de test a été créée, elle est vide.

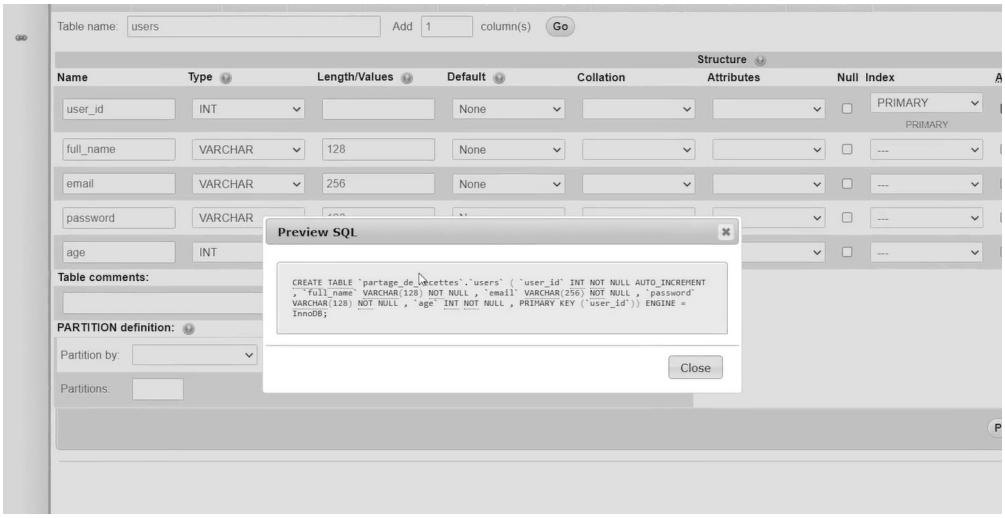
Il nous faut maintenant créer des tables. Commençons par la table `utilisateurs`, composée de cinq colonnes (clé primaire, nom/prénom, adresse e-mail, mot de passe et âge) :

- `u_id` : un entier (***INT***) identifiant chaque utilisateur de façon unique (cochez la case ***A_I – AUTO INCREMENT*** – pour que cette valeur soit automatique gérée par le logiciel) ;
- `nom` : une chaîne de type ***VARCHAR*** et de longueur `128` ;
- `courriel` : une chaîne de type ***VARCHAR*** et de longueur `256` ;
- `motPasse` : une chaîne de type ***VARCHAR*** et de longueur `128` ;
- `age` : un entier (***INT***).




Créer une table MySQL.

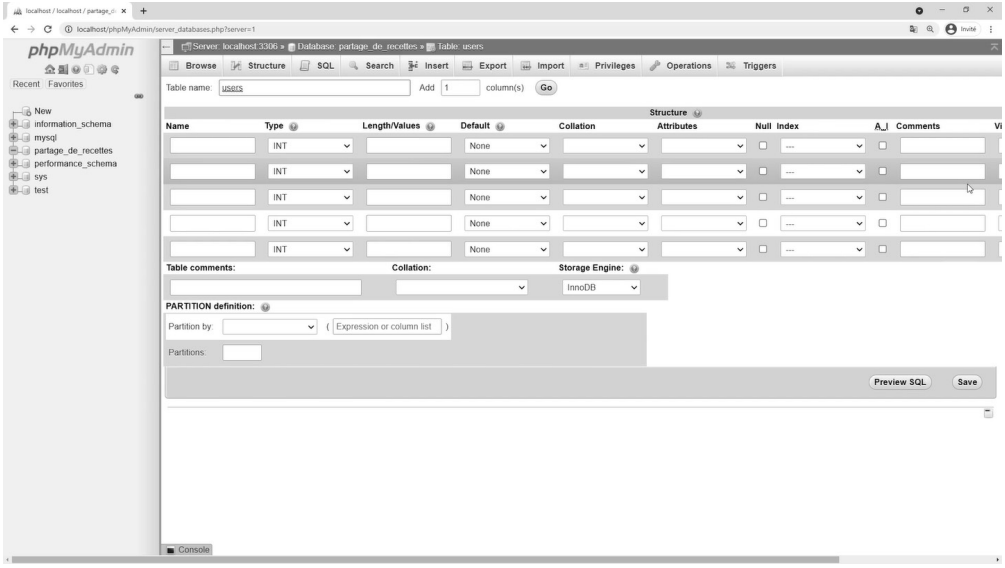
Lorsque vous aurez fini de définir les champs, cliquez sur ***Preview SQL***, en bas à droite. Une fenêtre vous affiche alors la requête destinée à créer la table. Même si le propos ici n'est pas de détailler le langage SQL, il est intéressant de regarder comment cette requête est formulée.




Requête correspondant à la table créée

Cliquez sur ***Save*** ; ça y est, vous avez créé votre première table et phpMyAdmin vous affiche toutes ses caractéristiques.

Ensuite, cliquez sur l'onglet ***Insert***, pour ajouter un nouvel enregistrement, c'est-à-dire un nouvel utilisateur. Complétez les champs à votre guise, sauf `u_id`, qui est automatiquement incrémenté.




Ajouter un nouvel utilisateur.

Avant d'aller plus loin, je voudrais revenir un peu plus en détail sur les types de champs et les index, notamment l'index ***PRIMARY*** que l'on a utilisé.

### Les types de champs MySQL

Alors que PHP ne propose que quelques types de données que l'on connaît bien maintenant (`int`, `string`, `bool`…), MySQL en propose de très nombreux autres. Dans la pratique cependant, vous n'aurez besoin de jongler qu'entre les quatre types suivants :

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

Cela couvrira 99 % de vos besoins. Avec l'expérience, vous apprendrez à optimiser vos bases de données et découvrirez l'intérêt des autres types proposés par MySQL.

### Les clés primaires

Toute table doit posséder un champ qui joue le rôle de clé primaire. Cette dernière permet d'identifier de manière unique une entrée dans la table. En général, on utilise le champ `id` comme clé primaire.

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*** afin que ce champ gère lui-même automatiquement les nouvelles valeurs (1, 2, 3, 4…).

## Modifier une table

Sachez qu'il est possible d'ajouter ou de supprimer des champs à tout moment. Ce n'est pas parce que votre table a été créée qu'elle reste figée. Il existe des options pour renommer les champs, les supprimer, en ajouter, etc.

Pour modifier la structure d'une table, cliquez sur cette dernière, puis sur l'onglet ***Structure***. Les boutons ***Drop*** et ***Change*** de chaque champ servent respectivement à le supprimer et à en modifier les caractéristiques.

Pour ajouter un champ, le bouton ***Add*** est plus discret, en bas à gauche de la liste des champs. Vous devez préciser combien de colonnes supplémentaires vous voulez créer, où vous souhaitez les placer par rapport à l'existant puis cliquer sur ***Go***. Cela vous amène sur un formulaire semblable à celui de la création de la table. Définissez vos nouveaux champs puis cliquez sur ***Save***.

Si vous ajoutez un champ *après* avoir créé des enregistrements, la structure de ceux-ci sera bien modifiée, mais il vous faudra les éditer pour donner une valeur au nouveau champ.

Si vous souhaitez modifier une valeur d'un enregistrement existant, cliquez sur le bouton ***Edit*** de ce dernier pour rentrer dans le mode d'édition.

Vais-je devoir passer par phpMyAdmin à chaque fois que je veux ajouter ou supprimer un élément ?

Non, bien sûr que non. Vous allez apprendre à le faire en PHP dans les chapitres suivants.

Il nous reste encore à découvrir deux des nombreuses fonctionnalités offertes par phpMyAdmin et nous aurons terminé notre tour d'horizon de cet outil : l'import et l'export de base de données.

## Importer et exporter des données

Vous allez ici vous intéresser à l'onglet ***Import*** de phpMyAdmin, dont le principal intérêt est de créer une base de données entière avec tables et données.

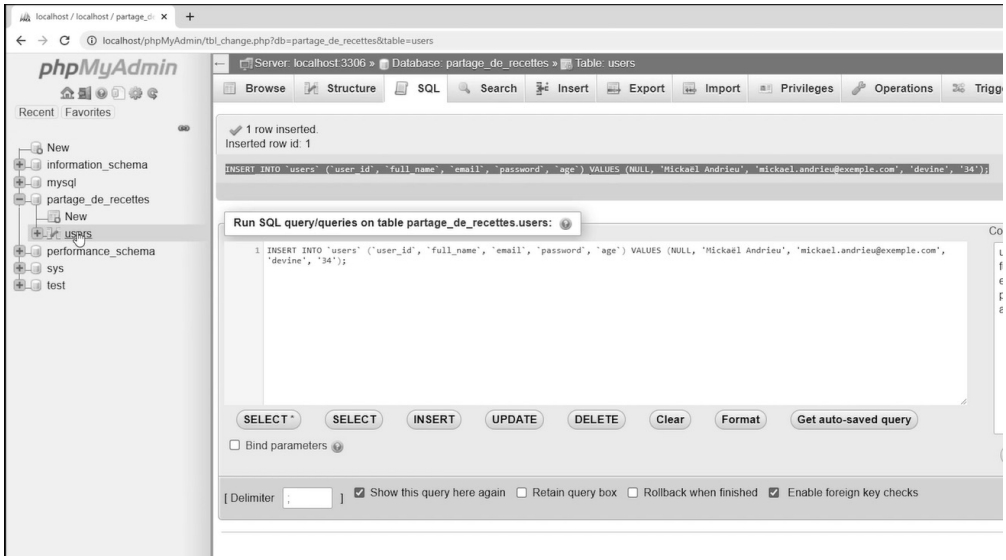
Vous avez vu, en cliquant sur le bouton ***Preview SQL***, que toute la construction d'une base de données, sa structure aussi bien que son contenu, était définie par des requêtes SQL.

Dans les fichiers à télécharger (voir plus loin), vous trouverez `creation_base.sql`, qui contient toutes les instructions pour construire la base de données de notre projet. Il vous suffit de l'importer dans phpMySQL : sélectionnez-le en cliquant sur ***Choisir un fichier***, puis validez en cliquant sur ***Go*** en bas de la page.

À l'issue de cette manipulation, vous disposerez de la base `partage_de_recettes` composée de ses deux tables `utilisateurs` et `recettes`, vides d'enregistrements pour l'instant.

C'est à votre tour d'essayer. Pour télécharger le fichier d'import de la base de notre projet fil rouge, cliquez sur le lien suivant : *https://github.com/OpenClassrooms-Student-Center/Concevez-votre-site-web-avec-PHP-MySQL/blob/main/P4/P4C2/sql/creation_base.sql*

Explorez maintenant l'onglet ***Exporter*** de phpMyAdmin. C'est ici que vous allez récupérer sur votre disque dur votre base de données sous la forme d'un fichier texte `.sql`. Ce fichier dit à MySQL **comment recréer votre base de données** (avec des requêtes en langage SQL), non seulement sa structure, mais aussi tout son contenu. Il est tout à fait lisible.




Requête SQL permettant de recréer votre base de données.

On peut s'en servir pour deux choses :

- **Transmettre la 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 le SGBD en ligne de votre hébergeur. Le fichier `.sql` que vous allez générer vous permettra de **reconstruire** la base de données à l'identique.
- **Faire une copie de sauvegarde de la base de données :** on ne sait jamais, si vous commettez une bêtise ou si quelqu'un réussit à détruire toutes les informations sur votre site (dont la base de données), vous serez bien content d'avoir une copie de secours sur votre disque dur !

## En résumé

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

# 19 Accéder aux données avec PDO

Dans ce chapitre, nous retournons à nos pages PHP. À partir de maintenant, nous allons apprendre à communiquer avec une base de données via PHP.

À la fin de ce chapitre, vous aurez obtenu une nouvelle version de la page d'accueil du site, cette fois créée à l'aide de PHP et MySQL.

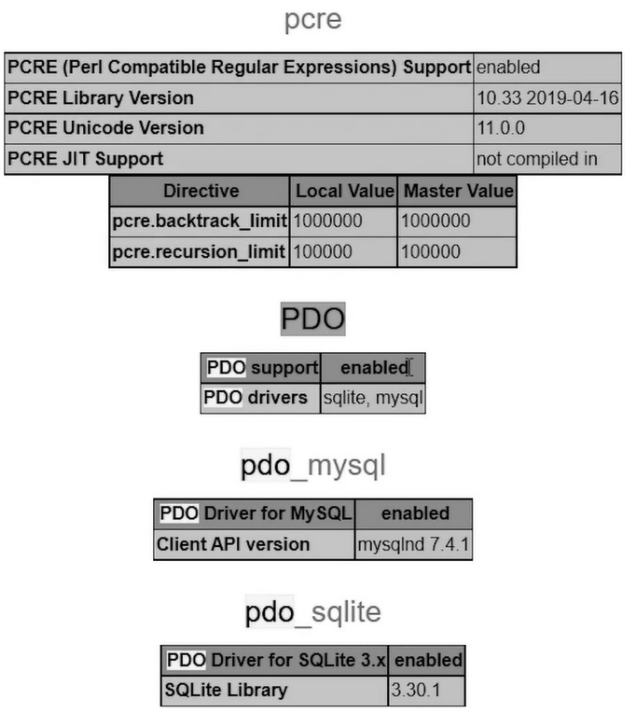
## Se connecter à la base de données en PHP

Pour pouvoir travailler avec la base de données en PHP, on doit d'abord s'y connecter. Il va donc falloir que PHP s'authentifie : on dit qu'il *établit une connexion* avec MySQL. Ensuite, vous pourrez réaliser toutes les opérations que vous voudrez sur votre base de données.

Nous allons ici utiliser une extension PHP appelée PDO *(PHP Data Objects)*, qui est fournie avec le langage (en français ***les fonctions PDO sont à votre disposition***) mais qu'il vous faudra parfois activer.

### Activer PDO

Lancez MAMP, puis allez dans l'onglet ***phpInfo***. Deux informations sont importantes pour nous. Repérez tout d'abord le chemin où trouver le fichier de configuration de PHP, `php.ini`. Ensuite, tapez ***Ctrl+F*** (ou ***Pomme+F***) pour rechercher le terme « PDO ». Contrôlez si l'option ***PDO support*** a bien pour valeur ***enabled***.



pcre

| PCRE (Perl Compatible Regular Expressions) Support | enabled |
|---|---|
| PCRE Library Version | 10.33 2019-04-16 |
| PCRE Unicode Version | 11.0.0 |
| PCRE JIT Support | not compiled in |

| Directive | Local Value | Master Value |
|---|---|---|
| pcre.backtrack_limit | 1000000 | 1000000 |
| pcre.recursion_limit | 100000 | 100000 |

PDO

| PDO support | enabled |
|---|---|
| PDO drivers | sqlite, mysql |

pdo_mysql

| PDO Driver for MySQL | enabled |
|---|---|
| Client API version | mysqlnd 7.4.1 |

pdo_sqlite

| PDO Driver for SQLite 3.x | enabled |
|---|---|
| SQLite Library | 3.30.1 |

Vérifiez que l'extension PDO est activée.

Si ce n'est pas le cas, éditez le fichier `php.ini` et recherchez-y à nouveau le terme « PDO ». Si vous trouvez un point-virgule devant l'extension `php_pdo_mysql.dll`, cela signifie que la ligne est mise en commentaire, donc PDO désactivé ; supprimez le point-virgule, sauvegardez `php.ini`, puis relancez MAMP.

```
;extension=php_pdo_firebird.dll
;extension=php_pdo_mssql.dll
extension=php_pdo_mysql.dll
;extension=php_pdo_oci.dll
;extension=php_pdo_odbc.dll
```

Je vous recommande de passer la ligne `display_errors` à `On` pour que les erreurs s'affichent ; cela va grandement vous aider :

```
display_errors = On
```

Si vous êtes sous Linux et utilisez XAMPP, recherchez la ligne qui commence par `pdo_mysql.default_socket` et complétez-la comme ceci :

```
pdo_mysql.default_socket = /opt/lampp/var/mysql/mysql.sock
```

Enregistrez le fichier puis redémarrez PHP. Il suffit pour cela de relancer votre logiciel favori (MAMP, XAMPP).

## Se connecter à MySQL avec PDO

Maintenant que nous sommes certains que PDO est activé, nous pouvons nous connecter à MySQL. Nous aurons besoin des quatre informations suivantes :

- **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, indiquez 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 `partage_de_recettes`. Nous l'avons 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.

Sur MAMP, l'identifiant et le mot de passe ont la même valeur : `root`.

Voici donc l'instruction PDO pour vous connecter à la base `partage_de_recettes` :

```
<?php
// Souvent on identifie cet objet par la variable $bdd
$bdd = new PDO(
   'mysql:host=localhost;dbname=partage_de_recettes;charset=utf8',
   'root',
   'root'
);
?>
```

Je ne comprends rien à ce code, c'est normal ?

Oui, il faut reconnaître qu'il contient quelques nouveautés. En effet, PDO est ce qu'on appelle une extension **orientée objet**. C'est une façon de programmer un peu différente des fonctions classiques qu'on a appris à utiliser jusqu'ici.

Pour l'instant, je vous invite à réutiliser les codes que je vous propose en suivant mes exemples. Vous comprendrez les détails de leur mode de fonctionnement un peu plus tard.

Cette ligne de code crée la connexion en indiquant dans l'ordre les paramètres suivants :

- le nom d'hôte (`localhost`) ;

- la base de données (`partage_de_recettes`) ;
- l'identifiant (`root`) ;
- le mot de passe (rappel : sous MAMP, c'est `root`).

Lorsque votre site sera en ligne, vous aurez sûrement un nom d'hôte différent ainsi qu'un identifiant et un mot de passe comme ceci :

```
<?php
$bdd = new PDO('mysql:host=sql.hebergeur.com;dbname=partage_de_
recettes;charset=utf8', 'pierre.durand', 's3cr3t');
?>
```

Lorsque vous enverrez votre site sur le Web, il faudra donc penser à changer cette ligne pour l'adapter à votre hébergeur et modifier les informations en conséquence.

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.

## Tester la présence d'erreurs

Si vous avez renseigné les bonnes informations (noms de l'hôte et de la base, identifiant et mot de passe), rien ne devrait s'afficher à l'écran. À l'inverse, s'il y a une erreur (vous vous êtes trompé de mot de passe ou de nom de base de données, par exemple), PHP risque d'afficher toute la ligne qui pose l'erreur, ce qui inclut le mot de passe !

Vous ne voudrez sans doute pas que vos visiteurs puissent voir le mot de passe si une erreur survient lorsque votre site est en ligne. Il est préférable de **traiter** l'erreur. En cas d'erreur, PDO renvoie ce qu'on appelle une **exception** qui permet de « capturer » l'erreur. Voici la solution que je vous propose :

```
<?php
try {
   $bdd = new PDO('mysql:host=localhost;dbname=partage_de_recettes;charset=
utf8', 'root', 'root');
}
catch (Exception $e) {
   die('Erreur : ' . $e->getMessage());
}
?>
```

Voilà encore un code un peu nouveau pour vous. Sans trop entrer dans le détail, il faut savoir que PHP essaie d'exécuter les instructions à l'intérieur du bloc `try`. S'il rencontre une erreur, il entre 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`. Votre page PHP ne devrait donc rien afficher pour le moment.

Ouh là ! Tout ça semble bien compliqué, je n'y comprends pas grand-chose !

Ce n'est pas grave du tout ! J'insiste sur le fait que PDO nous fait utiliser des fonctionnalités de PHP qu'on n'a pas étudiées jusqu'à présent (programmation orientée objet, exceptions). Contentez-vous pour le moment de réutiliser les codes que je vous propose et n'ayez crainte.

## Récupérer les données

Maintenant, nous allons apprendre à lire des informations dans la base de données. Nous verrons dans le chapitre suivant comment ajouter et modifier des données. L'objectif ici est de récupérer la liste des recettes qui étaient au départ dans une variable sous forme de tableau associatif, mais qui sont maintenant stockées dans votre base de données.

Nous allons donc nous adresser à MySQL en SQL. Pour cela, nous allons construire ce qu'on appelle une **requête** pour demander à MySQL de nous dire tout ce que contient la table `recettes`.

## Construire notre première requête SQL

Voici la première requête SQL que nous allons utiliser :

```
SELECT * FROM recettes
```

Cela peut se traduire par : « Prendre tout ce qu'il y a dans la table `recettes`. »

Analysons chaque terme 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 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 `titre` et `auteur`, il faudra taper :

```
SELECT titre, auteur FROM recettes
```

  Si vous voulez prendre tous les champs, tapez `*`. Cette petite étoile peut se traduire par « tout », autrement dit : « 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 celui de la table.
- `recettes` : c'est le nom de la table dans laquelle il faut aller chercher.

Lançons la requête à l'aide de l'objet PDO :

```
<?php
$requetePreparee = $bdd->prepare('SELECT * FROM recettes');
?>
```

## Afficher le résultat d'une requête

Le problème, c'est que `$requetePreparee` contient quelque chose d'inexploitable directement : un objet `PDOStatement` (*https://www.php.net/manual/fr/class.pdostatement.php*). Il contient la requête SQL et, quand il aura exécuté cette dernière, les informations récupérées en base de données. Il faut demander à cet objet d'exécuter la requête, puis de nous fournir les données dans un format que nous saurons exploiter, c'est-à-dire un tableau PHP.

```
<?php
$requetePreparee->execute();
$recettes = $requetePreparee->fetchAll();
?>
```

*fetch*, en anglais, signifie « va chercher ».

Une fois que vous avez récupéré les données sous forme d'un tableau PHP, vous en revenez à ce que vous connaissez déjà. Résumons tout ce que vous venez d'apprendre, de la connexion via PDO à l'affichage du résultat de la requête :

```
<?php
try {
   // On se connecte à MySQL
   $bdd = new PDO('mysql:host=localhost;dbname=partage_de_recettes;charset=
utf8', 'root', 'root');
}
catch(Exception $e) {
   // En cas d'erreur, on affiche un message et on arrête tout
   die('Erreur : '.$e->getMessage());
}

// Si tout va bien, on peut continuer

// On récupère tout le contenu de la table recettes
$requeteSQL = 'SELECT * FROM recettes';
$requetePreparee = $bdd->prepare($requeteSQL);
$requetePreparee->execute();
$recettes = $requetePreparee->fetchAll();

// On affiche chaque recette une à une
foreach ($recettes as $recette) {
```

```
?>
   <p><?php echo $recette['auteur']; ?></p>
<?php
}
?>
```

### Afficher seulement le contenu de quelques champs

Vous devriez maintenant être capable d'afficher ce que vous voulez. Personne ne vous oblige à afficher tous les champs. Par exemple, si j'avais juste voulu lister le nom des recettes et celui des auteurs, j'aurais utilisé la requête SQL suivante :

```
$requeteSQL = 'SELECT titre, auteur FROM recettes';
```

Vous récupérez seulement les informations dont vous avez besoin, ce qui est plus performant.

## Filtrer les données

Rappelez-vous votre objectif : lister les recettes qui sont activées.

En modifiant vos recettes SQL, il est facile de filtrer et trier les données. Nous allons nous intéresser ici aux mots-clés suivants du langage SQL :

- `WHERE` ;
- `ORDER BY` ;
- `LIMIT`.

Plus vous avez de conditions et plus la requête devient complexe. Avant d'écrire le code PHP, vous pouvez déjà vérifier que la requête SQL est correcte en la testant dans l'onglet ***SQL*** de phpMyAdmin.

Il faut utiliser les mots-clés dans l'ordre indiqué : `WHERE` puis `ORDER BY` puis `LIMIT`, sinon MySQL ne comprendra pas votre requête.

### WHERE

phpMyAdmin propose un onglet SQL où nous pouvons vérifier la validité de nos requêtes.

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

Puisque vous souhaitez récupérer uniquement les recettes dont le champ `estValide` vaut `1`, la requête initiale devra être complétée par `WHERE estValide = 1` :

```
$requeteSQL = 'SELECT * FROM recettes WHERE estValide = 1';
```

Pour délimiter les chaînes de caractères, on doit les placer entre apostrophes. En revanche, ce n'est pas nécessaire pour les nombres.

Il est par ailleurs possible de combiner plusieurs conditions. Par exemple, si vous souhaitez obtenir toutes les recettes valides de Mickaël, il vous faudra combiner les deux conditions avec l'opérateur logique `AND` (ET) :

```
$requeteSQL = 'SELECT * FROM recettes
WHERE estValide = 1 AND auteur='mickael.andrieu@exemple.com';
```

Il existe également le mot-clé `OR` (OU logique). Par exemple, pour obtenir toutes les recettes écrites par Mickaël ou Laurène, vous écrirez la requête suivante :

```
$requeteSQL = 'SELECT * FROM recettes
WHERE auteur='mickael.andrieu@exemple.com' OR auteur='laurene.castor@
exemple.com';
```

## ORDER BY

`ORDER BY` sert à ordonner nos résultats. Nous pourrions ainsi classer les recettes par ordre alphabétique :

```
$requeteSQL = 'SELECT * FROM recettes
WHERE auteur='mickael.andrieu@exemple.com' ORDER BY titre;
```

Par défaut, l'ordre ascendant est implicite. Il peut être précisé par le mot-clé `ASC`. Pour classer dans l'ordre inverse, il faut ajouter le mot-clé `DESC` :

```
$requeteSQL = 'SELECT * FROM recettes
WHERE auteur='mickael.andrieu@exemple.com' ORDER BY titre DESC;
```

`ORDER BY` classe aussi bien les chaînes de caractères (ordre alphabétique normal ou inverse) que les nombres (ordre croissant ou décroissant).

### LIMIT

`LIMIT` sert à ne sélectionner qu'une partie des résultats. C'est très utile lorsque vous obtenez beaucoup de résultats et que vous souhaitez les paginer (c'est-à-dire afficher, par exemple, les 30 premiers résultats sur la page 1, les 30 suivants sur la page 2, etc).

Par exemple, limitons-nous aux deux premières recettes trouvées :

```
$requeteSQL = 'SELECT * FROM recettes LIMIT 2;
```

On peut en plus préciser un `OFFSET`, qui est le résultat à partir duquel on compte les valeurs à conserver.

## Construire des requêtes en fonction de variables

Les requêtes que nous avons étudiées jusqu'ici étaient simples et exécutaient toujours la même opération. Or, les choses deviennent intéressantes quand on utilise des variables de PHP dans les requêtes.

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

On ne concatène **JAMAIS** une requête SQL pour passer des variables, au risque de créer des injections SQL (https://fr.wikipedia.org/wiki/Injection_SQL).

Dans un premier temps, on pourrait « préparer » la requête sans sa partie variable, qu'on représenterait avec un marqueur sous forme de point d'interrogation :

```
<?php
$requeteSQL = 'SELECT * FROM recettes WHERE auteur = ?';

$requetePreparee = $bdd->prepare($requeteSQL);
$requetePreparee->execute(['mickael.andrieu@exemple.com']);
$recettes = $requetePreparee->fetchAll();
?>
```

Si la requête contient beaucoup de parties variables, il est plus pratique d'utiliser des arguments nommés au lieu des points d'interrogation. En guise d'exemple complet, voici la requête pour retrouver les recettes valides de Mathieu, écrite en respectant les meilleures pratiques :

```
<?php
```

```
$requeteSQL = 'SELECT * FROM recettes WHERE auteur = :auteur AND estValide
= :estValide';

$bdd->prepare($requeteSQL);
$recettes = $bdd->execute([
   'auteur' => 'mathieu.nebra@exemple.com',
   'estValide' => 1,
]);
?>
```

Les points d'interrogation ont été remplacés par les marqueurs nominatifs `:auteur` et `:estValide` (ils sont précédés du caractère deux-points, comme vous le voyez).

Cette fois, ces marqueurs sont remplacés par les variables à l'aide d'un tableau associatif. Quand il y a beaucoup de paramètres, cela donne plus de clarté. De plus, contrairement aux points d'interrogation, nous ne sommes cette fois plus obligés d'envoyer les variables dans le même ordre que la requête.

## Traquer les erreurs

Lorsqu'une requête SQL échoue, bien souvent PHP vous dira qu'il y a eu une erreur à la ligne du `fetchAll()` :

```
Fatal error: Call to a member function fetchAll() on a non-object in
index.php on line 13
```

Ce n'est pas la ligne du `fetchAll()` qui est en cause : c'est souvent vous qui avez mal écrit votre requête SQL quelques lignes plus haut.

Pour afficher des détails, il faut activer les erreurs lors de la connexion à la base de données via PDO en adaptant la création de l'objet `$bdd` :

```
<?php
$bdd = new PDO(
   'mysql:host=localhost;dbname=partage_de_recettes;charset=utf8',
   'root',
   'root',
   [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION],
);
?>
```

Désormais, toutes vos requêtes SQL qui comportent des erreurs les afficheront avec un message beaucoup plus clair. Supposons par exemple que j'écrive mal le nom du champ :

```
SELECT titree FROM recettes
```

L'erreur suivante s'affichera alors :

```
Unknown column 'titree' in 'field list'
```

Cela signifie : « la colonne `titree` est introuvable dans la liste des champs ». En effet, le champ correct s'appelle `titre`.

Lorsque vous rencontrez un problème avec une requête et que vous demandez de l'aide, pensez **toujours** à activer les erreurs lors de la connexion à la base de données ; cela vous retournera un message d'erreur détaillé. Personne ne pourra vous aider si vous donnez juste le message par défaut `Call to a member function fetchAll() on a non-object`.

## En résumé

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

# 20 Écrire des données

Nous avons vu dans le chapitre précédent qu'on pouvait facilement récupérer et trier des informations de notre base de données. Il est maintenant temps de proposer aux utilisateurs un formulaire de création et de modification des recettes. Vous devrez créer les liens et formulaires nécessaires à votre projet.

Pour cela, nous allons aborder de nouvelles requêtes SQL fondamentales : `INSERT`, `UPDATE` et `DELETE`.

Pour faciliter la compréhension de ce chapitre, vous pouvez télécharger une version mise à jour du projet fil rouge : https://drive.google.com/file/d/1U5dkxVy0dxlQ78NnhyoFfn2DKU2ppkHT/view?usp=sharing.

## INSERT INTO : ajouter des données

Pour ajouter une recette, vous aurez besoin de connaître la requête SQL suivante :

```
INSERT INTO recettes(titre, recette, auteur, estValide) VALUES (:titre,
:recette, :auteur, :estValide)
```

Étudions un peu cette requête.

- D'abord, vous devez commencer par les mots-clés `INSERT INTO` qui indiquent que vous voulez insérer une entrée.
- Vous précisez ensuite le nom de la table (ici, `recettes`), puis listez entre parenthèses les noms des champs dans lesquels vous souhaitez placer des informations.

- Enfin – et c'est là qu'il ne faut pas se tromper –, vous inscrivez `VALUES` suivi des valeurs à insérer **dans le même ordre que les champs que vous avez indiqués**.

Utilisons cette requête pour notre projet. Votre mission, si vous l'acceptez, consiste à créer un formulaire pour ajouter des recettes. Vous procéderez en trois temps :

1. Ajouter un formulaire PHP de création de recette.
2. Vérifier les données soumises en PHP.
3. À l'aide de PDO, insérer la nouvelle recette dans la base de données.

Vous connaissez déjà les deux premières étapes, donc nous ne les détaillerons pas.

Généralement, on récupérera des variables de `$_POST` (issues d'un formulaire) pour insérer une entrée dans la base de données :

```
<? php
// Traitement du formulaire d'insertion de recette
session_start();

include_once('./../config/mysql.php');
include_once('./../config/user.php');
include_once('./../variables.php');

// Vérification du formulaire soumis
if (!isset($_POST['titre']) || !isset($_POST['recette'])) {
  echo 'Il faut un titre et une recette pour soumettre le formulaire.';
  return;
}
$titre = $_POST['titre'];
$recette = $_POST['recette'];

// Insérer dans la base
try {
  $bdd = new PDO('mysql:host=localhost;dbname=partage_de_
recettes;charset=utf8', 'root', 'root');
}
catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}

// Écriture de la requête
$requeteSQL = 'INSERT INTO recettes(titre, recette, auteur, estValide)
VALUES (:titre, :recette, :auteur, :estValide)';

// Préparation
$insererRecette = $bdd->prepare($requeteSQL);

// Exécution ! La recette est maintenant en base de données
$insererRecette->execute([
  'titre' => $titre,
  'recette' => $recette,
  'auteur' => $_SESSION['UTILISATEUR_CONNECTE'],
  'estValide' => 1, // 1 = true, 0 = false
]);
?>
```

Vous remarquerez qu'aucune valeur n'est passée pour le champ `id` de la recette. En effet, le champ a la propriété `AUTO_INCREMENT`, donc MySQL lui attribuera automatiquement une valeur. Par ailleurs, vous remarquerez que j'ai choisi le type `INT` pour le champ `estValide` ; cette information vaut `1` pour `true` et `0` pour `false`. Il est également possible de déclarer ce champ de type `BOOL` pour lui passer les valeurs PHP `true` et `false`.

## UPDATE : modifier des données

Vos utilisateurs souhaitent maintenant pouvoir modifier leurs recettes. Pour cela, vous aurez besoin de deux nouveaux mots-clés : `UPDATE` et `SET`.

En imaginant qu'on fournit un formulaire d'édition et que l'on autorise les utilisateurs à éditer les champs `titre` et `recette`, voici la requête SQL qui correspond :

```
UPDATE recettes SET titre = :titre, recette = :recette WHERE r_id = :id
```

- Tout d'abord, le mot-clé `UPDATE` indique qu'on va modifier une entrée.
- Ensuite, vient le nom de la table (`recettes`).
- 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 et leur valeur, séparés par des virgules. Ici, on modifie uniquement les champs `titre` et `recette`.
- Enfin, le mot-clé `WHERE` est tout simplement indispensable. Il indique à MySQL quelle entrée il doit modifier (sinon, toutes les entrées seraient affectées).
- On se base très souvent sur le champ `r_id` pour indiquer **quelle entrée** doit être modifiée. Toutefois, d'autres champs peuvent convenir. Par exemple, pour désactiver toutes les recettes d'un utilisateur, vous écrirez la requête suivante :

```
UPDATE recettes SET estValide = 0 WHERE auteur = 'mickael.andrieu@exemple.com'
```

- Elle peut se traduire par : « dans la table `recettes`, modifier toutes les entrées dont le champ `auteur` est égal à `mickael.andrieu@exemple.com` et passer la valeur de `estValide` à `0` (`false`) » .

Quel que soit le nombre d'entrées, cette requête suffit pour mettre à jour toute la table.

## DELETE : supprimer des données

Enfin, voilà une dernière requête qui pourra se révéler utile : `DELETE`. Rapide et simple à utiliser, elle est quand même un peu dangereuse : après suppression, il n'y a aucun moyen de récupérer les données, alors faites bien attention !

Par exemple, voici comment supprimer une recette à partir de son identifiant :

```
DELETE FROM recettes WHERE r_id = :id
```

Il n'y a rien de plus facile :

- `DELETE FROM` : pour dire « supprimer dans » ;
- `recettes` : le nom de la table ;
- `WHERE` : indispensable pour indiquer quelle(s) entrée(s) supprimer.

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

Je vous laisse essayer cette requête en PHP.

## Traiter les erreurs SQL

Comme vous le savez, SQL est un langage à part entière dont on se sert en PHP. Au même titre qu'en PHP, on peut commettre des erreurs en SQL. Il se peut par exemple que votre requête soit mal écrite, que la table que vous voulez ouvrir n'existe pas, etc. Bref, les erreurs possibles sont là encore nombreuses.

Repérez la requête qui pose problème et forcez l'affichage de l'erreur s'il y en a une, comme ceci :

```
<?php
$suppressionRecette = $bdd->prepare('DELETE FROM recettes WHERE r_id = :id');
$suppressionRecette->execute([
   'id' => $id,
]) or die(print_r($bdd->errorInfo()));
?>
```

Si la requête fonctionne, aucune erreur ne sera affichée. Si en revanche la requête échoue, PHP arrêtera de générer la page et vous affichera l'erreur.

À partir de là, il va falloir vous débrouiller tout seul, car les erreurs SQL sont assez nombreuses et je ne peux pas toutes les lister. En général, MySQL vous dit : `You have an error in your SQL syntax near 'XXX'`. Relisez bien votre requête SQL ; l'erreur se trouve généralement près de l'endroit qu'on vous indique.

Sur le même modèle et pour améliorer votre application, pourquoi ne pas créer un formulaire de création de compte ? Il vous suffira d'adapter la logique expliquée pour les recettes et les requêtes SQL pour insérer de nouveaux comptes utilisateurs en base de données.

## En résumé

- On utilise différents mots-clés en fonction du type de modification que l'on souhaite effectuer :
  - `INSERT INTO` : ajout d'une entrée ;
  - `UPDATE` : modification d'une ou plusieurs entrée(s) ;
  - `DELETE` : suppression d'une ou plusieurs entrée(s).
- Comme pour la sélection de données, on utilise les marqueurs pour personnaliser nos requêtes en fonction de variables.
- Lorsqu'on utilise `UPDATE` ou `DELETE`, il faut penser à filtrer avec un `WHERE` sinon toute la table sera affectée par l'opération.
- PHP indique si une erreur est survenue avec MySQL.

# 21 Ajouter des commentaires grâce aux jointures SSL

Jusque-là, vous n'avez travaillé que sur une seule table à la fois. Dans la pratique, vous aurez certainement plusieurs tables dans votre base, interconnectées pour la plupart. Cela vous aidera à mieux découper vos informations, à éviter les répétitions et à faciliter ainsi la gestion de vos données. Par exemple, dans notre table `recettes`, on répète à chaque fois l'adresse de l'auteur de la recette alors qu'elle existe déjà dans la table `utilisateurs`.

Pour bien comprendre l'intérêt de la notion de jointure entre tables, nous allons mettre en place une gestion des commentaires sur la page d'une recette. Il faudra lier le commentaire à la fois à un utilisateur et à la recette.

## Modéliser une relation

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

- un identifiant unique ;
- une recette ;
- un auteur ;
- une date de publication ;
- une note (disons de 0 à 5).

**Contenu de la table commentaires**

| IC_IDD | RECETTE | AUTEUR | DATE | NOTE | COMMENTAIRE |
|---|---|---|---|---|---|
| 1 | Cassoulet | mickael.andrieu@exemple.com | 03-08-2021 18:00:00 | 1 | Bof ! |
| 2 | Cassoulet | laurene.castor@exemple.com | 01-08-2021 12:00:00 | 1 | Super recette :) |
| 3 | Couscous | mathieu.nebra@exemple.com | 30-08-2021 12:00:00 | 1 | Pas bon du tout :( |
| 4 | Couscous | mickael.andrieu@exemple.com | ... | ... | ... |

Comme vous le voyez, l'auteur et le nom de la recette apparaissent autant de fois qu'il y a de commentaires d'un auteur ou sur une recette en particulier. Pourtant, vous aviez déjà centralisé les informations sur les utilisateurs dans la table `utilisateurs`.

**Contenu de la table utilisateurs**

| U_ID | NOM | COURRIEL | MOTPASSE | AGE |
|---|---|---|---|---|
| 1 | Mickaël Andrieu | mickael.andrieu@exemple.com | ... | 34 |
| 2 | Laurène Castor | laurene.castor@exemple.com | ... | 28 |
| 3 | Mathieu Nebra | ... | ... | ... |

À l'aide de MySQL et grâce au champ `u_id`, vous êtes déjà capable de récupérer toutes les informations sur un utilisateur sans avoir à les recopier dans une autre table.

Pour rappel :

```
# Toutes les informations sur Laurène !
SELECT * from utilisateurs WHERE u_id = 2
```

Maintenant, il faut modifier la structure de la table `commentaires` pour faire référence aux données disponibles dans la table `utilisateurs`. Pour cela, le mieux est de créer un champ **`u_id`** dans la table `commentaires`, qui fait référence au champ **`u_id`** dans la table `utilisateurs`.

**Modification de la table commentaires**

| C_IIDD | RECETTE | AUTEUR | DATE | NOTE | COMMENTAIRE |
|---|---|---|---|---|---|
| 1 | Cassoulet | **1** | 03-08-2021 18:00:00 | 1 | Bof ! |
| 2 | Cassoulet | **2** | 01-08-2021 12:00:00 | 1 | Super recette :) |
| 3 | Couscous | **3** | 30-08-2021 12:00:00 | 1 | Pas bon du tout :( |
| 4 | Couscous | **1** | ... | ... | ... |

MySQL sait donc que le `u _ id` de valeur `1` dans la table `commentaires` correspond à Mickaël ?

Non, il ne le sait pas. Il ne voit que des nombres et il ne fait pas la relation entre les deux tables. Il va falloir lui expliquer cette relation dans une requête SQL. Pour cela, on va effectuer une **jointure** entre les deux tables.

## Qu'est-ce qu'une jointure ?

Nous avons donc maintenant deux tables :

- `commentaires` ;
- `utilisateurs`.

Les informations sont séparées dans des tables différentes, ce qui évite de les dupliquer sur le disque.

Lorsqu'on récupère la liste des commentaires, si on souhaite obtenir le nom des auteurs, il faut adapter la requête pour récupérer aussi les informations issues de la table `utilisateurs`. Pour cela, on doit réaliser une **jointure**.

Il existe plusieurs types de jointures, qui nous permettent de choisir exactement les données que l'on veut récupérer. Je vous propose de découvrir les deux plus importantes :

- **les jointures internes**, qui ne sélectionnent que les données ayant une correspondance entre les deux tables ;
- **les jointures externes**, qui sélectionnent toutes les données, même si certaines n'ont pas de correspondance dans l'autre table.

Il est important de bien comprendre la différence entre ces deux types. Pour cela, imaginons que nous ayons une quatrième personne dans la table des utilisateurs, un certain Manuel Vache, qui n'a jamais publié de commentaire.

**Ajout d'une nouvelle personne dans la table utilisateurs**

| U_ID | NOM | COURRIEL | MOTPASSE | AGE |
|---|---|---|---|---|
| 1 | Mickaël Andrieu | mickael.andrieu@exemple.com | ... | 34 |
| 2 | Laurène Castor | laurene.castor@exemple.com | ... | 28 |
| 3 | Mathieu Nebra | ... | ... | ... |
| 4 | Manuel Vache | ... | ... | 58 |

Manuel Vache est référencé dans la table `utilisateurs`, mais il n'apparaît nulle part dans la table `commentaires`.

Si vous récupérez les données des deux tables à l'aide :

- d'une **jointure interne** : Manuel Vache n'apparaîtra pas dans les résultats de la requête ;
- d'une **jointure externe** : vous aurez toutes les données de la table des utilisateurs, y compris celles de Manuel Vache.

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.

**Données récupérées avec une jointure interne**

| NOM | COMMENTAIRE |
|---|---|
| Mickaël Andrieu | Bof ! |
| Laurène Castor | Super recette :) |
| Mathieu Nebra | Pas bon du tout :( |
| ... | ... |

**Résultats obtenus avec une jointure externe**

| NOM | COMMENTAIRE |
|---|---|
| Mickaël Andrieu | Bof ! |
| Laurène Castor | Super recette :) |
| Mathieu Nebra | Pas bon du tout :( |
| Manuel Vache | `NULL` |
| ... | ... |

Nous allons maintenant voir comment réaliser ces deux types de jointures en pratique.

## Les jointures internes

On réalise une jointure interne à l'aide du mot-clé `JOIN` :

```
# Liste des noms et commentaires associés
SELECT u.nom, c.commentaire
FROM utilisateurs u
INNER JOIN commentaires c
ON u.u_id = c.u_id
```

Ici, on récupère les données depuis une table principale (ici, `utilisateurs`) et on réalise une jointure interne (`INNER JOIN`) avec une autre table (`commentaires`). La liaison entre les champs est faite dans la clause `ON` un peu plus loin.

Si vous voulez filtrer (`WHERE`), ordonner (`ORDER BY`) ou limiter les résultats (`LIMIT`), vous devez l'écrire à la fin de la requête, après le `ON u.u_id = c.u_id`.

```
# Liste des noms et commentaires associés
# à la recette Cassoulet
# du plus récent au plus vieux
# limité à 10
SELECT u.nom, c.commentaire
FROM utilisateurs u
INNER JOIN commentaires c
ON u.u_id = c.u_id
WHERE u.recette = "Cassoulet"
ORDER BY c.date DESC
LIMIT 10
```

Cette requête signifie (inspirez un grand coup avant de lire) : « Récupère le commentaire et le nom de l'auteur dans les tables `utilisateurs` et `commentaires`, la liaison entre les tables se faisant sur le champ `u_id`, prends uniquement les commentaires concernant la recette de cassoulet, trie-les par date décroissante et ne garde que les 10 premiers. »

Il faut s'accrocher avec des requêtes de cette taille !

## Les jointures externes

Les jointures externes permettent de récupérer toutes les données, même celles qui n'ont pas de correspondance. On pourra ainsi obtenir Manuel Vache dans la liste même s'il n'a jamais commenté.

Il y a deux écritures à connaître : `LEFT JOIN` et `RIGHT JOIN`. Cela revient pratiquement au même, avec une subtile différence que nous allons voir.

### LEFT JOIN : récupérer toute la table de gauche

Reprenons la jointure à base de `INNER JOIN` et remplaçons tout simplement `INNER` par `LEFT` :

```
# Liste des noms et commentaires associés
SELECT u.nom, c.commentaire
FROM utilisateurs u
LEFT JOIN commentaires c
ON u.u_id = c.u_id
```

`utilisateurs` est appelée la « table de gauche » et `commentaires` 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 `commentaires`.

**Contenu de la table utilisateurs avec LEFT JOIN**

| NOM | COMMENTAIRE |
|---|---|
| Mickaël Andrieu | Bof ! |
| Laurène Castor | Super recette :) |
| Mathieu Nebra | Pas bon du tout :( |
| Manuel Vache | NULL |

## RIGHT JOIN : récupérer toute la table de droite

Le `RIGHT JOIN` demande à récupérer toutes les données de la table de droite, même si celle-ci n'a pas d'équivalent dans l'autre table. Prenons la requête suivante :

```
# Liste des noms et commentaires associés
SELECT u.nom, c.commentaire
FROM utilisateurs u
RIGHT JOIN commentaires c
ON u.u_id = c.u_id
```

La table de droite est `commentaires`. On récupérerait donc tous les commentaires, même ceux qui n'ont pas d'auteur associé.

Comment est-il possible qu'un commentaire n'ait pas d'auteur associé ?

Il y a deux cas possibles :

- Soit le champ `u_id` contient une valeur qui n'a pas d'équivalent dans la table `utilisateurs`, par exemple `56`.
- Soit le champ `u_id` vaut `NULL`, c'est-à-dire que la personne qui a commenté souhaite rester anonyme, par exemple.

Notre projet fil rouge ne permet pas à un visiteur de publier un commentaire sans s'authentifier à l'aide de son adresse électronique, mais rien ne vous empêche de développer cette fonctionnalité !

On obtiendrait donc les données présentées dans le tableau suivant.

**Résultats obtenus**

| NOM | COMMENTAIRE |
|---|---|
| Mickaël Andrieu | Bof ! |
| Laurène Castor | Super recette :) |
| Mathieu Nebra | Pas bon du tout :( |
| NULL | Ressorti brûlé à mon premier essai... |

D'accord, mais ne faudrait-il pas faire la même chose avec la table `recettes` ?

Si, bien sûr ! Vous coderez donc non pas une mais deux jointures. Par exemple, la liste des utilisateurs, commentaires et noms de recettes s'obtient comme suit :

```
SELECT u.nom, c.commentaire, r.titre
FROM utilisateurs u
JOIN commentaires c
  ON u.u_id = c.u_id
JOIN recettes r
  ON c.r_id = r.r_id
```

**Résultats obtenus**

| NOM | COMMENTAIRE | TITRE |
|---|---|---|
| Mickaël Andrieu | Bof ! | Cassoulet |
| Laurène Castor | Super recette :) | Cassoulet |
| Mathieu Nebra | Pas bon du tout :( | Cassoulet |

Si vous avez envie d'expérimenter la nouvelle fonctionnalité de gestion de commentaires, téléchargez la version mise à jour du projet : *https://drive.google.com/file/d/1b5I8kNigmPOq1KbFPz9VPAMWR61xEFoz/view?usp=sharing*.

## En résumé

- Les bases de données permettent d'associer plusieurs tables entre elles.
- Une table peut contenir les `id` d'une autre table, ce qui permet de faire la liaison entre les deux.
- Pour rassembler les informations au moment de la requête, on effectue des **jointures**, à l'aide du mot-clé `JOIN`.

- On distingue les jointures internes, qui retournent des données uniquement s'il y a une correspondance entre les deux tables, des jointures externes, qui retournent toutes les données même s'il n'y a pas de correspondance.

# 22 Aller plus loin avec les fonctions SQL

Vous connaissez déjà les fonctions en PHP, vous allez maintenant découvrir dans ce chapitre que SQL en propose lui aussi un certain nombre, notamment pour réaliser des calculs directement sur les données.

Ces fonctions sont moins nombreuses qu'en PHP, mais elles sont spécialement dédiées aux bases de données et se révèlent très puissantes dans la pratique. Pour reprendre notre exemple du site fil rouge, elles permettent entre autres de récupérer très simplement la moyenne des notes obtenues par une recette ou d'ajouter une date de publication aux commentaires. Elles se révèlent également indispensables lorsqu'on doit travailler avec des dates en SQL.

Les fonctions SQL peuvent être classées en deux catégories :

- **Les fonctions scalaires :** elles agissent sur chaque entrée. Par exemple, vous pouvez transformer en majuscules la valeur de chacune des entrées d'un champ.
- **Les fonctions d'agrégat :** lorsque vous utilisez ce type de fonction, des calculs sont faits sur l'ensemble de la table pour retourner **une** valeur (note moyenne par exemple).

## Utiliser une fonction scalaire SQL

Nous allons d'abord découvrir le mode d'emploi d'une fonction SQL de type **scalaire** : `DATE_FORMAT()`. Lorsque vous aurez appris à vous en servir, vous serez capable de faire de même avec toutes les autres fonctions scalaires.

Pour nos exemples, nous allons nous baser sur la table `commentaires` que nous connaissons bien maintenant. Pour rappel, voici à quoi elle ressemble en considérant les commentaires sur la recette n° `1` :

**Contenu de la table commentaires**

| C_ID | U_ID | R_ID | COMMENTAIRE | DATE | NOTE |
|---|---|---|---|---|---|
| 1 | 1 | 1 | Bof ! | 2021-07-16 13:56:48 | 2 |
| 2 | 3 | 1 | Pas bon du tout :( | 2021-07-14 18:27:32 | 0 |
| 3 | 2 | 1 | Super recette :) | 2021-07-12 16:12:00 | 5 |
| 4 | ... | ... | ... | ... | ... |

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 adoptée par les programmeurs.

La fonction `DATE_FORMAT()` sert à convertir un *timestamp* en une date lisible :

```
SELECT *, DATE_FORMAT(c.date, "%d/%m/%Y") FROM recettes r LEFT JOIN
commentaires c ON r.r_id = c.r_id WHERE r.r_id = 1
```

À l'aide de cette fonction, nous récupérons directement la date au format jour/mois/année, au lieu des heures/minutes/secondes qui ne nous intéressent pas ici.

**La table reste la même**. La fonction `DATE_FORMAT()` modifie seulement la valeur envoyée à PHP. Cela crée en fait un « champ virtuel », ou **alias**, qui n'existe que le temps de la requête. Il est conseillé de donner un nom à ce champ qui représente les dates des commentaires ; il faut utiliser pour cela le mot-clé `AS` :

```
SELECT DATE_FORMAT(c.date, "%d/%m/%Y") AS c_date FROM recettes r LEFT JOIN
commentaires c ON r.r_id = c.r_id WHERE r.r_id = 1
```

PHP ne récupère que le champ `c_date` (même s'il n'existe pas en tant que tel dans la table) :

| C_DATE |
|---|
| 16/07/2021 |
| 14/07/2021 |
| 12/07/2021 |
| ... |

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

```
SELECT *, DATE_FORMAT(c.date, "%d/%c/%Y") AS c_date FROM recettes r LEFT
JOIN commentaires c ON r.r_id = c.r_id WHERE r.r_id = 1
```

Vous savez maintenant utiliser une fonction SQL scalaire.

## Utiliser une fonction d'agrégat SQL

Comme précédemment, nous allons montrer comment on utilise une fonction d'agrégat dans une requête SQL. L'essentiel est de comprendre comment s'utilise ce type de fonction ; vous pourrez ensuite appliquer ce que vous connaissez à n'importe quelle autre fonction du même type.

Plutôt que de modifier des valeurs une à une, les fonctions d'agrégat réalisent des opérations sur plusieurs entrées pour retourner une seule valeur.

Par exemple, `ROUND()` est une fonction scalaire qui arrondit une valeur : on récupère donc autant d'entrées qu'il y en a dans la table. En revanche, une fonction d'agrégat comme `AVG()` renvoie **une seule entrée** : la valeur moyenne de toutes les lignes (sur un champ contenant un nombre).

Regardons de près la fonction d'agrégat `AVG()` et utilisons-la sur le champ `note` qui – pour rappel – permet à l'utilisateur d'évaluer une recette :

```
SELECT AVG(c.note) AS noteMoyenne FROM recettes r LEFT JOIN commentaires c
ON r.r_id = c.r_id WHERE r.r_id = 1
```

On donne là encore un alias au résultat retourné par la fonction. La particularité, c'est que cette requête ne va retourner qu'une seule entrée, à savoir l'évaluation moyenne de la recette :

| noteMoyenne |
|---|
| 2.33333 |

Ce n'est pas très beau ! Ne pourrait-on pas calculer un arrondi de la moyenne ?

Si ! Tout comme en PHP on peut appeler une fonction dans une fonction, ici il est possible d'appliquer la fonction `ROUND()` avec une décimale à la moyenne calculée :

```
SELECT ROUND(AVG(c.note),1) AS noteMoyenne FROM recettes r LEFT JOIN
commentaires c ON r.r_id = c.r_id WHERE r.r_id = 1
```

Pour afficher cette information en PHP, on pourrait procéder comme d'habitude (cela fonctionne), mais pourquoi s'embêterait-on à coder une boucle alors qu'on sait qu'on ne va récupérer qu'une seule valeur avec la fonction d'agrégat ?

```
<?php
$requeteSQL = 'SELECT ROUND(AVG(c.note),1) as noteMoyenne FROM recettes r
LEFT JOIN commentaires c ON r.r_id = c.r_id WHERE r.r_id = 1';

// Préparation
$calculNoteMoyenne = $bdd->prepare($requeteSQL);

// Exécution
$calculNoteMoyenne->execute();

/** La fonction fetch est plus performante que fetchAll()
 * quand nous sommes certain(e)s de ne récupérer qu'une ligne.
 * https://www.php.net/manual/fr/pdostatement.fetch.php
 */
$noteMoyenneCassoulet = $calculNoteMoyenne->fetch();
?>
```

Ce code est plus simple et plus logique. On récupère l'unique entrée avec `fetch()` et on affiche ce qu'elle contient.

**Attention : ne mélangez pas une fonction d'agrégat avec d'autres champs !**

Soyez attentif à ce point car il n'est pas forcément évident à comprendre : vous **ne devez pas** récupérer d'autres champs de la table quand vous utilisez une fonction d'agrégat, contrairement aux fonctions scalaires. En effet, cela n'aurait pas de sens de coder ce qui suit :

```
# Requête qui provoque une erreur !
SELECT AVG(note) AS noteMoyenne, u_id FROM commentaires
```

On récupérerait d'un côté la note moyenne de tous les commentaires et de l'autre la liste des auteurs de ces commentaires… Il est impossible de représenter ceci dans un unique tableau. Comme vous le savez, SQL renvoie les informations sous la forme d'un tableau. Or, on ne peut pas représenter la moyenne des notes (qui tient en une seule entrée), en même temps que la liste des auteurs. Pour obtenir ces deux informations, il faut lancer deux requêtes (c'est d'ailleurs ce qui est codé dans le projet fil rouge).

## Grouper des données avec GROUP BY et HAVING

Donc, on ne peut pas récupérer d'autres champs lorsqu'on utilise une fonction d'agrégat.

### GROUP BY : grouper des données

En revanche, ce qui pourrait avoir du sens, ce serait de demander la **note moyenne des commentaires pour chaque recette**. Pour cela, on doit utiliser un nouveau mot-clé : `GROUP BY` qui signifie « grouper par ». On utilise cette clause en combinaison avec une fonction d'agrégat (comme `AVG`) pour obtenir des informations intéressantes sur des groupes de données.

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

```
SELECT AVG(note) AS noteMoyenne, r_id FROM commentaires GROUP BY r_id
```

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 regroupés par recette. Par conséquent, on obtiendra la liste des recettes de la table associées chacune à sa note moyenne.

| NOTEMOYENNE | R_ID |
|---|---|
| 2.50 | 1 |
| 3.00 | 2 |
| 4.50 | 3 |

Cette fois les valeurs sont cohérentes.

### HAVING : filtrer les données regroupées

`HAVING` est un peu l'équivalent de `WHERE`, mais 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.

Par exemple, la requête suivante récupère uniquement les recettes dont la note moyenne est supérieure ou égale à 3 :

```
SELECT AVG(note) AS noteMoyenne, r_id FROM commentaires GROUP BY r_id
HAVING note >= 3
```

`HAVING` ne doit s'utiliser que sur le résultat d'une fonction d'agrégat. Voilà pourquoi on l'utilise ici sur `noteMoyenne` et non sur `r_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, tandis que `HAVING` agit en second, *après* le groupement des données. On peut d'ailleurs très bien combiner les deux, comme dans l'exemple suivant :

```
SELECT AVG(note) AS noteMoyenne, r_id FROM commentaires
WHERE u_id = 1 GROUP BY r_id HAVING noteMoyenne >= 2
```

Ici, on demande à récupérer la note moyenne par recette de Mickaël (`WHERE`) lorsqu'elle est supérieure ou égale à 2 (`HAVING`).

## En résumé

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

# Index