Apprendre PHP Pour Les Zéros

- Mis en page par KoraS.
- Mis en partage sur le réseau eDonkey le Dimanche 6 juin 2004 par Koras
- Ce cours de PHP viens du site : http://www.siteduzero.com/

Détestant lire sur mon PC et aimant le travail bien fait, j'ai mis en page a fin d'impression ce cours très complet et si j'ai décider de le mettre en partage sur le réseau pour les gens qui ne connaitrais pas le site du zéro (a visiter ABSOLUMENT) et/ou qui voudrais l'imprimer CORRECTEMENT pour le lire tranquillement dans leurs fauteuils
Aller, bonne lecture les gars !!!!

- Programmez, crackez, scannez, rippez et mettez tout ça en partage sur le réseau eDonkey, on va leurs apprendre a nous vendre des logiciels a 200 Euros, des jeux a 60 Euros, des albums a 20 Euros, des livres a 50 Euros, des magazines a 8 Euros a tous ces empafés de capitalistes !

- A propos du réseau, j'ai remarqué depuis quelques temps une certaine tendance de la communauté eDonkey Française a prendre et a ne pas laisser en partage, cela est sans doute du au fait de la démocratisation d'Internet et des connections haut débits et donc a l'émergence de nombreux newbies (débutants), qui prennent sans se demander pourquoi, comment, par qui…….
- Vous vous devez de laisser en partage quelques temps ce que vous venez de prendre pour qu'il y est plus de « full sources » (personne possédant le fichier en entier), c'est **OBLIGATOIRE** !!!! Qu'est ce que vous risquez en n'appliquant pas cette loi ? C'est simple vous risquez un jours ou l'autre de devoir acheter tous ce que vous prenez **GRATUITEMENT** avec votre Mule les gars !!!! Et oui, déjà qu'avec les lois pourrisses qui ont étaient votées ces derniers temps de nombreuses personnes luttent contre nous, les FAI's (Fournisseurs d'Accès à Internet) collaborent avec les keufs et autres R.I.I.A, les espions sont partout sur le réseau et même sur nos PC's (et je suis pas parano, ce que je dit est prouver), alors si les utilisateurs se mettent a partir en couilles aussi, je ne donne pas 2 ans au Roi des réseau Peer To Peer (eDonkey bien surs) ! Contre ça, partageons le plus et le plus longtemps possible.
- J'ai également remarquer une montée en flèche du nombres de fichiers incomplets, quand vous décidez de mettre un fichier en partage, vous endossez une responsabilité, définissez la priorité de ce fichier au maximum (release ou très haute selon les programmes utilisés) en cliquant droit dessus dans la partie partage de votre logiciel, surtout, surtout, surtout, assurez vous avant de l'enlever qu'il y est au moins 20 personnes qui l'ont en entier et mieux encore, ne l'enlever pas, assumez jusqu'au bout votre rôle.
- La survie du réseau est entre nos mains les gars, faites tourner l'info, **IL FAUT PARTAGER !!!**

Longue vie a eDonkey et a tous les Hackers, Crackers, Rippers, Suppliers, Webmestres, etc, qui se décarcasse pour notre plaisir sans rien y gagner… au fait si, ils y gagnent notre estime et nos éternels remerciements !!!!!!

KoraS

Introduction À PHP

Ca y est ? Votre choix est fait : vous allez vous mettre au PHP. Alors je vous souhaite la bienvenue.

Faisons les présentations tout de suite : je suis M@teo21, et je serai votre guide tout au long de ce cours. Je vais vous faire découvrir PHP dans cette première partie, et je veillerai à ce que tout ce que je dis soit le plus clair possible. Si vous me suivez bien, je vous garantis que PHP n'aura bientôt plus de secret pour vous...

Ah, et je vous présente aussi l'éléPHPant. C'est la mascotte du PHP, vous le retrouverez sur la plupart des sites francophones traitant de PHP. C'est un signe de reconnaissance en quelque sorte.

Bon, comme vous ne savez toujours pas ce que c'est PHP, je vais vous l'expliquer ci-dessous. Mais vous verrez que vous redécouvrirez sans cesse PHP, car c'est un univers tellement riche et varié qu'on ne peut pas prétendre le connaître entièrement. Il y a toujours quelque chose à découvrir.

Qu'Est-Ce Que C'Est PHP ?

Est-ce que vous savez ce que c'est un site web ? Non, je vous prends pas pour des abrutis, mais j'ai dit que je partais de Zér0 alors faut que je tienne ma promesse.
Un site web, ben vous en avez un sous les yeux : le mien par exemple. Pour aller sur un site web, on tape son adresse, par exemple : http://www.siteduzero.com. En tapant l'adresse d'un site web, votre navigateur (Internet Explorer par exemple), vous emmènera visiter ce site web.

On peut faire beaucoup de choses sur un site web : apprendre (c'est ce que vous êtes en train de faire), jouer, discuter, échanger, s'informer etc...

Maintenant, deuxième question : avez-vous entendu parler du HTML ? Si oui, tant mieux. Si non, alors il faut absolument que vous sachiez ce que c'est avant de continuer. Vous pouvez lire mon cours sur le HTML en cliquant ici.

Pour rappel, le HTML c'est un langage qui vous permet de créer des pages web. En tapant un code spécial (les "tags", ou "balises"), on peut mettre du texte en gras, insérer une image etc etc...
Voici à quoi peut ressembler une page avec son code HTML :

Source 1.1.1 : un exemple de code HTML (page .html)

```
<html>

<head>
<title>Titre de la page</title>
</head>

<body bgcolor="blue" vlink="red">
Bienvenue sur mon site web !<br>
Cliquez <a href="http://www.siteduzero.com">ici</a> pour entrer !
</body>

</html>
```

Et PHP dans tout ça ? Eh bien PHP, c'est un autre langage qui vient se mettre au milieu de ce code HTML. Voici par exemple ce que ça peut donner (c'est un petit aperçu de ce que vous allez apprendre) :

Source 1.1.2 : du code PHP au milieu du code HTML (page .php)

```
<html>

<head>
<title>Titre de la page</title>
</head>

<body bgcolor="blue" vlink="red">
Bienvenue sur mon site web !
<? echo("Vous êtes le visiteur n°" . $nbre_visiteurs); ?> <br>
Cliquez <a href="http://www.siteduzero.com">ici</a> pour entrer !

</body>

</html>
```

Qu'est-ce qui est nouveau ici ? C'est cette ligne :

<? echo("Vous êtes le visiteur n°" . $nbre_visiteurs); ?>

Il y a toujours du langage HTML autour, mais on trouve au milieu des **instructions PHP**. Ce que je vais vous apprendre c'est à savoir manier des lignes de ce type. Oui, ça fait peut-être un peu peur ces caractères bizarres au milieu ($; ? >), mais bientôt cela vous sera familier (si si je vous l'assure).

Comme vous le voyez, une page qui ne contient que du HTML possède l'extension ".html". Une page qui contient du code PHP, elle a l'extension ".php".
Comme il y a eu plusieurs versions de PHP, il n'est pas rare que vous rencontriez des extensions .php3 ou .php4. La version actuelle de PHP est la v4.

Existe-t-il des pages qui ne contiennent que du PHP ?

Mmh, en fait non, on a quand même toujours besoin du HTML pour faire une page web. On ne peut pas y échapper !

En résumé : le HTML est pratique un moment, mais il est limité. A l'aide de PHP, vous pourrez réaliser bien plus de choses pour votre site web. Des exemples ?

- Un forum, où tout le monde peut discuter, échanger, s'entraider si quelqu'un a un problème.
- Un Chat, pour discuter en temps réel avec d'autres personnes !
- Un livre d'or : si votre site web plaît à vos visiteurs, ils peuvent laisser un message disant que votre site web est super, et tout le monde pourra le lire !
- Une newsletter : c'est très facile à mettre en place. Vous rédigez votre newsletter, vous cliquez sur un bouton, et là le mail s'envoit automatiquement à toutes les personnes inscrites à votre newsletter !
- Un compteur de visiteurs, visible ou caché, c'est vous qui voyez ce que vous préférez. Et comme c'est vous qui allez le créer, il n'y aura pas de pub (ceux qui utilisent un compteur avec une pub se font arnaquer je vous le dis de suite).

- Un système de news automatisé : vous allez sur une page, vous tapez le texte de le nouvelle news, et immédiatement après la page d'accueil de votre site s'actualise et tous vos visiteurs voient cette news !
- On peut imaginer alors qu'ils réagissent à cette news : ils donnent leur avis, se proposent pour vous aider etc...

PHP peut faire encore beaucoup plus que ça, mais c'était pour vous mettre l'eau à la bouche.

Ce qu'il faut bien retenir donc, c'est que *PHP vous permet de créer des pages web dynamiques*, qui se mettent à jour toutes seules sans que vous ayez à passer par là. En clair, vous pouvez être en vacances aux Bahamas, et votre site continuera à évoluer tout seul !
Autre gros avantage, vous allez vous en rendre compte, PHP inaugure l'ère du Webmaster Fainéant (avec un grand F) : une fois que vous avez mis votre site en place, il se met à jour tout seul, se transforme, sans que vous ayez à lever le petit doigt.

Si ça c'est pas la belle vie ! Vous comprenez un peu mieux maintenant pourquoi on s'intéresse de plus en plus au PHP ?

Différences Entre HTML Et PHP

Ce que je vais vous apprendre maintenant, ce n'est pas très compliqué, et pourtant beaucoup de gens se lancent dans le PHP sans le savoir !
Croyez-moi : si vous faites l'effort de comprendre comment ça marche (ça vous prendra 10 minutes), non seulement vous allez gagner beaucoup de temps ensuite, mais en plus *vous comprendrez* ce que vous ferez. Et ça, ça n'a pas de prix croyez-moi.

De quoi je vais vous parler ? Je vais vous expliquer ce qui se passe exactement quand un visiteur veut aller sur votre site web. Il tape l'adresse ok, mais ensuite ? La page s'affiche, d'accord, mais entre-temps que s'est-il passé ?
Ca c'est vraiment important, parce qu'en HTML et en PHP ça ne fonctionne pas vraiment pareil.

Il y a une notion fondamentale à connaître : les relations entre le client et le serveur. Quoi "beârk" ? Non non, il n'y a rien de sorcier là-dedans !

- Le client : celui qu'on appelle "le client", c'est vous :o). C'est vous qui êtes tranquille pépère installé devant votre ordinateur, et qui demandez à voir une page web. Tous les visiteurs d'un site web sont des clients. On va représenter l'ordinateur du client par cette machine :

- Le serveur : il n'y en a qu'un seul. Le serveur, c'est une sorte de gros ordinateur tout le temps connecté à Internet (avec une connexion très rapide). Cet ordinateur est installé quelque part dans le monde, il est tout le temps allumé, et personne n'y touche. Il travaille 24h/24, et ne s'occupe que de distribuer votre site web. En d'autres termes, personne ne joue dessus.
  Sa fonction ? Il contient votre site web sur son disque dur, et dès qu'un client demande à voir une page web, il la lui envoie. Pour représenter le serveur, je vais utiliser cette machine (notez qu'en général le serveur n'a pas d'écran : ça ne sert à rien puisque personne ne travaille dessus) :

Vous voyez ? C'est en fait très simple.
Pour ceux qui n'auraient pas tout bien compris, voici un exemple...

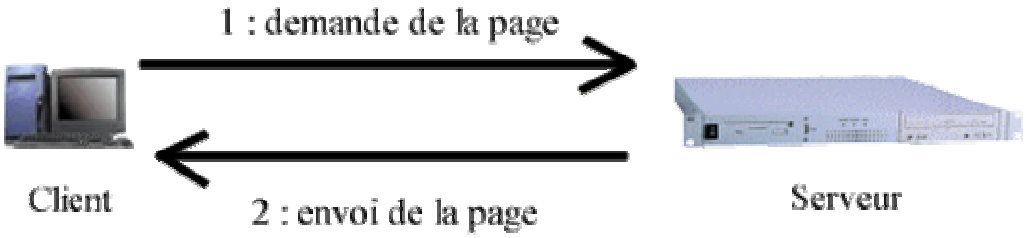
Imaginez un restaurant. Vous rentrez dedans, vous êtes **le client**. Vous commandez un Couscous Royal (arf j'ai faim). Le cuisinier, lui, c'est **le serveur** : vous lui avez demandé tel plat, il vous le livre.
Dès qu'un autre client se présente et demande un autre plat, le cuisinier le lui donne. Et il travaille ainsi inlassablement tout le temps.

Eh bien c'est pareil sur Internet : le serveur est un ordinateur qui envoie des pages web aux clients qui le lui demandent. Et il travaille sans arrêt comme un forcené.

Bien, maintenant que vous avez compris ça, je vais vous montrer le petit plus qui fait toute la différence entre une page HTML et une page PHP.

Avant : En HTML

Je vous rappelle qu'une page HTML possède l'extension .html, comme exemple.html
Je ne vais pas entrer dans les détails, mais en gros voici comment ça fonctionne pour une page HTML :



Il y a 2 étapes :

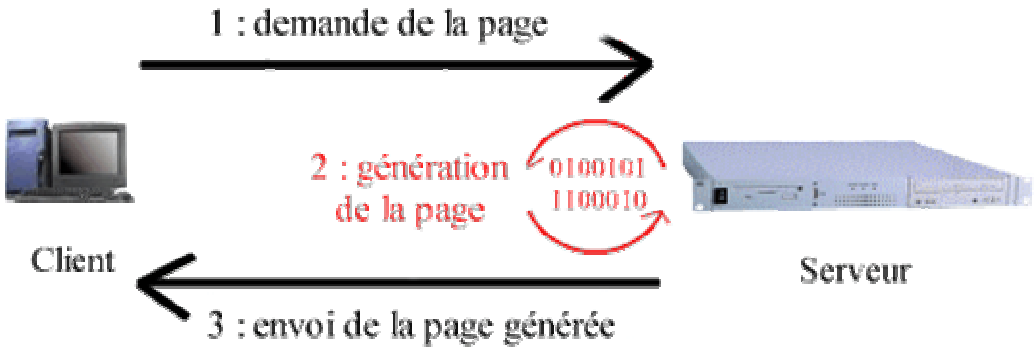
1. Le client (c'est vous) demande à voir une page web. Il va donc faire une demande au serveur : "S'il te plaît, envoie-moi la page vacances.html".
2. Le serveur lui répond en lui envoyant la page vacances.html : "Tiens, voici la page que tu m'as demandée".

Tout ça se passe très poliement bien entendu.
Le client voulait consulter la page vacances.html sur un site web : il l'a demandée au serveur qui gère ce site, et le serveur lui a envoyé la page que le client voulait. La page s'affiche alors sur l'écran du client, sous ses yeux ébahis.

Cela se passe à chaque fois que vous consultez une page HTML. Mais qu'est-ce qui peut bien changer avec PHP ?

Maintenant : En PHP

Il y a une étape qui vient s'ajouter entre les deux : la page PHP est générée par le serveur avant l'envoi. Schématiquement ça donne ça :



Voyons à nouveau les étapes :

1. Le client demande à voir une page PHP. Pour lui il n'y a aucune différence. Il demande la page au serveur, toujours aussi poliement : "S'il te plaît, envoie-moi la page vacances.php".
2. Mais là, il y a une étape très importante, qui fait toute la différence en PHP. Le serveur n'envoie pas de suite la page au client. Il la **génère**. En effet, le client n'est pas capable de lire une page PHP (seul le serveur sait faire ça). Le client ne peut lire que des pages HTML.
   Ce que fait le serveur est simple : il va transformer la page PHP en page HTML, pour que le client puisse la lire.
3. Enfin, une fois que la page est générée, elle ne contient plus que du code HTML. Le serveur peut l'envoyer au client : "Tiens, voici la page que tu as demandé".

Je vais vous en dire un peu plus sur cette deuxième étape : celle de la génération de la page. Il est important de bien comprendre ce qui s'y passe.

Que veut dire "génération de la page PHP" ?

Je vous ai montré un bout de code PHP au début de ce chapitre. Le revoici :

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

Les ordinateurs des clients ne savent pas lire ce code PHP : ils ne connaissent que le HTML. C'est donc au serveur de transformer le code PHP en HTML.

Mais à quoi sert le code PHP alors ?

Il contient des **instructions**. Il demande au serveur d'effectuer des actions : donner l'heure, le nombre de personnes connectées sur le site etc... Bref, le PHP donne des ordres au serveur.
Ce genre de choses était impossible en HTML. Avec PHP, c'est possible, et vous verrez que ça change tout.

N'oubliez pas qu'une page PHP contient aussi du code HTML.
Tant qu'il y a du code HTML, le serveur n'y touche pas. Dès qu'il tombe sur du code PHP, il le lit, il l'exécute (il fait ce que le code lui demande), et il transforme ça en HTML.

En fin de compte, la page générée ne contient plus que du HTML : le client peut alors la lire.

Ce qui est particulier ici, c'est que cette page générée est destinée à un seul client. Quand un nouveau client se présente, le serveur recommence à générer une page HTML.
Ca veut dire qu'en fait la page générée peut être à chaque fois unique. C'est bien ça qui est génial par rapport au HTML : en HTML la page envoyée était toujours la même, le serveur envoyait juste le fichier. En PHP, le serveur travaille pour le client et lui offre une page personnalisée.

Ce premier chapitre s'achève ici. Mon but était de vous amener en douceur vers PHP.J'ai essayé principalement de vous parler des relations entre le client et le serveur. Ca peut paraître un peu bizarre pour commencer le PHP, mais je vous assure que ça va vous servir.
Si vous avez l'impression d'être un peu embrouillé, ce n'est pas bien grave : ça n'aura pas de conséquence pour la suite du cours. Le principal c'est que vous ayez au moins lu ce chapitre, comme ça si quelqu'un vous parle de "relations client/serveur" vous ferez pas une mine déconfite.

Vous vous demandez peut-être à quelle sauce vous allez être mangés... Rassurez-vous, il n'y a rien de bien compliqué dans tout ça.Je sais aussi que vous êtes friands de Travaux Pratiques (TP) : ne vous inquiétez pas ça viendra, c'est prévu au programme. Je tiens à ce que vous pratiquiez un peu, histoire que vous me montriez petit à petit ce que vous êtes capables de faire !

Un Programme Utile : EasyPHP

Le premier chapitre vous aura servi d'introduction dans l'univers de PHP. C'était l'idéal pour commencer, mais il n'y avait rien de très concret.
Alors comme je sais que vous aimez passer à l'acte, je ne vous fais pas plus attendre : dans ce chapitre on va commencer à faire des manipulations !

Oh, il ne s'agit encore que de préparatifs, mais ils en valent la peine. Ce chapitre va porter autour d'un programme français appelé EasyPHP, qui va nous être extrêmement utile par la suite !

Pourquoi Ai-Je Besoin De Ce Programme ?

Oui, pourquoi diable allez-vous avoir besoin de ce programme ?
Voilà une bonne question pour commencer.

Comme je vous l'ai expliqué dans le chapitre précédent, seul le serveur peut lire le PHP. Le client (c'est-à-dire vous), ne peut pas lire le PHP.
Ouaïe aïe aïe problème ! Comment allez-vous pouvoir vérifier si votre travail en PHP fonctionne ? Votre PC ne sait pas lire le PHP !
Il va donc falloir trouver un moyen pour "apprendre" le PHP à votre ordinateur. Vous pourrez alors travailler dessus pour réaliser votre site en PHP.

EasyPHP est la solution, qui vous épargnera bien des maux de tête. Parce qu'en effet, vous vous en doutez c'était trop facile d'installer un programme "PHP" et puis basta ! Non, vous allez avoir besoin de plusieurs programmes...
EasyPHP est en fait un "package" qui contient tous les programmse nécessaires pour pouvoir traiter du PHP ! Vous n'aurez rien à faire : ils s'installeront tous seuls !

 Le site web de EasyPHP est : www.easyphp.org

Pour info, voici les programmes qu'installe EasyPHP :

- Apache : c'est le programme qu'utilisent les serveurs. Il permet au serveur de distribuer des pages web... mais il ne connaît que le HTML !
- PHP : PHP est comme un "plugin" de Apache. Il a besoin d'Apache pour fonctionner, et grâce à lui Apache saura travailler sur des pages PHP. En clair, Apache + PHP = un serveur PHP.
- MySQL : c'est un programme qui va nous être sacrément utile par la suite, mais pour le moment je ne vous en parle pas. Sachez juste que c'est lui qui permet d'utiliser des bases de données. Vous ne savez pas ce qu'est une base de données ? Vous prenez pas la tête, je vous l'expliquerai lorsque le moment sera venu !
- PHPmyAdmin : cela vous permettra de gérer vos bases de données (si ce mot "base de données" vous fait peur, ne craignez rien, on n'en parlera que plus tard).

Il n'est pas important pour le moment de comprendre comment fonctionnent ces programmes. Il y a en fait une chose que vous devez retenir : vous allez devoir télécharger EasyPHP car on va sacrément en avoir besoin par la suite.
C'est un programme discret : une fois qu'il est lancé il reste en fond et pas besoin d'y toucher.

On va maintenant voir comment installer EasyPHP.

Installer EasyPHP

Trève de bavardages, à l'abordage !

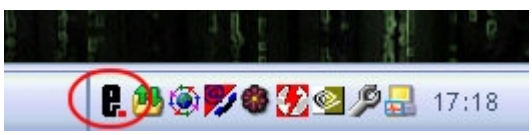
EasyPHP est assez gros. Et pour cause, je vous l'ai expliqué plus haut : il contient plusieurs programmes. Mais ce téléchargement est vraiment indispensable, alors que vous soyez ADSL ou 56K, vous allez devoir vous taper les 10 Mo à télécharger.

easyphp1-7_setup.exe (10,8 Mo)

Installez le programme qui se trouve dans le ZIP, comme vous le feriez pour n'importe quel autre programme.
A la fin, on vous proposera deux options. Moi tout ce que je vous demande c'est de lancer EasyPHP, alors vous pouvez cocher la case "Lancer EasyPHP". Vous pourrez toujours démarrer le programme à l'aide du menu Démarrer.

 Mais... Comment savoir si EasyPHP est démarré ?

Je vous l'avais dit, EasyPHP est discret. Lorsque vous le démarrez, vous pouvez juste voir une icône à droite de la barre des tâches (pas loin de l'horloge) :



Si tout se passe bien, l'icône se met à clignoter. Si vous pointez dessus, vous pourrez lire "EasyPHP (Démarré)". C'est que tout va bien.
Félicitations ! Vous venez d'installer EasyPHP.

Configurer EasyPHP

Dernière étape : il faut configurer EasyPHP. Je vous rassure de suite c'est très rapide et très simple.

Faites un clic droit sur l'icône EasyPHP dans la barre des tâches. Un petit menu s'ouvre :

C'est "Administration" qui va nous servir. Cela permet de configurer EasyPHP.

Pour fermer complètement EasyPHP, cliquez sur "Quitter" en bas.

Tant que EasyPHP et ses programmes (Apache, PHP...) tournent correctement, l'icône de la barre des tâches clignote. Si les programmes sont arrêtés, l'icône ne clignote plus.

Bon, vous vous en doutez, je vais vous demander de cliquer sur "Administration". Et là Ô surprise : ça ouvre une page web. Attention, ne vous y trompez pas : cette page web que vous voyez là est située sur votre disque dur. Il y a marqué dans la barre d'adresse : "http://192.168.0.1", cela veut dire que vous êtes sur votre disque dur. En revanche, si vous voyez "http://www.siteduzero.com", là vous êtes sur un site web, situé sur Internet. Compris ?

Bon, c'est là qu'on va configurer EasyPHP.
Voici un petit aperçu de cette page telle que vous devriez la voir :

Je me suis permis de placer des numéros sur cette image, pour que vous puissiez distinguer facilement à quoi se rapportent les descriptions ci-dessous :

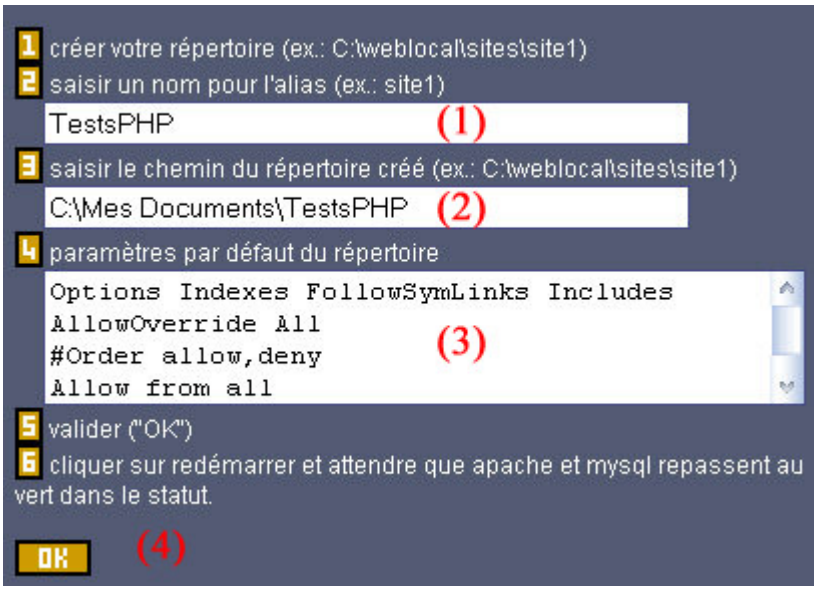
1. Apache > Alias : c'est là qu'on va se rendre pour configurer EasyPHP. Cela permet d'indiquer les sites web que vous avez sur votre disque dur, pour que EasyPHP les reconnaisse.
2. PhpMyAdmin > Gestion BDD : c'est par ici que vous pourrez gérer votre base de données. On verra ce que c'est dans la partie II de ce cours.

En-dessous de "Alias", cliquez sur "ajouter".

On vous demande quelques informations.
C'est là qu'il va falloir vous organiser un peu : peut-être que vous avez déjà un projet de site web, et que vous lisez ce cours pour apprendre à utiliser PHP dans votre site web. Peut-être aussi que vous n'avez pas d'idées pour le moment mais que vous lisez ce cours dans le but d'apprendre PHP, tout simplement. Quoiqu'il en soit, tout au long de ce cours on va avoir l'occasion de faire pas mal de "tests", et je pense que vous auriez intérêt à créer un dossier "Tests PHP" par exemple.

A vous de vous organiser, mais je vous conseille de créer un dossier "Tests PHP" dans "Mes documents", dans lequel vous essaierez de faire ce que je vous apprends tout au long de ce cours. C'est un dossier brouillon quoi.

Il va falloir remplir les champs que vous avez devant les yeux :



1. Nom de l'alias : c'est le nom de votre site. Il faut bien lui donner un nom quoi. Dans notre exemple, on mettra "TestsPHP".
2. Le chemin du répertoire : il faut indiquer dans quel dossier se trouve votre site. Si vous avez fait comme je vous l'ai dit, vous devriez avoir créé un dossier "Tests PHP" dans "Mes documents".

Pour connaître le chemin du répertoire, allez dans "Mes documents", et ouvrez le dossier que vous venez de créer. En haut de la fenêtre vous devriez avoir un champ "Adresse :", qui contient quelque chose du genre "C:\...". Copiez ce chemin, tout simplement.

3. Paramètres par défaut du répertoire : dès que vous voyez des mots bizarres en anglais, vous faites comme moi : vous n'y touchez pas.
4. OK : une fois que tout est rempli, cliquez sur ce petit bouton et c'est bon.

Ensuite, vous faites un clic droit sur l'icône de EasyPHP dans la barre des tâches, et vous cliquez sur "Redémarrer". Ca va relancer les programmes de EasyPHP (en particulier le serveur Apache). Pourquoi les relancer ? Parce que comme ça, les changements que vous venez d'effectuer vont être pris en compte.

Si tout est bon, ça revient à la page de tout à l'heure, mais cette fois vous avez un lien du genre "TestsPHP", à côté de "Vos alias".
C'est là-dessus qu'il faudra cliquer pour accéder à votre site web, stocké sur votre disque dur.

Votre ordinateur est fin prêt à avaler du PHP.
Dès le prochain chapitre on attaque le code : on va commencer à découvrir des instructions PHP. Cela veut dire que vous allez faire vos premières manipulations !

Premiers Pas Avec PHP

Comme le titre du chapitre l'indique, c'est maintenant que vous allez faire vos premiers pas en PHP.

Vous allez découvrir vos premières instructions et la joie des scripts qui font planter votre ordi...
Bah quoi partez pas ?! Vous allez voir, je ne vais pas vous faire faire des trucs compliqués, juste les bases de la programmation PHP.

C'est partiii !

Les Balises PHP

A partir d'ici on va commencer à rentrer dans le code source de vos pages web. Vous êtes censés connaître le langage HTML, comme je vous l'ai demandé dans le premier chapitre. Pour rappel, si jamais vous avez besoin de vous rafraîchir la mémoire, le cours de HTML est disponible ici.

Pour éditer le code d'une page web, vous avez plusieurs solutions. La plus simple, c'est d'utiliser un éditeur de texte tout simple, comme Bloc-Notes. Pour l'ouvrir, faites Démarrer / Programmes / Accessoires / Bloc-notes.
Sinon, il existe des logiciels spécialisés, qui ont l'avantage de colorer le code pour rendre la lecture plus claire. C'est le cas de Dreamweaver de Macromedia par exemple, que j'utilise. Seul problème : ces logiciels sont payants.

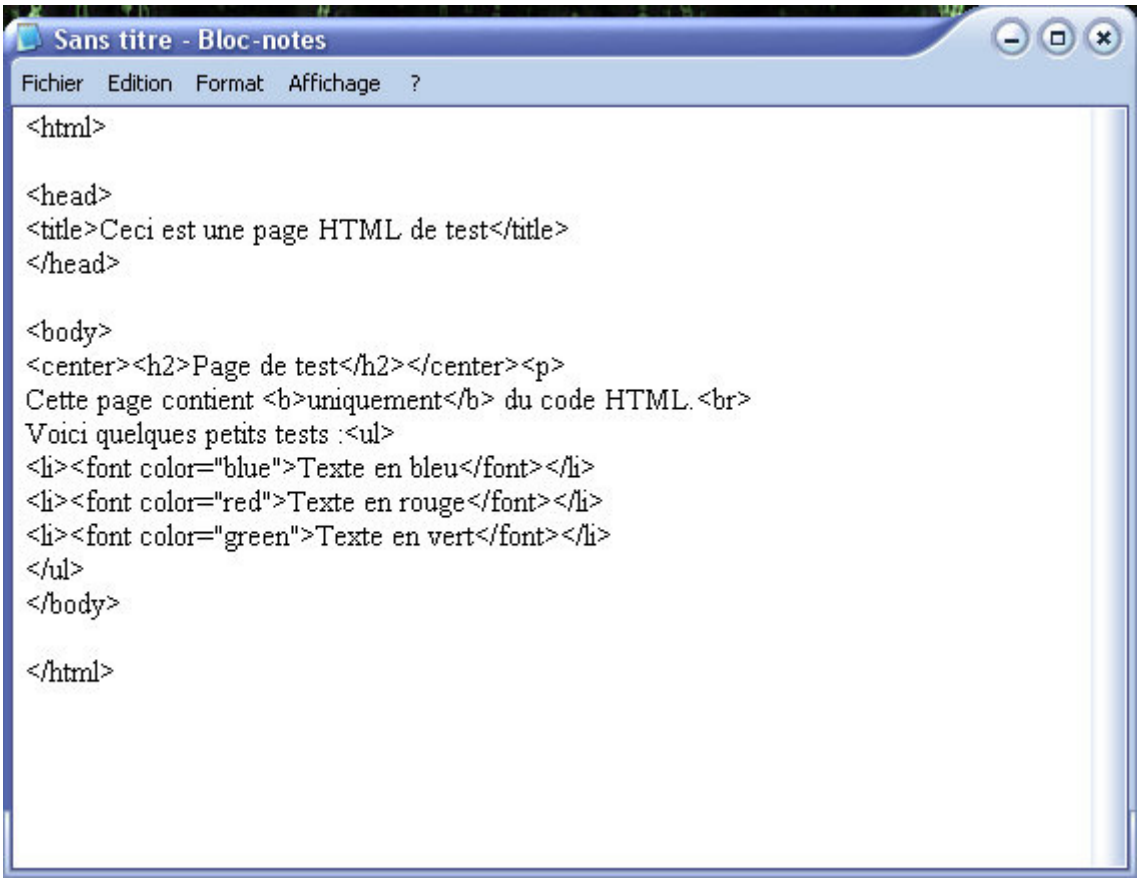
Quoiqu'il en soit rassurez-vous, ça ne change pas du tout la manière dont vous allez apprendre le PHP : les manipulations seront exactement les mêmes pour tout le monde. Et afin qu'on soit tous au même niveau, je vais détailler la procédure à suivre avec Bloc-notes.

On va commencer par créer une page HTML toute simple, car je vous l'ai dit le PHP a toujours besoin du HTML.
Le code ci-dessous ne contient que du HTML, recopiez-le dans Bloc-Notes :

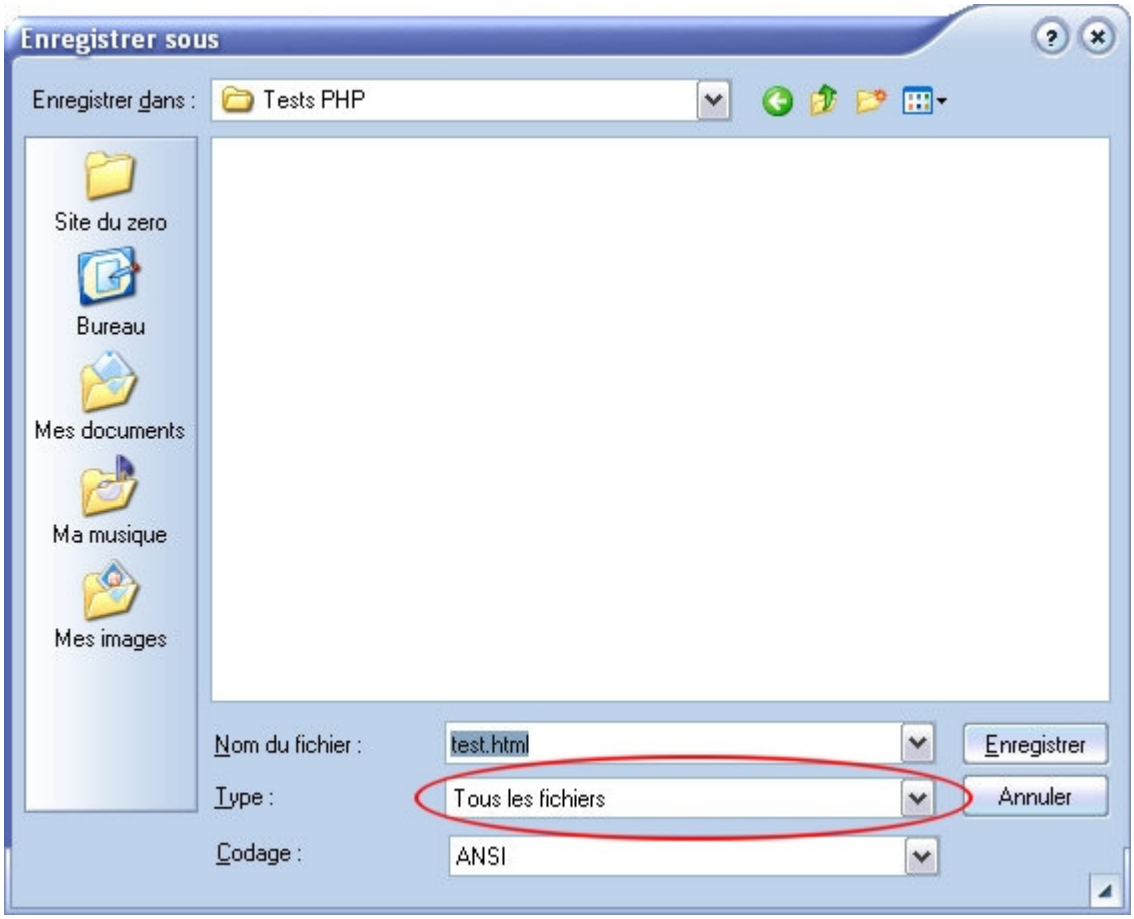
Source 1.3.1 : une page HTML de test

```
<html>

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

<body>
<center><h2>Page de test</h2></center><p>
Cette page contient <b>uniquement</b> du code HTML.<br>
Voici quelques petits tests :<ul>
<li><font color="blue">Texte en bleu</font></li>
<li><font color="red">Texte en rouge</font></li>
<li><font color="green">Texte en vert</font></li>
</ul>
</body>

</html>
```

Ce code doit vous sembler familier vu que vous connaissez le HTML.
Si vous le recopiez dans bloc-notes, vous devriez voir ceci :



Pour enregistrer la page HTML, vous devrez faire Fichier / Enregistrer. Dans la fenêtre qui s'ouvre, sélectionnez en bas "**Type : tous les fichiers**", et enregistrez votre page avec l'extension ".html". Par exemple : "test.html". Vous devriez donc avoir ceci sous les yeux :



Bien, jusque-là je ne vous surprends pas trop, tout ça vous savez le faire.
Vous savez donc que le code source d'une page HTML est constitué de "**balises**", aussi appelées "**tags**". Par exemple <ul> est une balise.

Si je vous parle de cela, ce n'est pas par hasard. C'est que pour utiliser du PHP, on va devoir introduire une nouvelle balise... celle-ci est un peu spéciale. Elle commence par <? et se termine par ?>. C'est dedans que l'on mettra du code PHP, ce que je vais vous apprendre tout au long de ce cours.

Source 1.3.2 : une balise PHP

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

Une chose importante : en général, le code PHP tient sur plusieurs lignes. On peut sans problème agrandir la taille de la balise sur plusieurs lignes. Par exemple, on peut faire ceci :

Source 1.3.3 : une balise PHP agrandie

```
<? // Code PHP ligne 1
// Code PHP ligne 2
// Code PHP ligne 3
// Code PHP ligne 4 ?>
```

Tout ce qu'il faut retenir pour mettre du code PHP, c'est cette balise <? ?>

Il existe d'autres balises pour utiliser du PHP, par exemple : <?php ?>, <?php4 ?> etc... Ne soyez donc pas étonnés si vous en voyez.
Pour ma part j'ai toujours utilisé <? ?>, donc je vous apprendrai à vous servir de la même balise que moi.

On place le PHP au beau milieu du reste du code HTML. Par exemple :

Source 1.3.4 : une page contenant des balises PHP

```
<html>

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

<body>
<center><h2>Page de test</h2></center><p>
Cette page contient du code HTML avec des balises PHP.<br>
<? // Ici on mettra du code PHP ?>
Voici quelques petits tests :<ul>
<li><font color="blue">Texte en bleu</font></li>
<li><font color="red">Texte en rouge</font></li>
<li><font color="green">Texte en vert</font></li>
</ul>
</body>
<? // Encore du PHP
// Toujours du PHP ?>
</html>
```

Bien entendu cette page ne fonctionne pas vu que nous n'avons pas encore mis de code PHP.
Tout ce qu'il vous faut retenir ici, c'est que dès que vous voulez mettre du code PHP, hop, vous ouvrez une balise PHP : <? ?>

Afficher Du Texte

Bon tout ça c'est bien beau, mais il va falloir commencer à mettre du code PHP non ?
Grande nouvelle : c'est maintenant que vous allez apprendre votre première instruction en PHP.

Bon ne vous attendez pas à quelque chose d'extraordinaire, votre PC ne va pas se mettre à danser la samba tout seul.

La fonction que je vais vous apprendre permet d'afficher du texte. Je vais vous faire manipuler d'abord pour que vous voyez ce que ça donne, puis je vous expliquerai en détail comment ça marche.

Ouvrez Bloc-Notes (ou un autre éditeur de texte), et recopiez-y le code ci-dessous :

Source 1.3.5 : afficher du texte en PHP

```
<html>

<head>
<title>Notre première instruction : echo</title>
</head>

<body>
<center><h2>Affichage de texte avec PHP</h2></center><p>

Cette ligne a été écrite entièrement en HTML.<br>
<? echo "Celle-ci a été écrite entièrement en PHP."; ?>

</body>
</html>
```

Enregistrez la page avec l'extension .php, par exemple "affichertexte.php", dans le dossier "Tests PHP" que je vous ai fait créer.

Avant d'enregistrer, surtout vérifiez que **Type : "Tous les fichiers"** est sélectionné, sinon ça va créer un fichier affichertexte.php.txt et ça ne marchera pas !

Pour tester votre page PHP : démarrez EasyPHP si ce n'est déjà fait. Allez dans le menu "Administration", la page d'administration s'ouvre. Là, vous devriez avoir en haut dans "Vos alias" un lien du style "TestsPHP". Cliquez dessus.

Il existe aussi un autre moyen pour tester votre page PHP. Ouvrez votre navigateur (Internet Explorer par exemple), et tapez l'adresse suivante : http://127.0.0.1/alias/
Remplacez "alias" par le nom de l'alias que vous avez créé. Par exemple ici, je devrai taper : http://127.0.0.1/testsphp/

Une page web s'ouvre indiquant tous les fichiers qui se trouvent dans le dossier "Tests PHP". Vous devriez avoir le fichier "affichertexte.php". Cliquez dessus : votre ordinateur génère alors le code PHP puis ouvre la page. Vous avez le résultat devant vos yeux.

Je pense que vous êtes étonnés et surpris de ce que je vous ai fait faire : ça a l'air d'être inutile, et ce n'est pas tout à fait faux. Le code PHP a "écrit" une ligne à l'écran, tout simplement.

Mais euh c'est pas plus simple de l'écrire en HTML ?

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

Bon, comment ça marche ce truc ?

Reprenons la ligne qui nous intéresse, celle qui est en PHP :

```
<? echo "Celle-ci a été écrite entièrement en PHP."; ?>
```

Comme vous le voyez, le code PHP est dans la balise <? ?>
Ce qui nous intéresse est à l'intérieur, il s'agit de :

```
echo "Celle-ci a été écrite entièrement en PHP.";
```

"echo", c'est l'instruction, le mot qui donne un ordre à l'ordinateur. Ici, l'ordre est "Afficher le texte". Ensuite, entre guillemets, il y a le texte à afficher. On met toujours le texte entre guillemets, ça permet à l'ordinateur de repérer ce qu'on lui demande d'afficher.
Enfin, la ligne finit par un point-virgule ;. Ce signe doit être placé à la fin de chaque instruction. A chaque fois que vous écrirez une instruction en PHP, vous devrez écrire un ; à la fin. Cela permet d'indiquer à l'ordinateur que c'est la fin de l'instruction.

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

Si on traduit ce code en français, ça donnerait : *Afficher le texte : "Celle-ci a été écrite entièrement en PHP." **(Fin d'instruction)***

On a aussi le droit de demander d'afficher des balises. Par exemple le code suivant fonctionne :

Source 1.3.6 : des balises dans une instruction echo

```
<? echo "Celle-ci a été écrite <b>uniquement</b> en PHP."; ?>
```

"uniquement" sera affiché en gras grâce à la présence des balises et

Comment faire pour afficher un guillemet ?

Bonne question. Si vous mettez un guillemet, ça veut dire pour l'ordinateur que le texte à afficher s'arrête là. Ca va donc faire planter votre beau code.
La solution consiste à faire précéder le guillemet d'un backslash \ :

Source 1.3.7 : afficher des guillemets dans l'instruction echo

```
<? echo "Celle-ci a été écrite \"uniquement\" en PHP."; ?>
```

Je vous ai à peu près tout dit sur la fonction echo. A vous de vous amuser à écrire n'importe quoi (bon ok c'est pas super drôle comme jeu).
Essayez par exemple de mettre 2 ou 3 instructions echo à la suite (une par ligne). Pour que chacune s'inscrive sur une ligne différente, pensez à mettre une balise
 à chaque fois !
Par exemple, vous pouvez faire :

```
<? echo "Celle-ci a été écrite \"uniquement\" en PHP.<br>"; ?>
```

Je vous rappelle qu'il est possible de mettre des balises HTML dans une instruction echo

Notez qu'il existe une instruction identique appelée "print", qui fait exactement la même chose. Alors laquelle utiliser ? C'est une question de goût, moi j'utilise echo, donc je vous apprendrai à utiliser echo.

Les Commentaires

Bon, mine de rien je viens de vous apprendre pas mal de choses d'un coup, ça doit vous faire un choc. D'accord ce n'était pas extraordinaire, mais vous allez pas tarder à comprendre toute la subtilité de la chose.

Avant de terminer ce chapitre, je tiens à vous parler de quelque chose qui à mes yeux a une très grande importance en PHP, comme dans tout langage de programmation : les commentaires.

Un **commentaire** est un texte que vous mettez pour vous 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 voyez ce texte

Mais alors à quoi sert un commentaire ?

C'est pour vous. Cela permet de vous y retrouver dans votre code PHP, parce que si vous n'y touchez pas pendant des semaines et que vous y revenez, vous risquez d'être un peu perdu.
Vous pouvez écrire tout et n'importe quoi, le tout est de s'en servir à bon escient.

Pour indiquer que vous écrivez un commentaire, vous devez taper 2 slash : //. Tapez ensuite votre commentaire.
Un exemple ?

Source 1.3.8 : des commentaires dans du code PHP

```
<? echo "J'habite en Chine.<br>"; // cette ligne indique où j'habite

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

Je vous ai mis 2 type de commentaires (ils sont écrits en violet normalement) :

- Le premier est à la fin d'une ligne.
- Le second est sur toute une ligne

A vous de voir où vous placez vos commentaires : si vous commentez une ligne précise, mieux vaut mettre le commentaire à la fin de cette ligne. Si vous commentez plusieurs lignes, je vous conseille de placer votre commentaire avant.

Ici les commentaires n'ont pas grande utilité, mais vous verrez comment je les utilise dans les prochains chapitres. Ils vous seront très utiles, et vous apprendrez vite à bien vous en servir.

Vous devez être en train de vous demander vraiment à quoi peut bien servir PHP... Ici c'est vrai, ça n'a pas l'air d'être très utile, ça complique plutôt les choses.
Pourtant, vous allez voir très bientôt quel est l'intérêt de la fonction echo, et vous allez même vous rendre compte cela permet de simplifier votre travail !

Dans le prochain chapitre on va travailler sur un autre élément fondamental en PHP : les variables. Ces petites bébêtes sont vraiment très utiles, vous allez le voir.

Les Variables

Attention, chapitre fondamental !
Les variables sont un élément indispensable dans tout langage de programmation, et en PHP on n'y échappe pas. Ce n'est pas un truc de programmeurs tordus, c'est au contraire pour nous simplifier la vie. Sans elles, vous n'iriez pas bien loin.

Ce chapitre est un peu long, aussi n'hésitez pas à en lire seulement la moitié un jour, puis l'autre moitié un autre jour. Il ne faut pas le prendre à la légère, car vous allez y apprendre des choses vraiment importantes. Vous allez, vers la fin de ce chapitre, commencer à comprendre pourquoi PHP est si apprécié !

Qu'Est-Ce Qu'Une Variable ?

Déjà dans le mot, 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 changer. Mais qu'est-ce que c'est concrètement ?

Une variable, c'est une petite information stockée en mémoire temporairement. Ca n'a pas une grande durée de vie. 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.

C'est à vous de créer des variables. Vous en créez quand ça vous arrange.
Ce qu'il faut retenir, c'est qu'une variable est toujours constituée de 2 choses :

- Son nom : pour pouvoir la reconnaître, vous devez donner un nom à votre variable. Par exemple "age_du_visiteur".
- Sa valeur : c'est l'information qu'elle contient, qui peut changer. Par exemple "17 ans".

Ici, je vous ai donné l'exemple d'une variable appelée "age_du_visiteur" qui a pour valeur "17 ans". On peut modifier quand on veut la valeur de cette variable, faire des opérations dessus etc etc... Et quand on en a besoin, on l'appelle (par son nom), et elle nous dit gentiment la valeur qu'elle contient.

Par exemple vous pouvez demander à un moment :

- Hep ! Toi, la variable age_du_visiteur, que contiens-tu ?
- 17 ans.
- Merci !

Vous allez voir que ces petites bébêtes, même si elles peuvent vous sembler encore un peu floues, seront vraiment indispensables pour votre site en PHP.
Par exemple, vous pourrez retenir temporairement le nom du visiteur. Dans une variable "nom_du_visiteur", vous stockez son pseudo, par exemple "M@teo21". Dès que vous en avez besoin vous pouvez l'utiliser, par exemple pour afficher un message de bienvenue personnalisé : "Salut M@teo21 ! Bienvenue sur mon site !".

Vous vous souvenez comment on fait pour afficher du texte en PHP n'est-ce pas ? La fonction "echo" que je vous ai fait apprendre dans le chapitre précédent va nous être très utile ici pour faire des expériences !

On va maintenant voir comment il faut faire pour utiliser des variables en PHP.

Affectation Et Affichage

On va dans un premier temps **affecter** une valeur à une variable, et ensuite on affichera ce qu'elle contient. Vous allez mieux comprendre l'intérêt d'utiliser des variables.

Affecter Une Valeur À Une Variable

Ici, on va tout simplement créer une variable, et lui donner la valeur qu'on veut. Pour le fun.

Par exemple, si on tapait ceci :

Source 1.4.1 : affectation de texte

```
<?
$pseudo_du_visiteur = "Mateo21";
?>
```

Si on tapait ça, ça créerait une variable :

- dont le nom serait *pseudo_du_visiteur*
- dont la valeur serait *Mateo21*

Notez qu'on ne peut pas mettre d'espaces pour un nom de variable. A la place, utilisez un underscore _ (c'est le symbole sous le chiffre 8 de votre clavier).
Evitez aussi les accents, les cédilles et tout autre symbole pour le nom. PHP ne les apprécie pas trop... En revanche pour la valeur vous pouvez mettre ce que vous voulez

Il y a plusieurs nouveaux élements. D'abord, le symbole Dollar ($) : il précède toujours le nom d'une variable. C'est comme un signe de reconnaissance si vous préférez : ça permet de dire à PHP "J'utilise une variable". Donc vous reconnaîtrez toujours qu'il y a une variable par la présence du symbole Dollar ($).
Ensuite, il y a le signe Egal (=) : celui-là c'est logique, c'est pour dire que $pseudo_du_visiteur est égal à...
A la suite, il y a la valeur de la variable, entre guillemets puisqu'il s'agit de texte.
Enfin, il y a l'inoubliable symbole point-virgule (;), qui permet de terminer l'instruction.

Concrètement, qu'est-ce qu'afficherait le code 1.4.1 ?

Rien du tout ! Eh oui, 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.

Maintenant, une variable n'est pas obligée de contenir du texte. On peut aussi y mettre des nombres ou des booléens !

Bouletquoi ?

Je vais vous expliquer.
Retenez qu'on peut mettre 3 sortes de "données" différentes dans une variable : texte, nombres, ou booléens. Voici comment on les utilise :

- Le texte : ça je viens de vous le montrer. Pour mettre du texte dans une variable, on le place entre guillemets comme ceci :

Source 1.4.1 : affectation de texte

```
<?
$pseudo_du_visiteur = "Mateo21";
?>
```

- Les nombres : la seule différence avec le texte, c'est qu'on ne met pas de guillemets. Regardez :

Source 1.4.2 : affectation d'un nombre

```
<?
$nombre_de_freres = 3;
?>
```

- Ainsi, PHP comprend qu'il s'agit d'un nombre et non pas d'un texte. Donc la seule chose à retenir, c'est que si vous voulez stocker juste un nombre il ne faut pas mettre de guillemets.
- Les booléens : je suppose que la plupart d'entre vous savaient déjà ce qu'étaient le texte et les nombres (du moins j'espère).
  Mais les booléens, c'est probablement quelque chose de nouveau pour vous. En fait, ça sert à exprimer si quelque chose est vrai (true en anglais), ou si c'est FAUX (false en anglais). Il n'y a que deux possibilités.
  En PHP, il faut taper true ou false pour dire à une variable qu'elle vaut vrai ou qu'elle vaut faux.

Pour ne pas confondre avec du texte, il ne faut pas mettre de guillemets (comme pour les nombres quoi). Exemple :

Source 1.4.12 : affectation d'un booléen

```
<?
$je_suis_un_zero = true;
$je_suis_bon_en_php = false;
?>
```

- Ici, j'ai créé deux variables booléennes différentes (pour que vous voyiez bien les deux possibilités). $je_suis_un_zero vaut true (vrai), et $je_suis_bon_en_php vaut false (faux). Ca se comprend assez bien non ?
  Vous vous demandez certainement à quoi peuvent bien servir les booléens ? Ca, je ne peux pas vous le dire maintenant. Vous allez en voir l'utilité un peu plus loin, dans le chapitre sur les conditions.

C'est compris ? On peut mettres 3 types d'éléments dans une variable : texte, nombres et booléens.
Pour le texte, on le met entre guillemets.
Pour les nombres et les booléens, on ne met pas de guillemets.

Si vous avez retenu ça, vous savez ce qu'il faut. On peut passer à la suite

Afficher La Valeur D'Une Variable

Allez, maintenant une petite expérience : on va utiliser la fonction echo avec des variables. C'est très simple à faire regardez :

Source 1.4.3 : afficher la valeur d'une variable

```
<?
$pseudo_du_visiteur = "Mateo21";
echo "$pseudo_du_visiteur";
?>
```

Qu'est-ce que ça affiche ? Eh oui, c'est magnifique, c'est magique : ça écrit Mateo21 !
A vous de faire vos propres essais pour vérifier que ça marche ! Changez la valeur de la variable dans la première ligne, et ça affichera quelque chose de différent ! Une expérience tout bête en somme, mais que vous devez faire pour bien comprendre comment les variables fonctionnent.

Avec l'instruction echo, vous pouvez donc afficher le contenu d'une variable. Mais vous n'êtes pas obligés d'afficher uniquement la valeur de la variable !
Voilà un petit exemple qui peut être très utile :

Source 1.4.4 : une instruction echo utilisant une variable

```
<?
$pseudo_du_visiteur = "Mateo21";
echo "Bonjour $pseudo_du_visiteur !";
?>
```

Vous voyez, dans l'instruction echo on a écrit le texte qu'on voulait (comme on faisait dans le chapitre précédent), mais on a mis au milieu la variable ($pseudo_du_visiteur). Lorsque la page PHP sera générée, $pseudo_du_visiteur sera remplacé par ce qu'il contient.
Du coup, ça affichera : *Bonjour Mateo21 !*

Faites vos essais, en écrivant le texte que vous voulez, en affichant au milieu la valeur de 1, 2, 3 variables... Cette technique (pas bien compliquée) sera très souvent réutilisée dans les chapitres qui suivent, alors apprenez à faire pareil que moi !

Faire Des Calculs Simples

On va maintenant faire travailler votre ordinateur, vous allez voir qu'il encaisse les calculs sans broncher. Eh oui, PHP sait aussi faire des calculs !
Oh je vous rassure, on ne va pas faire des calculs tordus, juste des additions, des soustractions, des multiplications et des divisions. C'est pas trop dur pour vous j'espère ?

Bon, ici on ne va travailler que sur des variables qui contiennent des nombres.
Voici les signes à connaître pour faire les 4 opérations de base (vous les trouverez sur votre pavé numérique, à droite du clavier) :

Symbole	Signification
+	Addition
-	Soustraction
*	Multiplication
/	Division

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

Source 1.4.5 : des calculs simples

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

Allez quoi, boudez pas, un peu de calcul mental ça n'a jamais fait de mal à personne.
Vérifiez mes calculs, comme vous pouvez le voir 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 :

Source 1.4.6 : encore des calculs, toujours des calculs

```
<?
$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, et vous êtes un pro des variables.

Transmettre Des Variables

Un des aspects intéressants de PHP, c'est qu'on peut se transmettre des variables de page en page. Vous allez voir que c'est rudement pratique, par exemple pour transmettre le nom du visiteur. En effet, je vous rappelle que les variables sont détruites une fois que la page PHP est générée. Alors comment récupérer leur valeur dans une autre page ?

Transmettre En Modifiant L'Adresse

Vous avez certainement eu le résultat sous vos yeux un bon nombre de fois. Vous ne vous êtes jamais demandés pourquoi certaines adresses était si longues ?

http://www.monsite.com/infos.php?jour=27&mois=07&annee=2003&titre=Informations

Elles sont là vos variables ! C'est comme ça qu'on fait pour les transmettre d'une page à une autre !

 Comment ça marche ?

Eh bien c'est du pur HTML. Comme vous le savez, pour faire un lien vers une autre page on utilise la balise <a>. Par exemple :

Source 1.4.7 : un lien simple en HTML

```
<a href="http://www.monsite.com/infos.php">Cliquez ici pour accéder aux infos !</a>
```

Eh bien, à la suite du infos.php, il faut écrire un point d'interrogation (?). Ensuite, vous tapez le nom de la variable, un égal, puis sa valeur :

http://www.monsite.com/infos.php?jour=27

Cela va créer une variable un peu particulière : $_GET['jour'] qui aura pour valeur 27 !
Et si vous voulez créer d'autres variables, il vous suffit de les séparer par des & :

http://www.monsite.com/infos.php?jour=27&mois=07&annee=2003&titre=Informations

Ici, 4 variables seront créées. Cela correspondrait à faire les 4 instructions suivantes :

- $_GET['jour'] = 27;
- $_GET['mois'] = 07;
- $_GET['annee'] = 2003;
- $_GET['titre'] = "Informations";

Je reconnais que ces variables ont une forme un peu bizarre, mais ne vous arrêtez pas pour ça.

On va faire un petit exemple pour que vous voyiez ce que ça donne concrètement.
Pour faire ce test, on aura besoin de 2 pages :

- Celle qui contient le lien (<a href="...">)
- Et celle dans laquelle on va récupérer les variables.

Source 1.4.8 : code de appel.php

```
Notez que cette page ne contient que du HTML.<br>
Voici 3 liens vers la page cible.php, avec des variables aux valeurs différentes :<p>

<a href="cible.php?nom=Dupont&prenom=Michel">Lien vers
cible.php?nom=Dupont&prenom=Michel</a><br>
<a href="cible.php?nom=Guichard&prenom=Patrick">Lien vers
cible.php?nom=Guichard&prenom=Patrick</a><br>
<a href="cible.php?nom=Surret&prenom=Coralie">Lien vers
cible.php?nom=Surret&prenom=Coralie</a>
```

Source 1.4.9 : code de cible.php

```
Bonjour !<p>

Votre nom est <? echo $_GET['nom']; ?> , et votre prénom est <? echo $_GET['prenom']; ?> .<p>

Faites un autre essai, <a href="appel.php">cliquez ici</a> pour revenir à appel.php
```

Alors, qu'en pensez-vous ? C'est plutôt sympa non ?
Vous êtes en train d'apercevoir pour la première fois un aspect vraiment génial de PHP : le code source de cible.php est tout petit, et pourtant la page affiche quelque chose de différent à chaque fois ! La page cible.php peut en effet afficher n'importe quoi, sans que vous ayez à changer son code !

Là surtout n'hésitez pas à faire vos propres tests pour vous familiariser avec cette transmission de variables.

Transmettre En Utilisant Un Formulaire

Il y a un autre moyen de transmettre des variables, lui aussi très pratique. Il s'agit d'utiliser un formulaire (vous savez, avec des zones de texte, des cases à cocher, des boutons etc etc...)

En fait, on dédiera un chapitre entier aux formulaires dans la partie III de ce cours de PHP (lol, quand je pense qu'on n'en est qu'à la partie I). En effet, c'est assez vaste et il y a quelques trucs un peu compliqués.
Je n'ai nullement envie de vous embrouiller, on va simplement s'intéresser à l'aspect le plus simple, qui vous permettra déjà de faire quelque chose de pas mal du tout.

L'aspect le plus simple, c'est la zone de texte : 
Comme vous le savez, vous pouvez écrire n'importe quoi dedans. Notre objectif sera de récupérer ce que le visiteur a écrit.

On va fonctionner de la même manière que tout à l'heure, avec une page appel.php (qui contiendra la zone de texte) et une page cible.php (qui affichera ce que vous avez tapé dans la zone de texte).

Source 1.4.10 : code de appel.php

```
Cette page, elle aussi, ne contient que du HTML.<br>
Veuillez taper votre prénom :<p>

<form action="cible.php" method="post">
<center>
<input type="text" name="prenom"> <input type="submit" value="Valider">
</center>
</form>
```

Source 1.4.11 : code de cible.php

```
Bonjour !<p>

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

Si tu veux changer de prénom, <a href="appel.php">clique ici</a> pour revenir à appel.php
```

Quand on récupère les valeurs d'un formulaire, on utilise le préfixe $_POST['xxxx'].
Quand on récupère les valeurs depuis l'adresse (comme on a fait tout à l'heure), on utilise le préfixe $_GET['xxxx']

Là, vous pouvez vous amuser à l'infini à inventer n'importe quel nom (bon ok je reconnais qu'il y a mieux pour s'amuser). Mais bon un peu de sérieux quand même, nous ce qui nous intéresse c'est "Comment que ça marche ce truc ?"
La page appel.php, c'est un formulaire. Si vous avez lu mon cours sur le HTML, vous devriez savoir vous en servir. Au besoin, voici un petit lien qui vous amènera directement vers le chapitre sur les formulaires pour vous rafraîchir la mémoire.

Le seul truc à savoir, c'est que "action" indique la page à afficher (cible.php) lorsqu'on a cliqué sur le bouton, et que le nom de la zone de texte sera le nom de la variable créée. Ici, la balise est :

<input type="text" name="prenom">

Ici le nom de la zone de texte est "prenom".
Dans la page cible.php, une variable $_POST['prenom'] sera créée, qui aura pour valeur ce que vous avez entré dans la zone de texte. C'est une variable un peu particulière, il n'est pas utile de s'y attarder pour le moment. Vous comprendrez comment ça marche un peu plus tard, en attendant grâce à ça vous pouvez faire des trucs sympas.

Les Fonctions

En PHP, vous allez être forcément amenés un jour ou l'autre à faire des calculs, et ceux-ci risquent d'être répétitifs. Dans le chapitre précédent je vous ai montré les calculs de base.

Ici je vais pas vous faire un cours de maths, mais plutôt vous montrer comment automatiser certaines tâches à l'aide de fonctions. Car en PHP comme dans n'importe quel autre langage, si vous vous rendez compte que vous faites quelque chose de répétitif, dites-vous bien qu'il y a forcément plus simple et plus rapide.

Créer Ses Propres Fonctions

 Qu'est-ce qu'une fonction ?

Une **fonction**, c'est une série d'instructions qui retourne une valeur. En gros, si vous avez besoin d'effectuer un calcul un peu long ou complexe et répétitif, vous faites appel à une fonction :
- Toi, la fonction CalculCube, donne-moi le volume d'un cube dont l'arête mesure 4 cm.
La fonction effectue les calculs demandés puis répond :
- Ce cube a un volume de 64 cm^3.

Si vous aviez eu à le faire une seule fois, vous auriez pu vous contenter de faire les calculs comme expliqué dans le chapitre précédent. Mais si vous aviez à le faire 5 fois ? 10 fois ? 100 fois ?
Je vais donc vous montrer par des exemples concrets pourquoi les fonctions vous seront utiles.

1er Exemple : Dis Bonjour Au Monsieur

C'est peut-être un peu fatigant de dire bonjour à chacun de ses visiteurs non ? Ca serait bien que ça le fasse automatiquement ! Les fonctions sont justement là pour nous aider !
Regardez le code ci-dessous :

Source 1.5.1 : dire bonjour plusieurs fois, c'est fatigant

```
<?
$nom = "Sandra";
echo "Bonjour, $nom !<br>";

$nom = "Patrick";
echo "Bonjour, $nom !<br>";

$nom = "Claude";
echo "Bonjour, $nom !";
?>
```

Vous voyez, c'est un peu fatigant à la longue... Alors nous allons créer une fonction qui le fait toute seule à notre place !

Source 1.5.2 : une fonction pour dire bonjour automatiquement

```
<?
function DireBonjour($nom)
{
echo "Bonjour $nom !<br>";
}

DireBonjour("Marie");
DireBonjour("Patrice");
DireBonjour("Edouard");
DireBonjour("Pascale");
DireBonjour("François");
DireBonjour("Benoît");
DireBonjour("Père Noël");
?>
```

Alors qu'y a-t-il de différent ici ? C'est surtout en haut qu'il y a une nouveauté : c'est la fonction. En fait, les lignes en haut permettent de définir la fonction (son nom, ce qu'elle est capable de faire etc...)
Pour créer une fonction, vous devez taper function (ça veut dire fonction en anglais). Ensuite, donnez un nom à votre fonction. Par exemple, celle-ci s'appelle "DireBonjour".

Ce qui est plus particulier après, c'est ce qu'on met entre parenthèses : il y a une variable dedans. Késako ? C'est ce qu'on appelle un **paramètre** : une information dont la fonction a besoin pour travailler. Ici, on doit lui indiquer le nom de la personne pour qu'elle sache à qui s'adresser.

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...)

Ensuite, vous repérez deux symboles curieux : des accolades. En fait, elle permettent de marquer les limites de la fonction. La fonction commence dès qu'il y a un { et se termine lorsqu'il y a un }. Entre les deux, il y a le contenu de la fonction.
Ici, la fonction contient une seule instruction (echo). J'ai fait simple pour commencer mais vous verrez que souvent il y a plusieurs instructions.

Voilà, la fonction est créée, vous n'avez plus besoin d'y toucher. Après, pour faire appel à elle, il suffit d'indiquer son nom, et de préciser ses paramètres entre parenthèses (ici, on doit indiquer le nom). Enfin, il ne faut pas oublier le fameux ; car il s'agit d'une instruction. Par exemple :
```
DireBonjour("Marie");
```

A vous d'essayer ! Créez une page avec cette fonction et dites bonjour à qui vous voulez, vous verrez : ça marche ! (encore heureux).

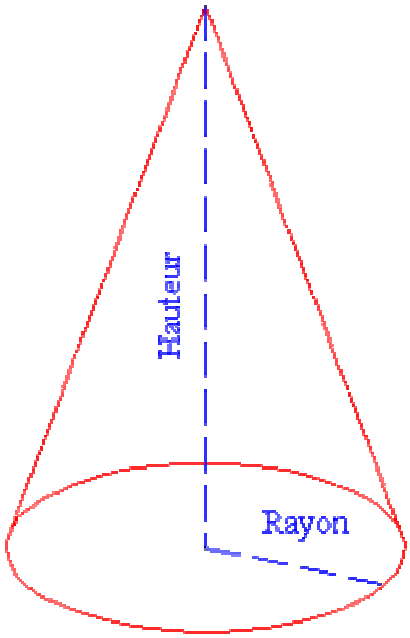
Un conseil pour que vous vous entrainiez sur les fonctions : basez-vous sur mes exemples et essayez de les retoucher petit à petit vous-mêmes pour voir ce que ça donne. Il peut y avoir des fonctions très simples comme des fonctions très compliquées, alors allez-y prudemment.

2ème Exemple : Calculer Le Volume D'Un Cône

Allez on passe à la vitesse supérieure, vous n'avez pas encore vu tout ce qu'on peut faire avec une fonction !

Ici notre fonction va servir à faire un calcul : le calcul du volume d'un cône. Le principe est le suivant : vous donnez le rayon et la hauteur du cône à la fonction, elle travaille et vous renvoie le volume que vous cherchiez.
Ce qui change par rapport à la première fonction qu'on a étudié ? C'est qu'ici, la fonction va retourner une valeur ! Vous allez voir.

Bon tout d'abord il faut connaître la formule pour calculer le volume d'un cône. Vous avez oublié comment on fait ?
Il faut connaître le rayon et la hauteur. Le calcul à faire pour trouver le volume est : rayon * rayon * 3.14 * hauteur * (1/3) (je vous demandais pas de le savoir).



Vous êtes capables de comprendre le code ci-dessous normalement, si vous avez bien suivi dans le chapitre précédent. Seul problème si on a à le faire plusieurs fois, c'est vite répétitif regardez :

Source 1.5.3 : des calculs de volume répétitifs

```
<?
// calcul du volume d'un cône de rayon 5 et de hauteur 2
$volume = 5 * 5 * 3.14 * 2 * (1/3);
echo "Le volume du cône de rayon 5 et de hauteur 2 est : $volume cm<sup>3</sup><br>";

// calcul du volume d'un cône de rayon 3 et de hauteur 4
$volume = 3 * 3 * 3.14 * 4 * (1/3);
echo "Le volume du cône de rayon 3 et de hauteur 4 est : $volume cm<sup>3</sup><br>";
?>
```

En PHP, on ne met pas de virgule pour les nombres décimaux, il faut mettre un point ! Par exemple, il ne faut pas écrire 3,14 mais 3.14 !

Nous allons donc créer une fonction **VolumeCone**, qui va calculer le volume du cône en fonction du rayon et de la hauteur. Cette fonction ne va rien afficher, on veut juste qu'elle nous renvoie le volume qu'on cherche.
Regardez attentivement le code ci-dessous, il présente 2 nouveautés :

Source 1.5.4 : une fonction pour calculer le volume d'un cône

```
<?
// Ci-dessous, la fonction qui calcule le volume du cône
function VolumeCone($rayon, $hauteur)
{
$volume = $rayon * $rayon * 3.14 * $hauteur * (1/3); // calcul du volume
return $volume; // indique la valeur à renvoyer, ici le volume
}

$volume = VolumeCone(3, 1);
echo "Le volume d'un cône de rayon 3 et de hauteur 1 est de $volume";
?>
```

Regardez bien la fonction, dedans il y a l'instruction :
return $volume;
Cette instruction indique ce que doit renvoyer la fonction. Ici la fonction renvoie le volume. Si vous aviez tapé return 15; , ça aurait à chaque fois affiché un volume de 15 (ce qui est un peu débile j'en conviens, mais faites l'essai !).

Alors ici la fonction n'est pas du tout utilisée de la même manière. Elle renvoit une valeur, donc on met cette valeur dans une variable :
$volume = VolumeCone(3, 1);
Ensuite, on peut afficher ce que contient la variable à l'aide d'une instruction echo.

Allons ne faites pas cette tête-là voyons. Je vous ai dit que la fonction renvoyait une valeur, eh bien quand vous écrivez VolumeCone(3, 1), PHP remplace ça par la valeur que retourne la fonction ! (ici ça renvoit 9.42)

Autre nouveauté, la fonction prend deux paramètres : le rayon et la hauteur. Comme vous le voyez, on peut mettre plusieurs paramètres, il suffit de les séparer par des virgules.

Les possibilités de création de fonctions sont quasi-infinies. Il est clair que normalement vous n'allez pas avoir à créer de fonction qui calcule le volume d'un cône (qui est assez fou pour faire ça ?). Tout ce que je vous demande en fait ici, c'est de comprendre qu'une fonction c'est très pratique et ça peut vous faire gagner du temps.
Accessoirement, si vous comprenez un peu comment fonctionne mon code c'est bien, si vous essayez de créer une ou deux fonctions de test chez vous c'est encore mieux. Pas besoin d'en savoir plus, en fait nous allons voir que PHP a déjà prévu le coup : il existe des centaines de fonctions toutes prêtes !

Transformer PHP En Horloge Parlante

Si je vous ai parlé des fonctions, ce n'est pas vraiment parce que vous allez avoir besoin de créer les vôtres tout de suite. En fait, ce que vous venez d'apprendre vous servira, mais bien plus tard.
Vous venez de voir comment est consituée une fonction, comment elle marche, à quoi elle peut servir. Mais bien souvent, vous n'aurez pas à vous prendre la tête à créer vos propres fonctions. En effet, en PHP il y a des centaines de fonctions toutes prêtes que vous pouvez utiliser !

Ces fonctions sont très pratiques et très nombreuses. En fait, c'est en partie là qu'est la force de PHP : ses fonctions sont vraiment excellentes.
J'ai en fait remarqué que, pratiquement à chaque fois que je m'apprêtais à écrire une fonction, celle-ci existait déjà.

Il faut surtout retenir qu'il existe deux types de fonctions :

- Celles qui effectuent des actions, et ne renvoient aucune valeur.
- Celles qui, après plusieurs calculs, renvoient une valeur (ce sont les plus fréquentes)

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

- Une fonction qui permet de rechercher et de remplacer des mots dans une variable
- Une fonction qui envoie un fichier sur un serveur
- Une fonction qui permet de créer des images miniatures (aussi appelées thumbnails)
- Une fonction qui envoie un mail avec PHP (très pratique pour faire une newsletter !)

- Une fonction qui permet de modifier des images, y écrire du texte, tracer des lignes, des rectangles etc...
- Une fonction qui crypte des mots de passe.
- Une fonction qui renvoie l'heure, la date...
- Etc etc...

Pratiquement à chaque fois, il faudra indiquer des paramètres à la fonction pour qu'elle sache sur quoi travailler.Nous allons nous intéresser rapidement à la fonction qui renvoie l'heure et la date. Il s'agit de date.C'est une fonction "toute prête". Vous n'avez pas à écrire "function" (le code de la fonction). En effet, vu que c'est une fonction toute prête, PHP sait déjà comment il faut faire (pas besoin de lui réexpliquer).
Vous avez juste besoin de donner un paramètre. Pour la fonction *date*, voici les 5 paramètres les plus utilisés :

 Attention ! Respectez les majuscules/minuscules, c'est important !

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

date est une fonction vraiment impressionnante, elle prend en fait beaucoup plus de paramètres (une trentaine). Vous verrez tout ça dans la partie III quand on détaillera plus la fonction.
Bon, si vous voulez afficher l'année, c'est très simple :

Source 1.5.5 : afficher l'année

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

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

Source 1.5.6 : afficher la date et l'heure

```
<?
// 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.
Normalement, quand vous avez cliqué sur "Essayer !", vous avez dû avoir la date et l'heure exactes (n'hésitez pas à essayer chez vous).

Si l'heure n'était pas bonne, sachez que c'est le serveur qui donne l'heure. Et le serveur de ce site étant situé à Paris, vous comprendrez le décalage horaire si vous habitez au Canada.

L'étude de fonctions comme celle-ci durera tout une partie du cours, et ce sera une partie très intéressante (car généralement les fonctions sont simples à utiliser et permettent de faire des choses très pratiques !).

En attendant, ce chapitre touche à sa fin, et il nous reste encore 2 chapitres à traiter pour finir la première partie ("Les bases de PHP"). Je reconnais que ces chapitres ne vous permettent pas encore de créer un site web super méga pratique génial en PHP. Mais patience, les bonnes choses arriveront bientôt, et vous verrez que tout ce que je vous apprends maintenant va vous être très utile dans quelques temps.
Aussi ne vous découragez pas et continuez à bien suivre cette première partie, ce que vous apprenez va bientôt prendre tout son sens.

Au fait, vous vous souvenez que, pour le calcul du volume du cône, on a utilisé le nombre Pi (3,14). Oui mais voilà, ce n'est pas très précis.
Heureusement vous savez quoi ? Il existe une fonction en PHP qui retourne la valeur de Pi. Cette fonction ne prend pas de paramètre, pour l'appeler tapez juste Pi(). Essayez d'afficher ce nombre vous verrez !

Fin du chapitre sur les fonctions !
Plus que quelques chapitres et vous pourrez vous vanter de ne plus être un débutant total en PHP !

Les Conditions

Ce chapitre est d'une importance capitale. En effet, vous serez très souvent amenés à employer des "conditions".
Nota : j'aurais dû appeler ce chapitre "Structures conditionnelles", mais j'ai préféré simplifier le titre, j'espère que vous me comprendrez .

Bon, ça sert à quoi d'utiliser des conditions ?

Eh bien, on a parfois besoin d'afficher des choses différentes en fonction de certaines données.
Par exemple, si c'est le matin, vous voudrez dire "bonjour" à votre visiteur, si c'est le soir il vaudrait mieux dire "bonsoir".

C'est là qu'interviennent les conditions. Elles permettent de donner des ordres différents à PHP selon le cas. Pour notre exemple, on lui dirait : *Si c'est le matin, affiche "Bonjour". Sinon, si c'est le soir, affiche "Bonsoir"*. Vous allez le voir, les conditions c'est vraiment la base pour rendre votre site dynamique, c'est à dire d'afficher des choses différentes en fonction du visiteur, de l'heure de la journée, de la date etc etc...
Voilà pourquoi ce chapitre est si important !

Allez, on y va !

La Structure De Base : If... Else

On appelle ça une structure parce que ça a une "forme" particulière.
Celle que je vais vous apprendre à utiliser maintenant, c'est la principale à connaître. Heureusement qu'il n'y a pas 50 façons d'utiliser des conditions.

Pour étudier la structure If... Else, nous allons suivre le plan suivant :

1. Les symboles à connaître : il va d'abord falloir retenir quelques symboles qui permettent de faire des comparaisons. Soyez attentifs car ils vous seront utiles pour les conditions.
2. La structure If... Else : c'est le gros morceau. Là vous allez voir comment fonctionne une condition avec If... Else. Inutile de vous dire qu'il est indispensable de bien comprendre cela.
3. Des conditions multiples : on compliquera un peu nos conditions. Vous allez voir en effet qu'on peut utiliser plusieurs conditions à la fois.
4. Le cas des booléens : nous verrons ensuite qu'il existe une façon particulière d'utiliser les conditions quand on travaille sur des booléens. Si vous ne savez pas ce que sont les booléens, revoyez le chapitre sur les variables.
5. L'astuce bonus : parce qu'il y a toujours un bonus pour récompenser ceux qui ont bien suivi jusqu'au bout.

Les Symboles À Connaître

Juste avant de commencer, je dois vous montrer les symboles que l'on sera amenés à utiliser. Je vais vous faire un petit tableau avec ces symboles et leur signification, essayez de bien les retenir ils vous seront utiles !

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, et il ne faut pas confondre ça avec le simple = que je vous ai appris dans le chapitre sur les variables. Ici, le double égal sert à tester l'égalité, à dire "*Si c'est égal à...*"
Dans les conditions, on utilisera toujours le double égal (==)

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

La Structure If... Else

Voici ce qu'on doit mettre dans l'ordre pour utiliser une condition :

1. Pour introduire une condition, on utilise le mot "If", qui en anglais signifie "Si".
2. On ajoute à la suite entre parenthèses la condition en elle-même (vous allez voir que vous pouvez inventer une infinité de conditions).
3. Enfin, comme pour les fonctions, on ouvre des accolades à l'intérieur desquelles on mettra les instructions à exécuter si la condition est remplie.

Puisqu'un exemple vaut toujours mieux qu'un long discours :

Source 1.6.1 : un exemple de condition

```
<?
if ($age <= 12)
{
echo "Salut gamin !";
}
?>
```

Ici, on demande à PHP : *Si la variable $age est inférieure ou égale à 12, affiche "Salut gamin !"*

Vous remarquerez que dans la quasi-totalité des cas, c'est sur une variable qu'on fait la condition. Dans notre exemple, on travaille sur la variable $age. Ce qui compte ici, c'est qu'il y a deux possibilités : soit la condition est remplie (l'âge est inférieur ou égal à 12 ans) et alors on affiche quelque chose ; sinon, eh bien on saute les instructions entre accolades, on ne fait rien.

Bon on peut quand même améliorer notre exemple. On va afficher un autre message si l'âge est supérieur à 12 ans :

Source 1.6.2 : une condition avec else

```
<?
$age = 8;

if ($age <= 12) // SI l'âge est inférieur ou égal à 12
{
echo "Salut gamin ! Bienvenue sur mon site !<br>";
$autorisation_entrer = "Oui";
}

else // SINON
{
echo "Ceci est un site pour enfants, vous êtes trop vieux pour pouvoir entrer. Au revoir
!<br>";
$autorisation_entrer = "Non";
}

echo "Avez-vous l'autorisation d'entrer ? La réponse est : $autorisation_entrer";
?>
```

Bon comment marche ce code ? Tout d'abord, j'ai mis plusieurs instructions entre accolades (il ne faut pas oublier que vous pouvez mettre plusieurs instructions).

Ensuite, vous avez remarqué que j'ai ajouté le mot "else", qui signifie en anglais "sinon". En clair, on demande : *Si l'âge est inférieur ou égal à 12 ans, fais ceci, sinon fais cela*.

Essayez ce bout de code chez vous, en vous amusant à modifier la valeur de $age (sur la première ligne). Vous allez voir que le message qui s'affiche change en fonction de l'âge que vous indiquez !

Bien entendu, vous mettez les instructions que vous voulez entre accolades. Ici par exemple j'ai affiché un message, et j'ai donné une valeur différente à la variable $autorisation_entrer, ce qui pourrait nous servir par la suite. Par exemple :

Source 1.6.3 : une autre condition avec elseif

```
<?
if ($autorisation_entrer == "Oui") // SI on a l'autorisation d'entrer
{
// instructions à exécuter quand on est autorisé à entrer
}

elseif ($autorisation_entrer == "Non") // SINON SI on n'a pas l'autorisation d'entrer
{
// instructions à exécuter quand on n'est pas autorisé entrer
}

else // SINON (la variable ne contient ni Oui ni Non, on ne peut pas agir)
{
echo "Euh, je ne connais pas ton âge, tu peux me le rappeler s'il te plaît ?";
}
?>
```

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

Bon la principale nouveautée ici, c'est le mot-clé "elseif" qui signifie "Sinon si". Dans l'ordre, PHP rencontre les conditions suivantes :

1. Si $autorisation_entrer est égal à "Oui", tu exécutes ces instructions...
2. Sinon si $autorisation_entrer est égal à "Non", tu exécutes ces autres instructions...
3. Sinon, tu redemandes l'âge pour savoir si on a ou non l'autorisation d'entrer.

Au fait, au départ, une variable ne contient rien. Sa valeur est vide, on dit qu'elle vaut NULL, c'est-à-dire rien du tout.
Pour vérifier si la variable est vide, vous pouvez taper : if ($variable == NULL)...

Des Conditions Multiples

Vous devez vous dire : "*Rhalala, qu'est-ce qu'il va encore nous sortir ce vieux tordu ?*"

Bah, on peut toujours faire plus compliqué, vous devriez commencer à avoir l'habitude.
Je pouvais difficilement passer à côté des conditions multiples, car elles sont très pratiques. Allez, un dernier petit effort et on a bientôt fini.

Ce qu'on va essayer de faire, c'est de donner plusieurs conditions à la fois. Pour cela, on aura besoin de nouveaux mots-clés. Voici les principaux à connaître :

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

Le symbole équivalent pour OR est consitué de 2 barres verticales. Pour taper une barre verticale, appuyez sur la touche "Alt Gr" et "6" en même temps (clavier français), ou "Alt Gr" et "&" (clavier belge).
Bah oui faut pas oublier que selon le pays le clavier change.

La première colonne contient le mot-clé en anglais, la troisième son équivalent en symbole. Les deux fonctionnent aussi bien, mais je vous recommande d'utiliser le mot-clé de préférence, c'est plus "facile" à lire (j'espère que vous connaissez un peu l'anglais quand même) Servez-vous de ces mots-clés pour mettre plusieurs conditions entre les parenthèses. Voici un premier exemple :

Source 1.6.4 : une condition avec AND

```
<?
if ($age <= 12 AND $sexe == "garçon")
{
echo "Bienvenue sur le site de Captain Mégakill !";
}
elseif ($age <= 12 AND $sexe == "fille")
{
echo "C'est pas un site pour les filles ici, retourne jouer à la Barbie !";
}
?>
```

C'est tout simple en fait et ça se comprend très bien : si l'âge est inférieur ou égal à 12 ans et que c'est un garçon, on lui permet d'accéder au site de son superhéro préféré.
Sinon, si c'est une fille dont l'âge est inférieur ou égal à 12 ans, on l'envoie gentiement ballader (hum hum, m'accusez pas de sexisme hein, c'était juste pour l'exemple).

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

Source 1.6.5 : une condition avec OR

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

Le Cas Des Booléens

Si vous regardez bien le dernier code source (avec $autorisation_entrer), vous trouvez pas qu'il serait plus adapté d'utiliser des booléens ?

On a parlé des booléens dans le chapitre sur les variables. Vous vous souvenez ?
Ce sont ces variables qui valent soit true (vrai) soit false (faux). Eh bien, les booléens sont particulièrement utiles avec les conditions ! Voici comment on teste une variable booléenne :

Source 1.6.8 : tester un booléen

```
<?
if ($autorisation_entrer == true)
{
    echo "Bienvenue petit Zér0 :o)";
}
elseif ($autorisation_entrer == false)
{
    echo "T'as pas le droit d'entrer !";
}
?>
```

Voilà, jusque-là rien d'extraordinaire. Vous avez vu que je n'ai pas mis de guillemets pour true et false (comme je vous l'ai dit dans le chapitre sur les variables).

Mais un des avantages des booléens, c'est qu'ils sont particulièrement adaptés aux conditions.
Pourquoi ? Parce qu'en fait vous n'êtes pas obligés d'ajouter le == *true*. Quand vous travaillez sur une variable booléenne, PHP comprend très bien ce que vous avez voulu dire :

Source 1.6.9 : une autre façon de tester un booléen

```
<?
if ($autorisation_entrer)
{
    echo "Bienvenue petit Zér0 :o)";
}
else
{
    echo "T'as pas le droit d'entrer !";
}
?>
```

PHP comprend qu'il faut qu'il vérifie si $autorisation_entrer vaut true. Avantages :

- C'est plus rapide à écrire pour vous.
- Ca se comprend bien mieux.

En effet, si vous "lisez" la première ligne, ça 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 si on veut vérifier si le booléen vaut faux. Comment on fait avec la méthode courte hein ?

Il y a un petit mot qui permet de vérifier juste si la variable vaut false : NOT. On écrit :

if (NOT $autorisation_entrer)...

C'est une autre façon de faire. Si vous préférez mettre *if ($autorisation_entrer == false)* c'est tout aussi bien, mais moi je trouve que c'est plus lisible d'utiliser la méthode "courte".

L'Astuce Bonus

Avec les conditions, il y a une astuce à connaître.
Sachez que les deux codes ci-dessous donnent exactement le même résultat :

Ces deux codes donnent le même résultat

```php
<?
if ($variable == 23)
{
echo "<b>Bravo !</b> Vous avez trouvé le nombre mystère !";
}
?>
```

```php
<?
if ($variable == 23)
{
?>

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

<?
}
?>
```

Comme vous le voyez, dans la seconde colonne on n'a pas utilisé de echo. En effet, il vous suffit d'ouvrir l'accolade ({), puis de fermer la balise php (?>), et vous pouvez mettre tout le texte à afficher que vous voulez en HTML !
Rudement pratique quand il y a de grosses quantités de texte à afficher, et aussi pour éviter d'avoir à se prendre la tête avec les backslash devant les guillemets (\").
Il vous faudra toutefois penser à refermer l'accolade après (à l'intérieur d'une balise PHP bien entendu).

Et après ça, ma foi, il n'y a rien de particulier à savoir. Vous allez rencontrer des conditions dans la quasi-totalité des exemples que je vous donnerai par la suite.
Vous ne devriez pas avoir de problèmes normalement pour utiliser des conditions, il n'y a rien de bien difficile. Contentez-vous de reprendre le schéma que je vous ai donné pour la structure If... Else, et de l'appliquer à votre cas. Nous aurons d'ailleurs bientôt l'occasion de pratiquer un peu, et vous verrez que les conditions sont souvent indispensables.

Une Alternative Pratique : Switch

En théorie, les if... elseif... else que je viens de vous montrer suffisent pour traiter n'importe quelle condition.

Mais alors pourquoi tu viens nous compliquer la vie avec encore un nouveau truc ?

Pour vous faire comprendre l'intérêt de Switch, je vais vous donner un exemple un peu lourd avec les if et elseif que vous venez d'apprendre :

Source 1.6.6 : des elseif qui se répètent

```
<?
if ($note == 0)
{
echo "Tu es vraiment un gros Zér0 !!!";
}

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";
}
?>
```

Je ne peux pas vous le cacher, cet exemple est tiré du script PHP que j'ai écrit pour les Q.C.M. en fin de chapitre (bon c'est un peu simplifié bien entendu).
Comme vous le voyez, c'est lourd, long, et répétitif. Dans ce cas, on peut utiliser une autre structure plus souple : c'est Switch.

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

Source 1.6.7 : switch pour mieux organiser la condition

```
<?
$note = 10;

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

case 0: // dans le cas où $note vaut 0
echo "Tu es vraiment un gros Zér0 !!!";
break;

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

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

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

case 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 ça vous fera de l'entraînement !

Tout d'abord, il y a beaucoup moins d'accolades (elles marquent seulement le début et la fin du switch). "case" signifie "cas". Dans le 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 etc...). Cela signifie : *Dans le cas où la valeur est 0... Dans le cas où la valeur est 10...*

Avantage : on n'a plus besoin de mettre le double égal ! Défaut : ça ne marche 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. Mais contrairement aux *elseif*, PHP ne s'arrête pas là et continue à lire les instructions des case qui suivent ! case 12, case 16 etc...

Pour empêcher cela, utilisez l'instruction break;. L'instruction "break" demande à PHP de sortir du switch. Dès que PHP tombe sur break, il sort des accolades et donc il ne lit pas les "case" qui suivent. En pratique, on utilise très souvent un break car sinon PHP lit des instructions qui suivent et qui ne conviennent pas.
Essayez d'enlever les break dans le code 1.6.7, vous allez comprendre pourquoi ils sont indispensables !

 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, et quand on a une série de conditions à analyser, on préfère utiliser Switch pour rendre le code plus clair.

Vous êtes en train d'assimiler sans le savoir les fondements de la programmation PHP qui détermineront avec quel "style" vous allez coder par la suite.
En effet, on peut parler de "style" de programmation car chaque programmeur va présenter son code différemment (le résultat est le même mais la façon de faire est parfois différente). Ici, je vous présente ma manière de faire, donc au début vous allez avoir un peu mon style, mais rassurez-vous petit à petit vous allez vous créer le vôtre.

Quoiqu'il en soit, c'est en ce moment-même que vous apprenez le plus de choses, et il ne faut surtout pas décrocher, d'autant plus qu'on en a presque fini avec les bases !

Les Boucles

On a bientôt fini la Partie I : les bases de PHP !
Ceci est l'avant-dernier chapitre. C'est une des dernières connaissances "de base" à acquérir avant que vous puissiez commencer à découvrir l'aspect vraiment intéressant de PHP.

Normalement, si vous avez bien compris les conditions, ce devrait être un chapitre facile à avaler (et à digérer).

Une Boucle Simple : While

 Qu'est-ce qu'une boucle ?

Une boucle, c'est une structure qui fonctionne sur le même principe que les conditions (if... else). D'ailleurs vous allez voir qu'il y a pas mal de similitudes avec le chapitre sur les conditions.
Concrètement, une boucle permet de répéter plusieurs fois des instructions. En clair, c'est un gain de temps, c'est très pratique et bien souvent indispensable.

On peut faire un schéma si vous voulez :

Voici ce qui se passe dans une boucle :

1. Comme d'habitude, les instructions sont d'abord exécutés dans l'ordre, de haut en bas (flèche rouge)
2. A la fin des instructions, on retourne à la première (flèche verte)
3. Et on recommence à lire les instructions dans l'ordre (flèche rouge)
4. Et on retourne à la première (flèche verte)
5. etc etc...

Le seul hic dans ce schéma, c'est que ça ne s'arrête jamais ! Les instructions seraient réexécutées 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 (ouf !).

Voici comment faire avec une boucle simple : While.

Source 1.7.1 : une boucle While

```
<?
while ($continuer_boucle == "oui")
{
// instructions à exécuter dans la boucle
}
?>
```

"While" peut se traduire par "Tant que". Ici, on demande à PHP : *TANT QUE $continuer_boucle est égal à "oui", exécuter ces instructions :*

Les instructions qui sont répétées en boucle se trouvent entre les accolades { et }. Mais bon là je vous apprends rien, vous commencez à avoir l'habitude de voir des accolades de partout.

Et puis voilà, ma foi vous savez tout.
Ceci dit, je vais quand même vous montrer 1 ou 2 exemples d'utilisation de boucles, pour que vous voyiez à quoi ça peut servir...

Pour notre premier exemple, on va supposer que vous avez été punis et que vous devez recopier 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 prenait des heuuuures. Maintenant, avec PHP, on va faire ça en un clin d'œil.

Regardez ce code :

Source 1.7.2 : une punition vite rédigée grâce aux boucles

```
<?
$nombre_de_lignes = 1;

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

La boucle pose la condition : *TANT QUE $nombre_de_lignes est inférieur à 100*
Dans cette boucle, il y 2 instructions :

- Le echo, qui permet d'afficher du texte en PHP. A noter qu'il y a une balise HTML
 à la fin : c'est pour aller à la ligne. Vu que vous connaissez le HTML, ça n'a rien de surprenant : chaque phrase sera écrite sur une seule ligne (et non pas à la suite, si vous enlevez le
 vous verrez).
- Une instruction bizarre ensuite : $nombre_de_lignes++; Késako ? 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 que l'on a ajouté 1 à la variable).

A chaque fois qu'on fait une boucle, la valeur de la variable augmente : 1, 2, 3, 4... 98, 99... Dès que la variable atteint 100, on arrête la boucle. Et voilà, on a écrit 100 lignes en un clin d'œil.
Et si la punition avait été plus grosse, pas de problème ! Il suffirait de changer la condition (par exemple mettre "TANT que c'est inférieur à 500" pour l'écrire 500 fois).

Il faut TOUJOURS s'assurer que la condition sera au moins remplie 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 efforts.

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

C'est vrai. Je peux difficilement vous dire à quoi ça va vraiment nous servir, mais sachez que ça sera très utile dans la partie II de ce cours. En effet, nous serons très souvent amenés à répéter plusieurs fois des instructions et la boucle While nous sera alors très utile !
Je vous demande juste pour le moment de pratiquer et de comprendre comment ça marche.

Bon, un autre exemple pour le fun ?
On peut écrire de la même manière une centaine de lignes, mais chacune peut être différente (on n'est pas obligés d'écrire la même chose à chaque fois).
Cet exemple devrait vous montrer que la valeur de la variable augmente à chaque passage dans la boucle :

Source 1.7.3 : des lignes numérotées

```
<?
$nombre_de_lignes = 1;

while ($nombre_de_lignes <= 100)
{
echo "Ceci est la ligne n°$nombre_de_lignes<br>";
$nombre_de_lignes++;
}
?>
```

Voilà, c'est tout bête, et cet exemple ressemble beaucoup au précédent. La particularité là, c'est qu'on affiche à chaque fois la valeur de $nombre_de_lignes (ça vous permet de voir que sa valeur augmente petit à petit).

Pour info, l'astuce que je vous avais donnée dans le chapitre sur les conditions marche aussi ici : vous pouvez fermer le tag PHP ?>, écrire du texte en HTML, puis réouvrir le tag PHP <?
Ca vous évite d'utiliser une instruction echo.
On aura l'occasion d'utiliser cette astuce de nombreuses fois dans la partie II.

Une Boucle Plus Complexe : For

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

For est un autre type de boucle, qui produit exactement le même résultat mais qui est adapté à un type particulier de boucles. Dans tous les cas, vous pouvez utiliser un While, ça marche à tous les coups. Pour ma part, je préfère toujours utiliser un While, mais je veux que vous voyiez rapidement le For pour que vous ne soyez pas étonnés si vous en rencontrez un jour.

Alors, comment ça marche un For ? Ca ressemble beaucoup au While, mais c'est la première ligne qui est un peu particulière. Pour que vous voyiez bien la différence avec le While, je reprends exactement l'exemple 1.7.3, mais cette fois avec un For :

Source 1.7.4 : l'exemple 1.7.3 avec un For

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

Que de choses dans une même ligne !
Bon, vous vous en doutez, je ne vais vous expliquer que la ligne du for, le reste n'a pas changé.

Après le mot for, il y a des parenthèses (si si je vous jure !).
Dans ces parenthèses, il y a 3 éléments, séparés par des point-virgules ;
Décrivons chacun de ces éléments :

- Le premier sert à l'**initialisation**. C'est la valeur que l'on donne au départ à la variable (ici elle vaut 1).
- Le second, c'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, la boucle s'arrête.
- Enfin, le troisième c'est l'**incrémentation**, qui vous permet d'ajouter 1 à la variable.

Les codes 1.7.3 et 1.7.4 donnent donc exactement le même résultat.
A votre avis, lequel des deux est le plus adapté dans ce cas ? C'est plutôt le For, car comme vous le voyez tout est prévu pour faire tenir ça dans une ligne.

 Comment savoir lequel prendre quand je dois choisir entre un While et un For ?

While marche à tous les coups.
For ne marche que quand on a un nombre qui s'incrémente, comme on a fait ici. Donc For est parfois plus adapté, mais personne ne vous tuera si, comme moi, vous préférez utiliser un While tout le temps.

Croyez-moi, les boucles c'est vraiment très pratique !
Grâce à elles, il y a des scripts PHP que l'on peut écrire en quelques lignes de code et qui pourtant effectuent beaucoup de calculs !

Vous aurez en particulier l'occasion de vous servir des boucles lorsque vous attaquerez la partie II : la base de données. D'ailleurs, c'est dans pas longtemps, vu qu'on a presque terminé les bases du PHP !

Les Tableaux (array)

Nous entamons ici un aspect très important du PHP : les array.
Vous allez voir qu'il s'agit de variables "composées", que l'on peut imaginer sous la forme de tableau.

On peut faire énormément de choses avec les array, et leur utilisation n'est pas toujours très facile. En réalité, un connaisseur en PHP sera peut-être un peu surpris de trouver ce chapitre dans "les bases du PHP".
Et pourtant, si je fais cela il y a bien une raison : en comprenant ce chapitre, vous n'aurez quasiment aucune difficulté à comprendre la base de données (et c'est légèrement le thème de la partie II de ce cours).

Seulement, pour ne pas trop compliquer les choses, j'ai décidé de séparer le chapitre en 2 : ici nous verrons les bases, juste le strict nécessaire.
Dans la partie III, vous retrouverez les array, et vous apprendrez à faire plein de choses avec.

Mais trève de bavardages, à l'abordaaaage !

Tableaux Numérotés

 Mais euh, c'est quoi un array au juste ?

Un array, c'est une variable. Mais une variable un peu spéciale.

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

Source 1.8.1 : une bonne vieille variable

```
<?
$prenom = "Nicole";
echo "Bonjour $prenom !"; // Cela affichera : Bonjour Nicole !
?>
```

Ce qui peut se matérialiser sous la forme :

Nom	Valeur
$prenom	Nicole

Ici, nous allons voir qu'il est possible d'enregistrer plein d'informations dans une seule variable (bien plus que "Nicole").
C'est très facile à imaginer. Regardez par exemple ce tableau :

$prenoms	
Numéro	**Valeur**
0	François
1	Michel
2	Nicole
3	Véronique
4	Benoît
...	...

$prenoms est un **array** : c'est ce que j'appelle une variable "tableau". Elle n'a pas qu'une valeur mais plusieurs valeurs (vous pouvez en mettre autant que vous voulez).
Dans un array, les valeurs sont rangées dans des "cases" différentes. Ici, nous travaillons sur un array numéroté.

Attention ! Un array numéroté commence toujours à la case n°0 !
Ne l'oubliez jamais, ou vous risquez de faire des erreurs par la suite...

Pour afficher "Véronique" par exemple, il ne faudra pas juste marquer $prenoms (PHP ne sait pas dans quelle case chercher !). Il va falloir lui dire :

Affiche-moi le contenu de $prenoms dans la case n°3

Et comment on lui dit ça ?

Il faut écrire le nom de la variable, suivi du numéro entre crochets. Pour afficher "Véronique", on utilisera l'instruction :

Source 1.8.2 : récupérer Véronique dans l'array

```
<?
echo $prenoms[3];
?>
```

C'est tout bête !
Par contre si vous oubliez de mettre les crochets, ça ne marchera pas (ça renverra "Array"...). Donc dès que vous travaillez sur des array, vous êtes obligés d'utiliser les crochets pour indiquer dans quelle "case" on doit aller chercher l'information.

Reste maintenant à voir comment créer un array. C'est un peu particulier, il faut utiliser la fonction *array*.
Cette exemple vous montre comment créer l'array $prenoms :

Source 1.8.3 : créer

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

L'ordre a beaucoup d'importance. Le premier élément ("François") aura le n°0, ensuite Michel le n°1 etc etc...
Et puis ma foi, c'est aussi simple que cela. Vous avez vu comment créer un array, et comment afficher le contenu d'une case de l'array.

Je vous propose maintenant de faire un petit script pour résumer. Il doit afficher tout le contenu de notre array $prenoms.
On va donc d'abord commencer par créer cet array comme nous venons juste le voir. Puis nous utiliserons une boucle. On peut se servir d'un while ou d'un for (ça marche tout aussi bien). Là je trouve qu'un for est plus approprié, regardez :

Source 1.8.4 : lister tous les prénoms

```
<?
// On crée notre array $prenoms
$prenoms = array ("François", "Michel", "Nicole", "Véronique", "Benoît");

// Puis on fait une boucle pour tout afficher :
for ($numero = 0; $numero < 5; $numero++)
{
    echo $prenoms[$numero]; // affichera $prenoms[0], $prenoms[1] etc...
    echo "<br>"; // pour aller à la ligne
}
?>
```

Magique, n'est-ce pas ?

Tableaux Associatifs

Bon, alors là on va pas traîner dessus 50 ans pour rien.
C'est exactement pareil que ce qu'on vient de voir, sauf qu'au lieu de repérer les "cases" par des numéros, on va nommer ("étiqueter") ces cases.

Par exemple, supposons que je veuille, dans un seul array, enregistrer les coordonnées de quelqu'un (nom, prénom, adresse, ville etc...). Si l'array est numéroté, comment savoir que le n°0 c'est le nom, le n°2 l'adresse ?...

C'est là que deviennent utiles les tableaux associatifs. Pour les créer, on utilisera la fonction *array* comme tout à l'heure, mais on va mettre "l'étiquette" devant chaque information :

Source 1.8.5 : un array associatif

```
<?
// On crée notre array $coordonnees
$coordonnees = array (
    "Prénom" => "François",
    "Nom" => "Dupont",
    "Adresse" => "3, rue du Paradis",
    "Ville" => "Marseille");
?>
```

Note importante : il n'y a qu'une seule instruction (un seul point-virgule). J'aurais pu tout mettre sur la même ligne, mais rien ne m'empêche de séparer ça sur plusieurs lignes pour que ça soit plus facile à lire.

Vous remarquez qu'on met une flèche (=>) pour dire "associé à". Par exemple, on dit "Ville associé à Marseille".

Et pour afficher le contenu de cet array ?

Eh bien c'est sensiblement pareil que tout à l'heure. On utilisera des crochets, mais on mettra souvent des apostrophes à l'intérieur (ce n'est pas obligatoire mais je préfère vous donner une bonne habitude de suite).
Par exemple, pour extraire la ville, on devra taper $coordonnees['Ville'].

Voici un exemple qui fonctionne (encore heureux) :

Source 1.8.6 : afficher le contenu d'un array associatif

```
<?
// On crée notre array associatif :
$coordonnees = array (
    "Prénom" => "François",
    "Nom" => "Dupont",
    "Adresse" => "3, rue du Paradis",
    "Ville" => "Marseille");

// Puis si je veux afficher la ville, je ferai :
echo $coordonnees['Ville'];
?>
```

Les array associatifs seront très importants dans la partie II de ce cours. En effet, dans la base de données vous aurez bien besoin de ce que vous venez d'apprendre !

Et voilà ! On a terminé la partie I !!!
Vous ne le savez peut-être pas, mais vous avez appris énormément de choses. En fait, vous venez

d'apprendre ce que j'estime le plus dur : le début. Au début, on ne sait rien et il faut s'accrocher pour comprendre des choses qui ont l'air de ne servir à rien. Vous en êtes arrivés au bout : félicitations !

A côté, tous les prochains chapitres devraient vous paraître agréables et simples à lire.
Continuez comme ça, vous êtes sur la bonne voie. Vous allez bientôt maîtriser le PHP comme des pros !

Présentation De MySQL

Nous voici enfin dans la seconde partie. Vous vous attendez à quelques "changements", non ?

Tout d'abord, il faut le dire, vous n'êtes plus de gros débutants. Vous avez certainement l'impression de ne pas être capables de créer un site web en PHP...
Et c'est vrai, mais pourtant tout ce que vous venez d'apprendre est très important, et c'est à partir de maintenant qu'on va vraiment pouvoir créer des scripts en PHP !

Et attention : pas des petits scripts. En fait, vous saurez faire à la fin de cette partie la plupart des scripts que vous rencontrez sur des sites web : système de news, commentaires, forum, livre d'or et j'en passe. Les parties suivantes, elles, vous aideront à améliorer la qualité de vos scripts et à faire des manipulations plus avancées (c'est très intéressant, mais bon on n'en est pas encore là).

Allez, il est temps de faire les présentations.

Euh... Qui C'Est Celui-Là ?

C'est MySQL, un système de base de données.


Base de quoi ?

Oui je sais, encore des mots qui font peur... C'est particulièrement lourd d'ailleurs comme nom : "base de données". Ne vous étonnez donc pas si je me permets de l'abréger par BDD (Base De Données).


Vous pourrez trouver aussi l'abrévation *SGBD* (Système de Gestion de Base de Données), qui est plus correcte. Mais mon abrévation en 3 lettres est plus courte, donc je garde la mienne, na !

La **base de données** est un système qui enregistre des informations. Un peu comme un fichier texte ? Non, pas vraiment. Ce qui est très important ici, c'est que ces informations sont toujours **classées**.
Et c'est ça qui fait que la BDD est si pratique : c'est un moyen simple de ranger des informations.


Et si je préfère rester bordélique ? Si j'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 ok, mais il me semblait que je n'en aurais besoin que très rarement.
Grave erreur ! Vous allez le voir, 99% du temps on range ses informations dans une base de données.

Pour le 1% restant, on pourra enregistrer dans un fichier, ce que nous verrons plus tard car on en a rarement besoin.

Imaginez par exemple une armoire, dans laquelle chaque dossier est à sa place.
Quand tout est à sa place, c'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 (par exemple des informations sur vos visiteurs), il vous sera très facile après de récupérer ce que vous cherchez.

PHP Travaille Avec MySQL

Jusqu'ici je ne vous ai présenté qu'un "personnage" : c'est PHP. Je fais exprès d'utiliser cette image de personnage, car je la trouve bien appropriée.
Jusqu'ici, on n'a fait que discuter avec PHP. On lui demandait par exemple "Combien font 2 + 2 ?", "Répète cette phrase 20 fois" etc... Bref, tout ça vous connaissez.

Eh bien maintenant, dans cette partie, on va s'adresser à quelqu'un d'autre : c'est MySQL, votre base de données.


Atchoum ! Euh, si je comprends bien, tu veux nous faire apprendre "autre chose" que le PHP ? Tu crois pas qu'on en a assez bavé là comme ça ?!

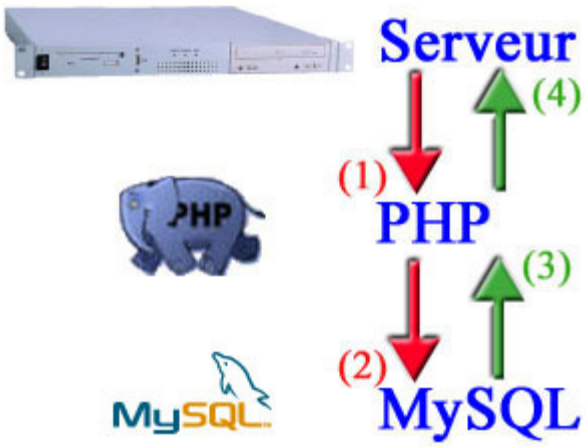
J'étais sûr que vous diriez ça !
Alors non, je vous rassure, je ne m'amuse pas à vous faire souffrir. Bien au contraire j'essaie de faire au plus simple. Seulement, vu que l'on s'adresse à une autre "personne", eh bien il va falloir lui parler différemment :

- Pour demander quelque chose à PHP, il fallait lui parler en PHP.
- Pour demander quelque chose à MySQL, il va falloir lui parler en... SQL !

Vous voyez vous commencez à comprendre.
Alors, avant que vous alliez chercher une chaise et une corde pour abréger vos souffrances, je tiens à vous rassurer : le SQL n'a rien à voir avec le PHP. C'est beaucoup beaucoup plus simple, et en plus cette fois on va lui parler avec des "phrases" (en anglais of course).

Seulement, pour compliquer un petit peu l'affaire (sinon c'est pas rigolo), on ne va pas pouvoir parler à MySQL directement. Eh non, seul PHP peut le faire !
C'est donc PHP qui va faire l'intermédiaire entre vous et MySQL. On devra demander à PHP : "Va dire à MySQL de faire ceci.".

Je crois qu'un petit schéma ne serait pas de refus...



Ca vous rappelle les bons souvenirs du premier chapitre, non ?

Ici on ne voit pas le client, on s'intéresse surtout à ce que le serveur fait lorsqu'il doit génèrer une page PHP.
Voici ce qu'il peut se passer lorsque le serveur a reçu une demande d'un client qui veut poster un message sur vos forums :

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

Voilà en gros comment on peut schématiser ça. Je n'ai pas mis le client pour ne pas vous embrouiller, mais il est clair qu'il aurait fallu le mettre tout en haut du schéma (c'est lui qui fait appel au serveur, comme nous l'avons vu dans le tout premier chapitre).

Bon, eh bien maintenant que nous avons fait les présentations, il va falloir voir comment est organisée une base de données (très très important).

Structure D'Une Base De Données

Oulah oulah ! Surtout faites très attention à ce qui va suivre ! C'est indispensable pour bien comprendre la base de données !

Et pis c'est pas parce que le titre vous donne la nausée que vous devez vomir sur le clavier.

Bon allez, un peu de sérieux, ce qui suit est => **VITAL** <=
Ce n'est pas compliqué (ouf !), mais ce sera une des rares fois où je vous demanderai de retenir du vocabulaire.

En effet, avec la BDD 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.
Ecoutez-moi attentivement, et n'hésitez pas à lire lentement, plusieurs fois si c'est nécessaire.

Je vous demande d'imaginer ceci :

- **La base**, c'est l'armoire. 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, dans le langage MySQL, 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 infos sur vos visiteurs, un autre qui contient les messages postés sur votre forum...
- Mais que contient une table ? C'est là 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, voici à quoi peut ressembler le contenu d'une table appelée "visiteurs" :

Table "visiteurs"

Numéro	Pseudonyme	E-mail	Age
1	Kryptonic	kryptonic@free.fr	24
2	Serial_Killer	serialkiller@unitedgamers.com	16
3	M@teo21	top_secret@siteduzero.com	18
4	Bibou	bibou557@laposte.net	29
...	...	...	...

Ce tableau est donc le contenu d'une table (le tiroir).
Les champs dans cet exemple sont : "Numéro", "Pseudonyme", "E-mail" et "Age".
Chaque ligne est une entrée. Ici, il y a 4 entrées, mais une table peut très bien en contenir 100, ou 1 000, ou même 100 000 ! (je vous souhaite d'avoir autant de visiteurs).

Très souvent, on crée un champ "Numéro", aussi appelé "ID". Comme nous le verrons plus tard, il est très pratique de numéroter ses entrées, même si ce n'est pas obligatoire.

Et pour finir, voici l'indispensable schéma pour que tout ça soit clair :

Il est interdit de se moquer de mon schéma !
(et puis d'abord c'est Xplosif qui a choisi les couleurs)

Bon de toute manière, l'essentiel c'est que vous compreniez qui contient qui.
Comme vous le voyez, on peut mettre autant de tables que l'on veut dans une base (ce qui fait qu'en général une seule base suffit).
Dans chaque table, les données sont enregistrées sous la forme d'un tableau, comme nous l'avons vu plus haut.

Pour vous donner quelques exemples concrets, voici quelques tables utilisées sur ce site web :

- *news :* stocke toutes les news qui sont affichées à l'accueil.
- *livre_or :* stocke tous les messages postés sur le livre d'or.
- *forum :* stocke tous les messages postés sur le forum.
- *newsletter :* stocke les adresses e-mails de tous les visiteurs inscrits à la newsletter.

Voilà, vous devriez commencer à comprendre pourquoi vous allez avoir besoin d'une BDD sur votre site.
Si quelque chose ne vous paraît pas clair, si vous avez l'impression de mélanger un peu "bases", "tables", "champs", "entrées", relisez de nouveau cette partie. Il faut que vous soyez capable de reproduire le schéma tout seul sur un bout de papier.

Hep ! J'Ai Une Question !

Avant de terminer le chapitre, voici une question que l'on se pose fréquemment quand on lit ce genre de chapitres sur MySQL.
Je suis sûr qu'il y a quelque chose qui vous titille dans ce chapitre. Ne mentez pas, tout débutant a ce problème, moi-même j'ai été bloqué quand j'ai appris le PHP, justement parce que je voyais pas bien ce que c'était une base de données.
Comme je ne veux pas qu'il vous arrive pareil, je vais essayer d'éclaircir les points sombres !


T'es gentil tu nous présentes tes jolis tableaux, tes bases, tes tables, tes champs etc... Mais je vois pas ce que c'est concrètement moi ça !? Où MySQL enregistre-t-il les données ?

Question typique, je dois avouer que la première fois c'est très troublant. On vous parle de quelque chose qui n'a pas l'air concret.
En fait, tout ce que je viens de vous montrer, c'est une façon de "visualiser" la chose. Il faut que vous imaginiez que ce sont des tableaux, parce que c'est la meilleure représentation qu'on peut se faire d'une base de données.

Mais concrètement, quand MySQL enregistre des informations, il les écrit bien quelque part. Oui comme tout le monde, il enregistre dans des **FICHIERS**.


Mais où sont ces $#%@#$ de fichiers ?!

Réponse : ils sont dans le dossier où MySQL est installé. Vous devriez trouver ces fichiers dans le dossier :
C:\Program Files\EasyPHP\mysql\data

Eh bah vous savez quoi ? On s'en fout que ça soit là !
Dans la pratique, on n'ira jamais toucher à ces fichiers directement. On demandera TOUJOURS à MySQL d'enregistrer, ou d'aller lire des choses. Après, c'est lui qui se débrouille pour classer ça comme il veut dans ses fichiers.

Et c'est bien ça le gros avantage de la base de données : pas de prise de tête pour le rangement des informations. Vous demandez à MySQL de vous sortir toutes les news de votre site enregistrées de Février à Juillet, il va lire dans ses fichiers, et vous ressort les réponses.
Vous vous contentez de "dialoguer" avec MySQL. Lui il se charge du sale boulot, c'est-à-dire ranger vos données dans ses fichiers.

Si vous avez bien compris et retenu le schéma, que vous avez suivi sans trop de mal ce chapitre et que vous avez tout juste au QCM, c'est que vous savez ce qu'il faut.
Cependant, tout ceci doit vous paraître un peu flou. C'est tout à fait normal. Heureusement dans le chapitre suivant nous allons pas mal manipuler, ce qui devrait vous aider à mieux comprendre tout cela.

PhpMyAdmin

Nous allons maintenant faire des manipulations sur une base de données. Vous allez "voir" ce que peuvent contenir une base et ses tables.
Pour cela, nous allons nous servir d'un système très pratique que beaucoup de sites utilisent : PhpMyAdmin.

PhpMyAdmin est livré avec EasyPHP, vous allez donc pouvoir l'utiliser tout de suite.
La quasi-totalité des hébergeurs permettent d'utiliser PhpMyAdmin. Renseignez-vous auprès de votre hébergeur pour savoir comment y accéder. Par exemple si vous êtes chez Free, l'adresse de PhpMyAdmin est http://sql.free.fr. Vous aurez très certainement besoin d'un login et d'un mot de passe.

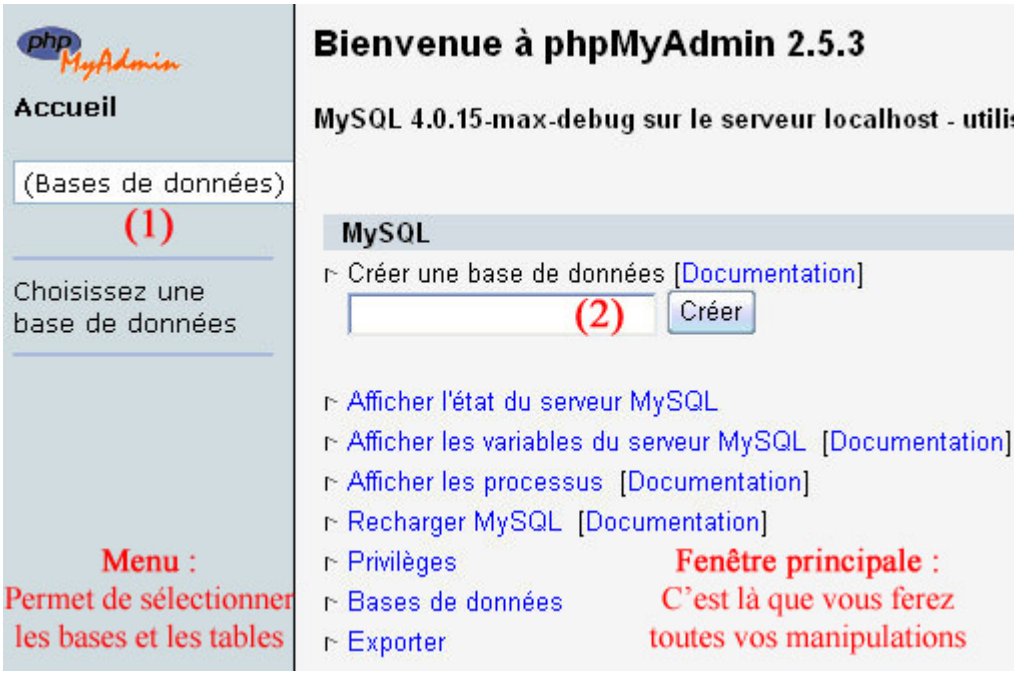
Concrètement, PhpMyAdmin est un ensemble de pages PHP. Ce n'est pas un programme, mais des pages PHP toutes prêtes dont on se sert pour gagner du temps.
On commence donc simplement : on ne va pas coder dans ce chapitre, pour le moment on va simplement manipuler.

La première chose que je vous demanderai de faire, c'est d'ouvrir PhpMyAdmin.
Comment ça "comment on fait" ?
Démarrez EasyPHP, ouvrez la page "Administration", et là... vous vous souvenez ? Je vous rappelle à quoi ressemble la page "Administration" :

Pour accéder à PhpMyAdmin, cliquez sur "Gestion BDD" (marqué d'un petit (2) sur mon image). Une nouvelle fenêtre s'ouvre. Ca y est, vous êtes dans PhpMyAdmin.

Créer Une Table

L'accueil de PhpMyAdmin ressemble à ceci :



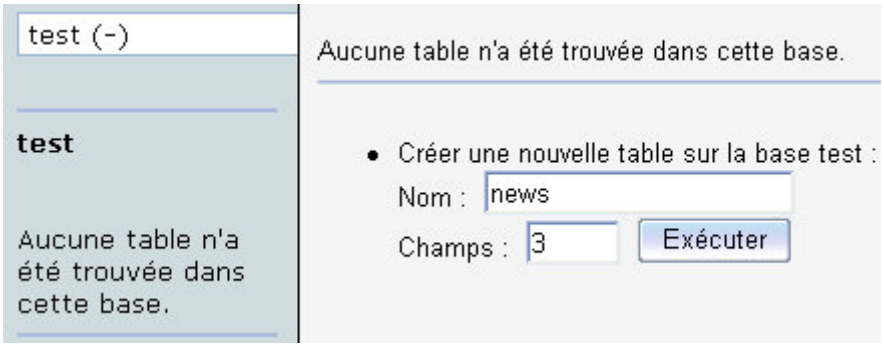
Vous avez 2 endroits importants :

1. Liste des bases : dans ce menu déroulant sont listées vos bases de données. Le nombre entre parenthèses, c'est le nombre de tables qu'il y a dans la base.

2. Créer une base : tapez un nom pour votre base de données, cliquez sur "Créer" et hop ! C'est fait.

Pour le moment, 2 bases sont déjà créées : "mysql" et "test". Ne touchez pas à la base Mysql, elle contient des informations importantes pour le fonctionnement de Mysql.
Nous, on va se servir de la base "test". Ouvrez donc cette base en cliquant sur le menu déroulant à gauche et en choisissant "test".

On vous indique à gauche qu'aucune table n'a été trouvée dans la base. Et si on en créait une ?
On va par exemple créer une table "news" qui contiendra 3 champs :



Cliquez sur "Exécuter".
La table n'est pas immédiatement créée, il faut maintenant indiquer le nom des champs et les données qu'ils peuvent contenir.
On va faire simple car c'est juste pour tester. On va donc créer 3 champs pour cette table :

- id : comme bien souvent, vous allez devoir créer un champ appelé "id". C'est le numéro d'identification. Grâce à lui, toutes vos entrées seront numérotées, ce qui est bien pratique. Il y aura ainsi la news n°1, n°2, n°3 etc...
- titre : ce champ contiendra le titre de la news.
- contenu : enfin, ce champ contiendra la news en elle-même.

Soyons clairs : je ne suis pas en train de vous apprendre à créer un système de news pour votre site. Ca on verra un peu plus tard. D'ailleurs, si on avait voulu faire ça bien on aurait aussi créé un champ "date", mais bon ne compliquons pas les choses inutilement.
Pour le moment on veut juste faire joujou.

Vous devriez avoir ceci sous les yeux :

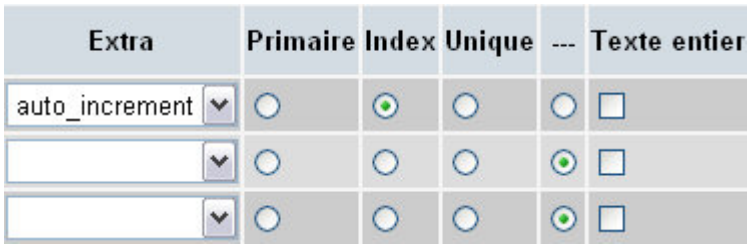
Vous remplissez à gauche le nom du champ, au milieu le type de champ, et à droite la taille maximale du champ.

 Mais qu'est-ce qu'un type de champ ?

Un champ peut contenir du texte, des nombres, des dates etc... Il faut donc définir quel type de données contiendra le champ.
Voici les principaux types de données que vous avez besoin de connaître (il y en a beaucoup d'autres) :

- INT : nombre entier. Il y a plusieurs variantes, selon la grandeur des nombres que ça peut comporter. Dans l'ordre, il y a TINYINT (très petit, c'est-à-dire 255 maximum), SMALLINT (jusqu'à 30 000), MEDIUMINT (8 000 000), INT (2 000 000 000), BIGINT (vraiment beaucoup !).
- TEXT : du texte. Là encore il y a plusieurs variantes, ça fonctionne de la même manière. A vous de choisir celui qui vous paraît le plus adapté.
- DATE : date de la forme "YYYY-MM-DD", "YY-MM-DD" ou "YYMMDD" (c'est le format américain, eh oui !)
- TIME : l'heure, de la forme "HH:MM:SS" ou "HHMMSS" ou "HHMM" ou "HH".
- DATETIME : mélange la date et l'heure, de la forme "YYYY-MM-DD HH:MM:SS"
- BLOB : plus particulier, ce type est rarement utilisé. Il permet de stocker des fichiers dans la base de données. Vu que c'est un cas particulier, on n'en parlera pas de suite, mais il faut que vous sachiez que ça existe.

Il reste à voir les quelques options qui sont proposées à droite de l'écran pour chaque champ :



Pour le premier champ, id, j'ai mis "auto_increment" pour Extra. Je vous recommande de le faire pour tous vos champs "id" : ainsi, le numéro de champ augmentera tout seul à chaque fois que vous rajouterez une entrée. Ca évite des prises de tête en plus...
J'ai aussi mis "Index" pour "id", je vous recommande de le faire pour tous vos champs id, ça accélèrera les recherches dans votre table.

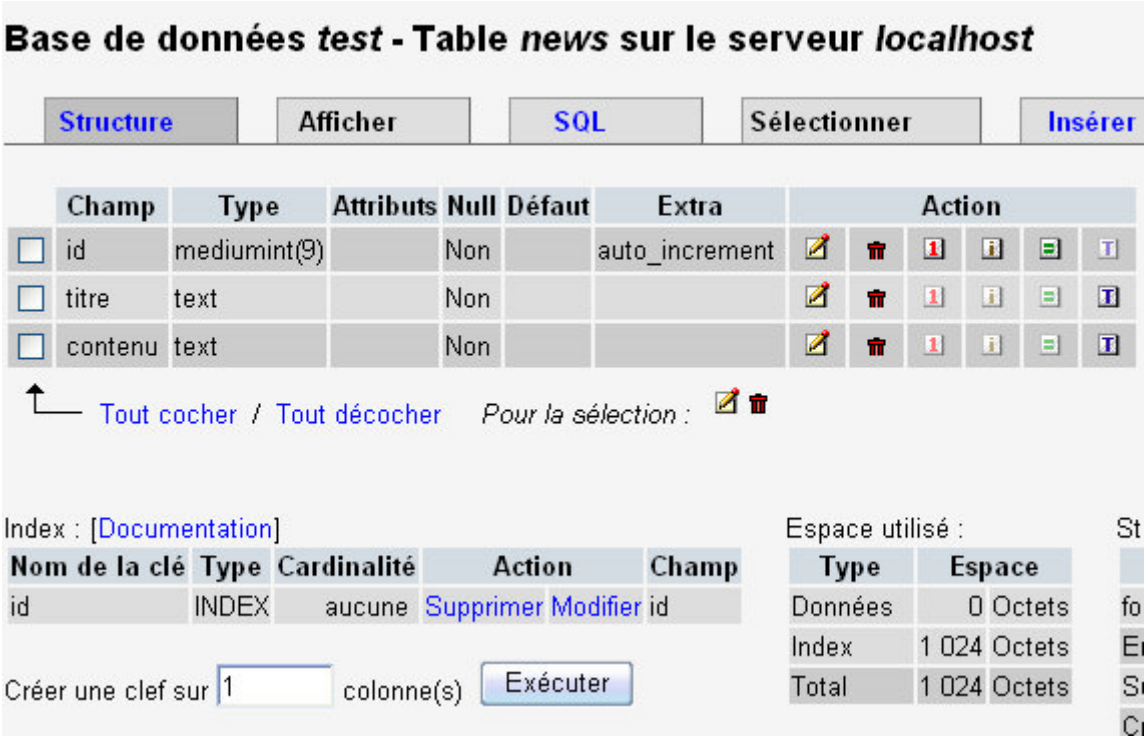
Voilà, je ne m'étends volontairement pas sur tout ça, vous en savez largement assez pour créer une table. Il est inutile de détailler toutes les possibilités, on y passerait des heures. Copiez mon modèle à chaque fois que vous créez une table et ça sera bon.
Cliquez enfin sur "Sauvegarder", et ça y est. Ouf ! On a créé une table.

Modifier Une Table

A gauche de votre écran, la table "news" devient visible :

Si vous cliquez sur "news", ça affichera à droite la structure de la table.
Si vous cliquez sur la petite image de tableau à gauche, ça affichera le contenu de la table.

Pour l'instant la table est vide. Si vous affichez la structure de la table, vous devriez voir ceci en haut :



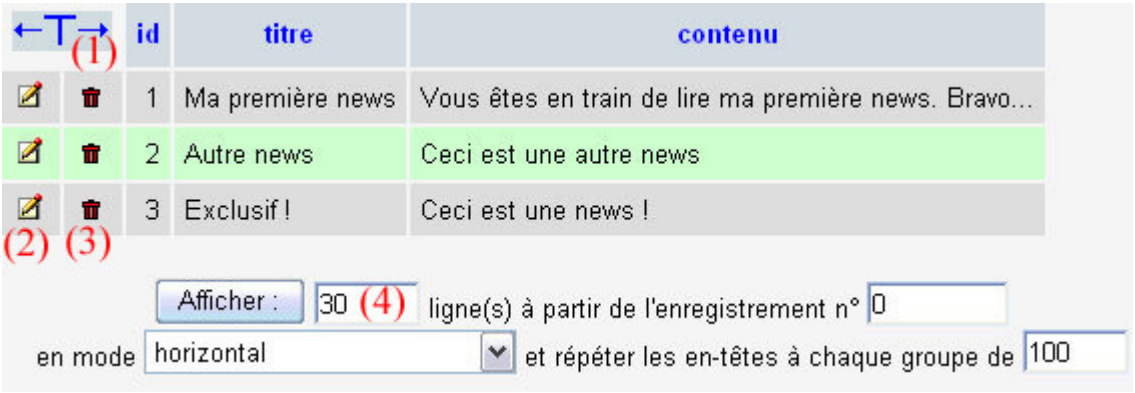
Rien de bien intéressant à toucher ici, si ce n'est les onglets en haut : "Structure", "Afficher", "SQL" etc etc... Cela vous amènera vers différentes options que nous verrons plus loin.

Nous allons rentrer des informations (des entrées) dans cette table. Cliquez sur l'onglet "Insérer" en haut. Vous pouvez maintenant créer une entrée. Faites comme moi :

Pour id, je n'ai rien mis car, je vous le rappelle, on avait indiqué "auto_increment". Le nombre sera calculé tout seul par Mysql, ne vous en occupez pas.
Indiquez simplement le titre et le contenu de votre news, puis cliquez sur "Exécuter".

Recommencez 1 ou 2 fois, en faisant la même manipulation et en laissant le champ "id" vide.

Maintenant, on va afficher ce que contient la base. Pour cela, cliquez sur l'onglet "Afficher" en haut, ou bien cliquez sur la petite image en forme de tableau à gauche de l'écran
Le contenu de la table s'affiche sous vos yeux ébahis.



Vous repérez ici les champs : id, titre et contenu. Cette table a 3 entrées, et comme vous pouvez le voir Mysql a bien fait les choses puisque les numéros d'id se sont créés tous seuls.

1. Afficher tout le texte : si vous cliquez sur le T majuscule, cela affichera la totalité du texte. Vous remarquerez sur mon image que si le texte est trop long, PhpMyAdmin le coupe. Avec ce bouton vous verrez tout le texte.
2. Modifier l'entrée : cette petite image vous permet de modifier l'entrée sélectionnée (si vous voulez apporter des modifications à votre news par exemple).
3. Supprimer l'entrée : ce bouton supprime l'entrée sélectionnée.
4. Afficher X lignes à partir de l'enregistrement n° X : s'il y a beaucoup d'entrées dans votre table, PhpMyAdmin n'en affichera qu'un bout (les 30 premières lignes normalement). Si vous voulez en afficher plus, il vous suffit de modifier ces valeurs puis de cliquer sur "Afficher".

Voilà, vous en savez suffisamment pour travailler sur une table. Avouez que ce n'était pas bien dur. l y a certes beaucoup de choses que je passe sous silence, mais c'est principalement parce que vous n'en aurez besoin que très rarement.

Bon, il nous reste à traiter encore de quelques fonctionnalités proposées par PhpMyAdmin, et ça sera bon pour ce chapitre.

Autres Opérations

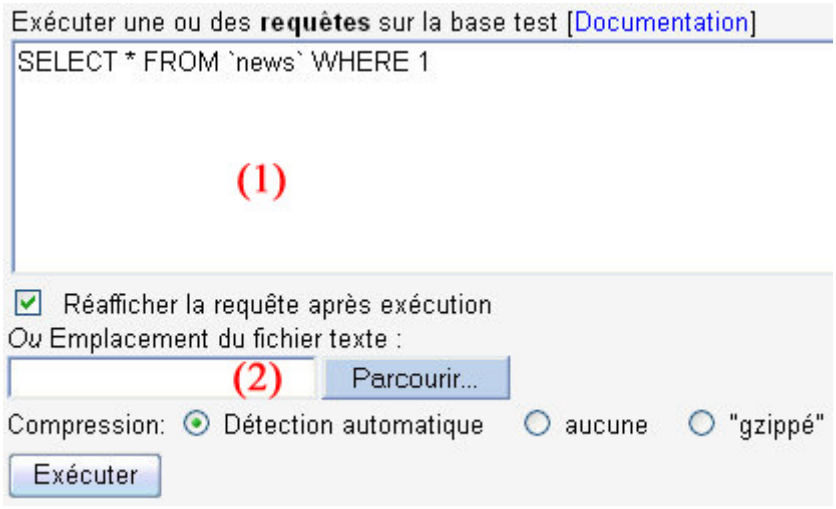
Nous allons séparer cette partie en 5 sous-parties, correspondant aux onglets suivants :

- SQL
- Exporter
- Opérations
- Vider
- Supprimer

SQL

Cliquez sur l'onglet : 

Il s'affiche à l'écran :



C'est ici que vous pouvez exécuter ce que l'on appelle des requêtes SQL pour demander à Mysql de faire quelque chose.
Vous avez 2 méthodes pour exécuter une requête SQL :

1. Dans la grande zone de texte, vous pouvez taper des requêtes SQL. Par exemple ici on a :
   *SELECT * FROM `news` WHERE 1*
   Cela signifie : "Afficher tout le contenu de la table 'news'"
   Je vous apprendrai ce langage SQL tout au long de la partie II.
2. Dessous, vous pouvez cliquer sur le bouton Parcourir pour rechercher un fichier sur votre disque dur qui contient des requêtes SQL. Ca revient exactement au même, mais il est parfois plus facile de s'échanger des requêtes SQL à l'aide d'un fichier texte.

Pour valider, cliquez sur "Exécuter".

Exporter

Il nous reste à voir les 4 onglets à droite :

Nous nous intéressons maintenant à l'onglet "Exporter". C'est ici que vous allez pouvoir récupérer votre base de données sur le disque dur sous forme de fichier texte (qui contiendra des tonnes de requêtes SQL).

Ce fichier que l'on va "exporter", est-ce que c'est le même que celui dont tu nous parlais tout à l'heure ? Celui situé dans C:\Program Files\EasyPHP\mysql\data ?

Non pas du tout. Ce que je vous ai montré tout à l'heure, c'était quelque chose d'illisible. Je vous avais dit qu'on n'y toucherai pas, je ne vous ai pas menti.

Le fichier que vous allez obtenir grâce à "l'exportation" de PhpMyAdmin, c'est un fichier qui dit à MySQL *comment recréer votre base de données* (avec des requêtes en langage SQL)

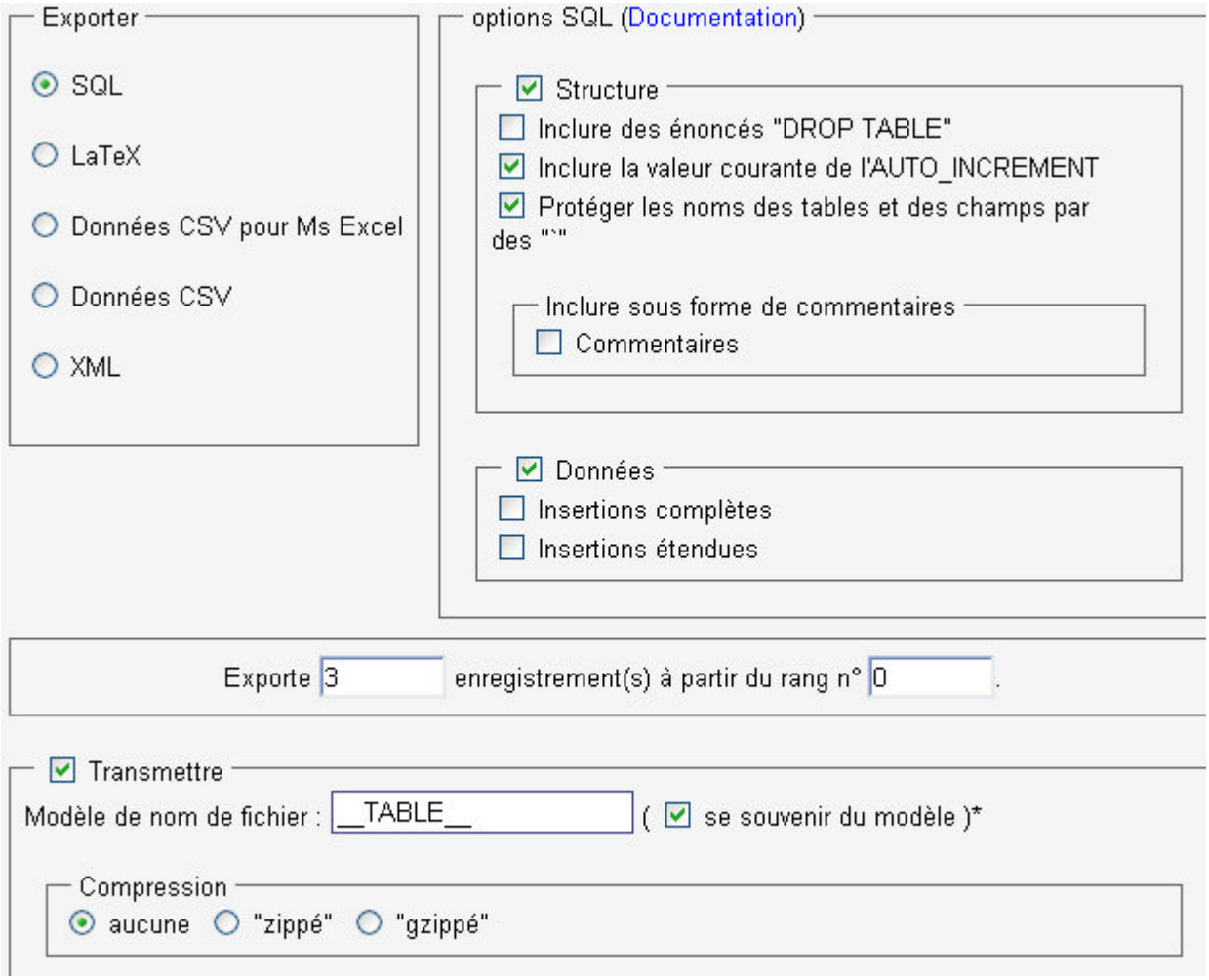
A quoi il sert ce fichier ?

On peut s'en servir pour deux choses :

- **Transmettre votre base de données sur Internet** : pour le moment, votre base de données se trouve sur votre disque dur. Mais si vous êtes hébergés sur Internet, chez Free par exemple, on va utiliser ce fichier généré pour "reconstruire" la base de données. Ainsi, sur Internet vous aurez la même base de données et votre site web pourra l'utiliser !
- **Faire une copie de sauvegarde de la base de données** : on ne sait jamais, si vous faites une bêtise ou qu'un hacker détruit 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 !

Attention, je vous rappelle un point important : le fichier que vous allez générer contient les informations pour "reconstruire" votre base de données. Ce n'est donc pas le fichier dans lequel MySQL enregistre vos données, dont je vous ai parlé à la fin du chapitre précédent.

Votre écran doit ressembler à ceci :



Je vous conseille de laisser les options par défaut, c'est largement suffisant.
Distinguez simplement la structure des données de la table. La structure d'une table se résume en

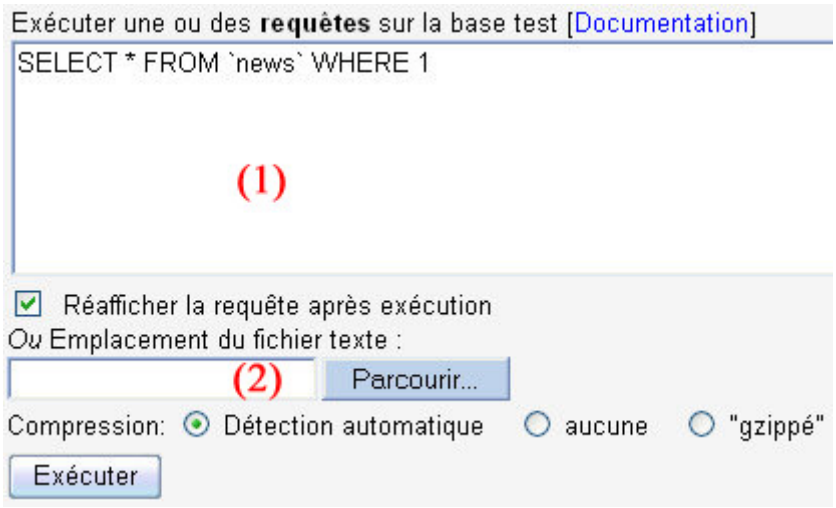
quelques lignes, ce sont en fait les noms des champs, leurs types etc... Par contre, les données correspondent aux entrées, et il peut y en avoir beaucoup ! Pour faire une sauvegarde complète, il faut donc prendre la structure ET les données.

A noter que vous pouvez demander une compression, ce qui est utile si votre table est très grosse.

Cliquez sur "Exécuter". On vous proposera alors de télécharger un fichier : c'est tout à fait normal. N'hésitez pas à regarder ce qu'il y a dans ce fichier : vous allez voir qu'il y a plein de requêtes SQL. C'est ce langage que je vais vous apprendre dans les chapitres qui suivent !

Bon, j'ai récupéré le fichier. Maintenant, comment je fais pour recréer la base de données sur mon site web ?

Il faut aller sur le PhpMyAdmin de votre hébergeur (il en a forcément un). Renseignez-vous pour connaître l'adresse.
Par exemple chez Free c'est : http://phpmyadmin.free.fr/phpMyAdmin (il faudra indiquer votre login et mot de passe). Une fois dessus, rendez-vous dans l'onglet "SQL", vous devriez voir ceci :



Oui je sais, on a déjà vu cette image toute à l'heure. Nous, on a besoin juste de la partie (2) : "Emplacement du fichier texte". Cliquez sur "Parcourir" pour indiquer où se trouve le fichier sur votre disque dur. Faites "Exécuter", attendez que ça l'envoie, et c'est bon ! Votre base de données est alors recréée sur Internet !

Opérations

Vous pouvez faire ici diverses opérations sur votre table.
Je ne vais pas les énumérer une à une, ni vous expliquer comment elles fonctionnent vu que c'est très simple. Sachez simplement que vous pourriez avoir besoin de :

- Changer le nom de la table : indiquez le nouveau nom pour cette table.
- Déplacer la table vers : si vous voulez mettre cette table dans une autre base de données.
- Copier la table : faire une copie de la table, dans une autre base ou dans la même (attention, dans ce cas il faudra qu'elle ait un nom différent).
- Réparer la table : ne me demandez pas comment ça fonctionne, tout ce que je sais c'est que si votre table semble poser problème, la réparation peut tout régler (je m'en suis servi une ou deux fois).

- Optimiser la table : à force d'utiliser une table, surtout si elle est grosse, on finit par avoir des "pertes" qui font que la table n'est plus bien organisée. Un clic là-dessus et hop, c'est ré-arrangé.

Vider

Vide tout le contenu de la table. Toutes les entrées vont disparaître, seule la structure de la table restera (c'est-à-dire les champs).

Attention ! Il n'est pas possible d'annuler cette opération !

Supprimer

Pour supprimer la totalité de la table (structure + données), cliquez sur cet onglet.
Là encore, réfléchissez-y à deux fois avant de tout supprimer, car vous ne pourrez rien récupérer par la suite.

Nous avons vu la plupart des fonctionnalités utiles de PhpMyAdmin.
C'est que PhpMyAdmin permet de faire beaucoup de choses, vous venez de le voir !

C'est pour vous un "outil" qui vous permettra d'administrer votre base de données, de voir ce qu'elle contient et dans quel état elle est.

Mais maintenant nous allons rentrer dans le vif du sujet : comment utiliser une base de données avec PHP ?
Les choses sérieuses vont commencer, et vous allez vite être capables de créer plein de scripts très utiles pour votre site !

Lire Des Données

Fini de faire joujou, on retourne à nos pages PHP !

Dans ce chapitre, nous allons nous entraîner à lire des données dans une base de données. C'est un chapitre très important, un peu gros certes mais c'est parce que vous avez beaucoup à apprendre. Je pense sincèrement que ce sera un chapitre très enrichissant pour vous, alors lisez-le avec soin !

Connexion À La BDD

Pour pouvoir travailler avec la base de données, il faut d'abord s'y connecter.

Se connecter à la base de données...
Hein !? Quoi ?! Il est où le modem ?!

Ne rigolez pas, c'est exactement ce que je me posais comme question quand on me disait "il faut te connecter à la base de données".
Alors vous vous aurez de la chance, vous ne resterez pas dans le flou comme moi.

Voici ce qu'il faut savoir :

Nous allons apprendre dans ce chapitre à lire des données dans une BDD. Or, je vous rappelle que PHP doit faire l'intermédiaire entre vous et MySQL.
Problème : PHP ne peut pas dire à MySQL dès le début "Récupère-moi ces valeurs". En effet, MySQL demande d'abord un nom d'utilisateur et un mot de passe. S'il ne faisait pas ça, tout le monde pourrait accéder à votre BDD et lire les informations qu'il y a dedans (parfois confidentielles !).

Il va donc falloir que PHP s'authentifie, on dit qu'il établit une connexion avec MySQL. Une fois que la connexion sera établie, vous pourrez faire n'importe quelle opération sur votre base de données.

On va pour commencer apprendre 3 étapes :

- La connexion
- Le choix de la base
- La déconnexion

La Connexion

Pour vous connecter, vous utiliserez une fonction PHP : *mysql_connect*.
Cette fonction a besoin de 3 arguments qu'il vous faudra renseigner :

- Le nom de l'hôte : c'est l'IP de l'ordinateur où MySQL est installé. Le plus souvent, MySQL est installé sur le même ordinateur que PHP. Dans ce cas, mettez la valeur "localhost" et ça marchera.
- Le login : ça permet de vous identifier. Renseignez-vous auprès de votre hébergeur pour le connaître. Le plus souvent (chez un hébergeur gratuit) c'est le même login que vous utilisez pour le FTP.
- Le mot de passe : là encore, il y a 99% de chances pour que le mot de passe soit le même que celui que vous utilisez pour accéder au FTP (ça ne vous coûte rien d'essayer).

On va supposer que le nom de l'hôte est "localhost" (c'est valable dans la quasi-totalité des cas), que mon login est "mateo21" et que mon mot de passe est "cFrrI954$sH".
Le code suivant permet d'établir une connexion à MySQL :

Source 2.3.1 : connexion à la BDD

```
<?
mysql_connect("localhost", "mateo21", "cFrrI954$sH");
?>
```

Si vous faites ça, c'est bon vous êtes connectés !
Il vous faudra peut-être rechercher un peu votre login et votre mot de passe (demandez à votre hébergeur), mais il y a des chances que ce soient les mêmes que pour votre FTP.

Pour vous connecter à MySQL avec EasyPHP (si vous faites des tests sur votre propre ordinateur), vous devez mettre l'hôte "localhost", le login "root", et pas de mot de passe.
C'est-à-dire : mysql_connect("localhost", "root", "");

Le Choix De La Base

OK, on est connecté, mais il faut maintenant sélectionner la base de données sur laquelle vous allez travailler.
Bien souvent, une seule base de données suffit, je vous le rappelle. D'ailleurs, la plupart des hébergeurs gratuits n'en propose qu'une seule, ce qui n'est pas bien grave.
Demandez à votre hébergeur le nom de la base qui a été créée (souvent c'est le même nom que votre login MySQL).

La fonction qui permet de sélectionner la BDD est : *mysql_select_db*
En temps normal, vous n'aurez besoin d'indiquer qu'un paramètre : le nom de la base.

Par exemple, si ma base s'appelle "mateo21", voici comment je dois procéder :

Source 2.3.2 : sélection de la BDD

```
<?
mysql_connect("localhost", "mateo21", "cFrrI954$sH"); // Connexion à MySQL
mysql_select_db("mateo21"); // Sélection de la base mateo21
?>
```

La Déconnexion

Enfin, dernière chose (après ça c'est bon, promis).

Une fois que vous vous êtes connectés, que vous avez choisi votre base de données, vous pouvez commencer votre travail. Mais une fois que vous avez fini de travailler sur votre BDD, il faut vous déconnecter !

Alors, pour se déconnecter, c'est tout bête : *mysql_close*
Et y'a même pas besoin de paramètre !

En résumé, voici comment on fait pour se connecter et se déconnecter de MySQL :

Source 2.3.3 : connexion et déconnexion

```
<?
mysql_connect("localhost", "mateo21", "cFrrI954$sH"); // Connexion à MySQL
mysql_select_db("mateo21"); // Sélection de la base mateo21

// On est connectés, on peut travailler sur la BDD
// ...
// ...

// On a fini de travailler, on ferme la connexion :
mysql_close(); // Déconnexion de MySQL
?>
```

C'est comme ça qu'on procède quand on utilise une BDD !

Le code source 2.3.3 n'affiche rien à l'écran si tout se passe bien.
Si la connexion a échoué, vous aurez un message d'erreur. Dans ce cas c'est que votre login, mot de passe ou nom d'hôte n'est pas bon. Demandez plus d'infos à votre hébergeur.

Récupérer Les Données

Normalement, quand on crée un site, on doit d'abord mettre des données, puis après on les lit.
Mais moi, pour que vous appreniez en douceur, je vais d'abord vous apprendre à lire des données, et après je vous apprendrai à écrire des données dans la BDD.

Mais... il nous faudrait une base de données "toute prête" qui servirait de support pour travailler.
Heureusement, c'est mon jour de bonté, je vais vous épargner tout ça. Je vous invite à télécharger la table que j'ai créée pour vous :

Télécharger la table (2,5 Ko)

Rien qu'au nom, vous pouvez vous douter que cette table contient quelque chose en rapport avec des jeux vidéos. En effet, vous allez le voir, cette table contient une liste d'une cinquantaine de jeux vidéos.

Pour cet exemple, plusieurs amis ont voulu répertorier tous les jeux vidéos qu'ils possèdent. La base de données est pour eux un moyen très pratique de classer et d'organiser tout cela, vous allez voir pourquoi.

Euh dis, j'en fais quoi moi de ton fichier jeux_videos.sql.gz ?

Inutile d'essayer de l'ouvrir, ça n'a pas d'intérêt. Il va falloir **importer la table** dans PHPMyAdmin (c'est le fichier que je vous ai donné).
Voici la procédure à suivre :

1. Rendez-vous dans PhpMyAdmin
2. Sélectionnez la base "test" dans le menu déroulant en haut à gauche
3. Cliquez ensuite sur l'onglet "SQL".
4. En bas, vous avez un bouton "Parcourir" : cliquez dessus.
5. Dans la boîte de dialogue qui s'ouvre indiquez où se trouve le fichier jeux_videos.sql.gz que je vous ai fait télécharger.
6. Ne touchez pas au reste et cliquez sur "Exécuter".

Petit aperçu :

Et voilà ! Vous devriez voir une nouvelle table apparaître à gauche : "jeux_videos". Vous pouvez vous amuser à regarder ce qu'elle contient, pour vous faire une idée.
Voici les 5 premières entrées qu'elle contient (il y en a une cinquantaine en tout !) :

ID	nom	possesseur	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	Florent	NES	4	1	Un jeu d'anthologie !
2	Sonic	Patrick	Megadrive	2	1	Pour moi, le meilleur jeu au monde !
3	Zelda : ocarina of time	Florent	Nintendo 64	15	1	Un jeu grand, beau et complet comme on en voit rarement de nos jours
4	Mario Kart 64	Florent	Nintendo 64	25	4	Un excellent jeu de kart !
5	Super Smash Bros Melee	Michel	GameCube	55	4	Un jeu de baston délirant !

Pour le moment ne modifiez pas cette table.

 Bon, et maintenant qu'est-ce qu'on va en faire ?

Notre objectif, c'est de *créer une page PHP qui va afficher ce que contient la table "jeux_videos"*.

Faire Une Requête

Maintenant arrive le grand moment que vous attendiez tous : on va demander quelque chose à MySQL. On va donc commencer à parler en "SQL" !
Pour cela, on va faire ce qu'on appelle une **requête**. On va demander poliement à MySQL de nous dire tout ce que contient la table "jeux_videos".

Nous allons nous servir de la fonction PHP : *mysql_query*

 "query" en anglais veut dire "requête"

- Cette fonction prend un paramètre : ce que PHP doit aller dire à MySQL (en langage SQL).
- Cette fonction renvoie une valeur, il faut donc qu'une variable récupère ce que MySQL nous a répondu.

On fera tout le temps comme ça :

Source 2.3.4 : faire une requête

```
<?
$reponse = mysql_query("Tapez votre requête SQL ici");
?>
```

$reponse contiendra la réponse de MySQL.
Nous allons voir comment demander à MySQL tout ce qu'il y a dans la table "jeux_videos".

Votre Première Requête SQL

Comme je vous l'ai dit, le SQL est un langage. C'est lui qui nous permet de communiquer avec MySQL.
Voici votre première requête SQL :

*SELECT * FROM jeux_videos*

Ceci peut se traduire par : "Prendre tout ce qu'il y a dans la table "jeux_videos".
Analysons chaque terme de cette requête :

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

Et voilà le travail !
Maintenant, on n'a plus qu'à mettre cette requête en paramètre de *mysql_query* :

Source 2.3.5 : récupérer toute la table jeux_videos

```
<?
$reponse = mysql_query("SELECT * FROM jeux_videos");
?>
```

Notre variable $reponse contient maintenant la réponse de MySQL !

Euh ouais, cool, et comment on affiche le résultat ?

Afficher Le Résultat D'Une Requête

Le problème, c'est que $reponse contient quelque chose d'inexploitable. MySQL nous renvoie un joyeux bazar pas bien organisé.

Vous imaginez toutes les informations qui sont dedans ? Si c'est une table à 10 champs, avec 200 entrées, ça fait plus de 2000 informations dans une variable !
Dur dur de tout caser... sauf... si on utilisait un array !

Bingo !
PHP dispose d'une fonction toute prête, *mysql_fetch_array*, qui va créer un array à partir de $reponse.

Ce sera un tableau associatif : vous mettrez entre crochets le nom du champ qui vous intéresse.
Par exemple, si vous vous intéressez au champ "console", vous utiliserez l'array $donnees['console'].

Il faudra faire une boucle pour parcourir chaque entrée une à une. A chaque fois que vous utilisez la fonction *mysql_fetch_array*, vous passez à l'entrée suivante. La boucle est donc répétée autant de fois qu'il n'y a d'entrées dans votre table.

Voici donc comment je fais pour afficher le résultat de la requête :

Source 2.3.6 : afficher le résultat de la requête

```
<?
mysql_connect("localhost", "mateo21", "mot de passe"); // Connexion à MySQL
mysql_select_db("coursphp"); // Sélection de la base coursphp

$reponse = mysql_query("SELECT * FROM jeux_videos"); // Requête SQL


// On fait une boucle pour lister tout ce que contient la table :

while ($donnees = mysql_fetch_array($reponse) )
{
?>

<b>Jeu</b> : <? echo $donnees['nom']; ?><br>
Le possesseur de ce jeu est : <? echo $donnees['possesseur']; ?>, et il le vend à <? echo
$donnees['prix']; ?> euros !<br>
Ce jeu fonctionne sur <? echo $donnees['console']; ?> et on peut y jouer à <? echo
$donnees['nbre_joueurs_max']; ?> au maximum<br>
<? echo $donnees['possesseur']; ?> a laissé ces commentaires sur <? echo $donnees['nom']; ?> :
<i><? echo $donnees['commentaires']; ?></i><p>

<?
}

mysql_close(); // Déconnexion de MySQL
?>
```

Alors, vous avez vu ?
Ca en fait un paquet de texte ! Il faut dire que la table que je vous ai donné contient une cinquantaine d'entrées, donc c'est normal que vous ayez beaucoup de résultats !

Et ceci mis à part, qu'en pensez-vous ? C'est puissant non ?!
Amusez-vous à changer mon script, faites des tests, c'est super important (bien entendu n'oubliez pas d'adapter le login et le mot de passe).

Concrètement que se passe-t-il ? On fait une boucle pour chaque entrée de la table. On commence par l'entrée n°1, puis l'entrée n°2 etc... A chaque fois qu'on fait une nouvelle boucle, on passe en revue un autre entrée.

Quelle est la différence entre $reponse et $donnees ?

$reponse contenait la réponse de MySQL en vrac.
$donnees est un array renvoyé par la fonction *mysql_fetch_array*. A chaque fois qu'on fait une boucle, *mysql_fetch_array* va chercher dans $reponse l'entrée suivante et organise les champs dans $donnees.

 "Fetch" en anglais signifie "va chercher".

Avec ce que je vous ai appris, vous devriez être capable d'afficher ce que vous voulez.
Personne ne vous oblige à afficher tous les champs ! Par exemple, si j'avais voulu lister juste les noms des jeux, j'aurais fait comme ça :

Source 2.3.7 : afficher uniquement le nom du jeu

```
<?
mysql_connect("localhost", "mateo21", "mot de passe"); // Connexion à MySQL
mysql_select_db("coursphp"); // Sélection de la base coursphp

$reponse = mysql_query("SELECT nom FROM jeux_videos"); // Requête SQL


// Avec cette boucle, on liste uniquement le nom des jeux :

while ($donnees = mysql_fetch_array($reponse) )
{
echo $donnees['nom'];
echo "<br>";
}

mysql_close(); // Déconnexion de MySQL
?>
```

Je sais pas vous, mais moi je trouve que là-dedans il y a quelque chose de merveilleux : ce code source 2.3.7 est inintelligible pour Mr-tout-le-monde, et pourtant il permet d'afficher d'un coup d'un seul la liste d'une cinquantaine de jeux vidéos.
Et croyez-moi, vous n'êtes pas au bout de vos suprises avec PHP et MySQL !

Maintenant vous faites ce-que-vous-voulez.
Vous affichez ça de la manière que vous voulez, dans un tableau (<table>), avec des sauts à la ligne, en gras, italique, souligné...
Bref, toute la puissance de MySQL combinée à PHP est là !

Vous croyez avoir tout vu ?
Laissez-moi vous prouver le contraire !

Les Critères De Sélection

Ici, nous allons nous occuper uniquement des requêtes SQL.
Vous allez voir qu'en les modifiant, vous pouvez trier et ordonner différemment vos données très facilement.

Nous allons nous intéresser aux éléments suivants :

- WHERE
- ORDER BY
- LIMIT

WHERE

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

Supposons par exemple que je veuille lister uniquement les jeux appartenant à Patrick. La requête au début sera la même qu'avant, mais je rajouterai à la fin "WHERE possesseur='Patrick'".
Ca nous donne la requête :

*SELECT * FROM jeux_videos WHERE possesseur='Patrick'*

Traduction : "*Sélectionner tous les champs de la table jeux_videos lorsque le champ possesseur est égal à Patrick.*

Un petit code pour voir ce que ça donne ?

Source 2.3.8 : afficher uniquement les jeux de Patrick

```
<?
mysql_connect("localhost", "mateo21", "mot_de_passe");
mysql_select_db("coursphp");

// Sélectionnons les champs nom et possesseur de la table "jeux_videos", uniquement lorsque le jeu appartient à Patrick
$reponse = mysql_query("SELECT nom, possesseur FROM jeux_videos WHERE possesseur='Patrick'");

while ($donnees = mysql_fetch_array($reponse) )
{
?>

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

<?
}

mysql_close();
?>
```

Si vous vous amusez à changer le nom du possesseur (par exemple "WHERE possesseur='Michel'), ça n'affichera que les jeux appartenant à Michel ! Essayez, vous verrez !

Il est par ailleurs possible de mettre deux conditions. Par exemple, si je veux lister les jeux de Patrick qu'il vend à moins de 20 euros, j'utiliserai cette requête SQL :

*SELECT * FROM jeux_videos WHERE possesseur='Patrick' AND prix<20*

Traduction : "*Sélectionner tous les champs de jeux_videos lorsque le possesseur est Patrick ET lorsque le prix est inférieur à 20*".

ORDER BY

ORDER BY nous permet d'ordonner nos résultats (histoire qu'ils ne soient pas trop en vrac...).
Nous pourrions classer les résultats en fonction de leur prix ! La requête SQL serait :

*SELECT * FROM jeux_videos ORDER BY prix*

Traduction : "*Sélectionner tous les champs de jeux_videos, et ordonner les résultats par prix croissant.*".

Application :

Source 2.3.9 : ordonner les jeux par prix croissant

```
<?
mysql_connect("localhost", "mateo21", "mot_de_passe");
mysql_select_db("coursphp");

// Sélectionner les champs "nom" et "prix" de jeux videos et ordonner les résultats par prix.
$reponse = mysql_query("SELECT nom, prix FROM jeux_videos ORDER BY prix");

while ($donnees = mysql_fetch_array($reponse) )
{
?>

<? echo $donnees['nom']; ?> coûte <? echo $donnees['prix']; ?> &euro;<br>

<?
}

mysql_close(); // Déconnexion de MySQL
?>
```

 Et si je veux classer par ordre décroissant ?

Facile : il suffit de rajouter le mot-clé DESC à la fin :

*SELECT * FROM jeux_videos ORDER BY prix DESC*

Traduction : "*Sélectionner tous les champs de jeux_videos, et ordonner les résultats par prix décroissant.*".

 A noter : si on avait utilisé ORDER BY sur un champ contenant du texte, le classement aurait été fait par ordre alphabétique.

LIMIT

Dernier mot-clé que nous apprendrons dans ce chapitre, LIMIT nous permet de ne prendre qu'une partie des résultats (par exemple les 20 premiers).

Il faut rajouter à la fin de la requête le mot clé LIMIT, suivi de 2 nombres séparés par une virgule. Par exemple :

*SELECT * FROM jeux_videos LIMIT 0, 20*

Mais ils veulent dire quoi ces deux nombres ?

Bonne question !

- On indique tout d'abord à partir de quelle entrée on commence à lire la table.
  Ici, j'ai mis 0. Pour MySQL c'est la première entrée (1 c'est la seconde, 2 la troisième etc...).

  Attention, n'oubliez jamais que pour MySQL la première entrée est l'entrée n°0 !
  Par ailleurs, sachez que LIMIT ne se base PAS sur le champ ID (ça fonctionne même s'il n'y a pas de champ ID).

- Ensuite, le deuxième nombre indique combien d'entrées on doit sélectionner. Ici, j'ai mis 20, on prendra donc 20 entrées.

Donc, si on met :

LIMIT 0,20 : ça affiche les 20 premières entrées.
LIMIT 5,10 : ça affiche les entrées n°6 à 15.
LIMIT 10,2 : ça affiche les entrées n°11 et 12.

Compris ?

Allez un petit exemple ! Si on veut afficher les 10 premiers jeux de la table, on utilisera le code suivant :

Source 2.3.10 : afficher les 10 premières entrées

```
<?
mysql_connect("localhost", "mateo21", "mot de passe");
mysql_select_db("coursphp");

// Sélectionner les 10 premières entrées de la table jeux_videos
$reponse = mysql_query("SELECT nom FROM jeux_videos LIMIT 0, 10");

echo "Voici les 10 premières entrées de la table jeux_videos :<p>";

while ($donnees = mysql_fetch_array($reponse) )
{
?>

<? echo $donnees['nom']; ?><br>

<?
}

mysql_close(); // Déconnexion de MySQL
?>
```

Et voilà le travail !

Bonjour, je suis masochiste, et avant de terminer cette section je souhaiterais mélanger toutes les requêtes SQL que je viens d'apprendre en une seule. C'est possible ?

Mais bien entendu mon petit !!!!

Voilà de quoi te triturer les méninges :

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

Il faut utiliser les mots-clés dans l'ordre que j'ai donné : WHERE puis ORDER BY puis LIMIT, sinon MySQL ne comprendra pas votre requête.

Essayez donc de traduire ça en français déjà, pour voir si vous avez compris, puis après testez cette requête chez vous pour voir si c'est bien ce à quoi vous vous attendiez.

Pfiouuu ! Eh bah, si avec ça vous devenez pas des pros du SQL….

Compter Le Nombre D'Entrées

Avant de terminer ce chapitre, on va apprendre à faire quelque chose qui nous sera parfois très utile : demander à mysql le nombre d'entrées dans une table. Cela vous permettra de dire par exemple : *Il y a 23 jeux vidéos en vente actuellement !*

Pour ce faire, on va utiliser la requête suivante :

Source 2.3.11 : compter le nombre d'entrées

```
<?
mysql_connect("localhost", "mateo21", "mot_de_passe");
mysql_select_db("coursphp");

// Combien d'entrées dans jeux vidéos ?
$retour = mysql_query("SELECT COUNT(*) AS nbre_entrees FROM jeux_videos");
$donnees = mysql_fetch_array($retour);


?>

Il y a <? echo $donnees['nbre_entrees']; ?> jeux vidéos en vente !

<?
mysql_close(); // Déconnexion de MySQL
?>
```

Comme vous pouvez le voir, la requête est un peu différente. Le mot-clé COUNT demande à MySQL de compter le nombre d'entrées, et de renvoyer le résultat dans l'array $donnees['nbre_entrees'].
On ne fait pas de boucle, il n'y en a pas besoin. MySQL a juste renvoyé le nombre de jeux vidéos inscrits dans la table.

Et n'oubliez pas que vous pouvez rajouter à la fin de la requête un WHERE, par exemple pour avoir juste le nombre de jeux vidéo appartenant à Florent !
A vous de jouer !

Vous êtes arrivés vivants jusqu'au bout ? Bravo !

Vous venez d'apprendre une quantité de choses impressionnantes dans ce chapitre ! Une fois que vous aurez lu le chapitre suivant, vous serez même capables de créer des scripts de news, de livre d'or, un forum etc etc...
Vu que ce chapitre était d'une importance capitale, n'hésitez pas à le relire (après vous être reposés), car il faut vraiment que vous maîtrisiez les requêtes SQL et leur affichage avec PHP !

Écrire Des Données

Nous avons vu dans le chapitre précédent que MySQL pouvait récupérer des données dans la BDD très facilement. Nous avons vu aussi que le langage SQL était très puissant, car il propose propose de nombreux critères de sélection (WHERE, ORDER BY etc...)

C'est bien beau tout ça, mais si vous savez juste lire dans une base de données et que vous ne savez pas écrire dedans, ça va pas le faire.

Vous l'aurez compris, ce chapitre est clairement la suite du précédent. En utilisant ce que vous aurez appris dans ces 2 chapitres, vous saurez réaliser de nombreux scripts PHP.

Ajouter Des Données

Votre mission, si vous l'acceptez : ajouter une nouvelle entrée à la table "jeux_videos" (sur laquelle nous avons travaillé dans le chapitre précédent).

Mouahahahah, mais c'est facile. Tu utilises PhpMyAdmin et hop ! C'est fait !
..... Quoi, j'ai dit quelque chose de mal ?

Non non ! C'est vrai que PhpMyAdmin permet de rajouter de nouvelles entrées dans la table (on l'a vu dans le chapitre 2 de la partie II). Mais ce qui nous intéresse ici, c'est de le faire avec un script PHP !

Tout d'abord, je vous rappelle à quoi ressemble la table "jeux_videos" :

ID	nom	possesseur	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	Florent	NES	4	1	Un jeu d'anthologie !
2	Sonic	Patrick	Megadrive	2	1	Pour moi, le meilleur jeu au monde !
3	Zelda : ocarina of time	Florent	Nintendo 64	15	1	Un jeu grand, beau et complet comme on en voit rarement de nos jours
4	Mario Kart 64	Florent	Nintendo 64	25	4	Un excellent jeu de kart !
5	Super Smash Bros Melee	Michel	GameCube	55	4	Un jeu de baston délirant !
...	...	...	...	...	...	...

Pour rajouter une entrée, vous aurez besoin de connaître la requête SQL. En voici une par exemple qui rajoute un champ :

INSERT INTO jeux_videos(ID, nom, possesseur, console, prix, nbre_joueurs_max, commentaires)
VALUES('', 'Battlefield 1942', 'Patrick', 'PC', '45', '50', '2nde guerre mondiale')

- D'abord, vous devez mettre INSERT INTO pour dire que vous allez insérer une entrée.
- Vous précisez ensuite le nom de la table (ici "jeux_videos"), puis mettez entre parenthèses les noms des champs.
- Enfin, et c'est là qu'il ne faut pas se tromper, vous devez écrire VALUES et mettre les valeurs à insérer **dans le même ordre que les champs que vous avez indiqués**.

Vous remarquerez que pour le premier champ (ID), je n'ai rien mis entre les apostrophes. C'est voulu : le champ a la propriété "auto_increment", MySQL mettra donc le numéro d'ID lui-même.

Enfin, si vous le désirez, sachez que vous n'êtes pas obligés de mettre les noms des champs d'abord, cette requête marche tout aussi bien (mais elle est moins claire) :

INSERT INTO jeux_videos VALUES('', 'Battlefield 1942', 'Patrick', 'PC', '45', '50', '2nde guerre mondiale')

Du temps que vous respectez le bon ordre des champs, tout ira bien !

Maintenant, voici le script PHP qui utilise cette requête :

Source 2.4.1 : ajouter une entrée

```
<?
mysql_connect("localhost", "mateo21", "mot de passe");
mysql_select_db("coursphp");

// On ajoute une entrée avec mysql_query
mysql_query("INSERT INTO jeux_videos VALUES('', 'Battlefield 1942', 'Patrick', 'PC', '45',
'50', '2nde guerre mondiale')");

mysql_close();
?>
```

Que fait ce code ? Il ajoute une entrée dans la BDD pour le jeu "Battlefield 1942", appartenant à "Patrick", qui fonctionne sur "PC", qui coûte "45" euros etc...
Entendons-nous bien : **ce code n'affiche rien**. Il ajoute juste des données dans la BDD. Ce n'est que si vous faites un SELECT (comme nous l'avons vu dans le précédent chapitre) que nous aurons quelque chose d'intéressant à afficher au visiteur.

Vous verrez dans la pratique qu'on combine les deux : on écrit et on lit dans la BDD.

Modifier Des Données

Vous venez de rajouter Battlefield dans la BDD, tout s'est bien passé.
Mais... vous vous rendez compte avec stupeur que Battlefield se joue en fait à 32 joueurs maximum (au lieu de 50), et que en plus son prix a baissé : on le trouve à 10 euros (au lieu de 45).

No problemo amigo ! Avec une petite requête SQL on peut arranger ça. En effet, en utilisant UPDATE vous allez pouvoir modifier l'entrée qui pose problème :

UPDATE jeux_videos SET prix='10', nbre_joueurs_max='32' WHERE ID='51'

Comment ça marche ?

- Tout d'abord, le mot-clé UPDATE permet de dire qu'on va modifier une entrée.
- Ensuite, le nom de la table (jeux_videos).
- Le mot-clé SET, qui sépare le nom de la table du reste.
- Et on met ensuite les champs qu'il faut modifier, séparés par des virgules. Ici, on modifie le champ "prix", on lui affecte la valeur "10" (prix='10'), et de même pour le champ nbre_joueurs_max.
  Les autres champs ne sont pas modifiés.
- Enfin, le mot-clé WHERE est tout simplement indispensable. Ca nous permet de dire à MySQL quelle entrée il doit modifier. On se base très souvent sur le champ ID pour indiquer quelle entrée est à modifier. Ici, on suppose que Battlefield a été enregistré sous l'ID n°51.


Pour connaître l'ID de Battlefield, il faudrait aller sous PhpMyAdmin et regarder quel n° d'ID MySQL lui a donné.

- Et si vous voulez, vous pouvez vous baser sur le nom du jeu au lieu de l'ID (pour le WHERE) : *UPDATE jeux_videos SET prix='10', nbre_joueurs_max='32' WHERE nom='Battlefield 1942'*

Dernière minute ! Florent vient de racheter tous les jeux de Michel ! Il va falloir modifier ça tout de suite !


Heu, va falloir modifier chaque entrée une à une ?

Dites-vous bien une chose : le langage SQL est un langage de feignasse. Il n'est pas question de passer des heures à modifier toute la table pour ça !
En clair, en réfléchissant environ 0,5 seconde vous allez trouver tous seuls la requête SQL qui permet de faire ce qu'on cherche.

C'est bon vous avez trouvé ? Allez, je vous donne la réponse, c'est vraiment facile :

UPDATE jeux_videos SET possesseur='Florent' WHERE possesseur='Michel'

Traduction : *Dans la table jeux_videos, modifier toutes les entrées dont le champ possesseur est égal à Michel, et le remplacer par Florent.*
Qu'il y ait 1, 10, 100 ou 1000 entrées, cette requête à elle-seule suffit pour mettre à jour toute la table !

Si c'est pas beau le SQL !

Supprimer Des Données

Enfin, voilà une dernière requête qui pourra se révéler utile : DELETE.
Rapide et simple à utiliser, elle est quand même un poil dangereuse : après suppression, il n'y a aucun moyen de récupérer les données, alors faites attention !

Voici comment on supprime par exemple l'entrée de Battlefield :

DELETE FROM jeux_videos WHERE nom='Battlefield 1942'

Y'a rien de plus facile :

- DELETE FROM : pour dire "supprimer dans"
- jeux_videos : le nom de la table
- WHERE : indispensable pour indiquer quelle(s) entrée(s) doivent être supprimée(s). Si vous l'oubliez, tout sera supprimé ! Cela équivaut à vider la table.

Et voilou vous savez tout !

La partie II de ce cours est terminée ! Nous avons vu ce qu'il fallait savoir pour MySQL. Dans la partie III vous allez apprendre des choses pas bien difficiles et pourtant très utiles. C'est maintenant que vous allez en apprendre le plus sur PHP !

Les Includes

Tout d'abord, bienvenue dans la partie III. Comme son nom l'indique, la partie III sera sans aucun doute la plus riche de toutes. Par ailleurs, elle ne comporte aucune difficulté particulière, ce qui fait que si vous avez bien suivi jusque là, vous allez pouvoir apprécier pleinement tout ce que j'ai encore à vous apprendre sur PHP.

On commence la partie III par un chapitre Ô combien important en PHP : nous allons parler d'includes. Derrière ce nom barbare, vous allez le voir, se cache une des fonctions PHP les plus utilisées.
Je vous le garantis, après lecture de ce chapitre votre site va vraiment changer de visage !

La Fonction Include

Nous n'allons parler que d'une seule fonction : *include*. Elle est très simple d'emploi et fréquemment utilisée car très puissante.

 Que fait cette fonction ?

Elle permet d'inclure le contenu d'une page PHP dans une autre page PHP.
Et c'est très utile ! Concrètement, supposons que sur votre site web il y ait un menu à gauche. Ce menu est affiché sur toutes les pages de votre site.
Jusqu'ici, vous deviez copier-coller ce menu dans toutes les pages, et si vous deviez modifier le menu eh bien il fallait modifier toutes les pages !

Grâce à l'include, vous dites à PHP sur chacune de vos pages : "*Mets ici le contenu de la page menu.php*". PHP va alors "prendre" le contenu de la page menu.php et le mettre là où vous lui avez dit. Ainsi, si vous voulez modifier votre menu, vous modifiez juste menu.php et toutes les pages de votre site web sont automatiquement mises à jour ! C'est vraiment quelque chose de génial, et pour tout vous dire c'est en découvrant ça que j'ai décidé de me mettre au PHP.

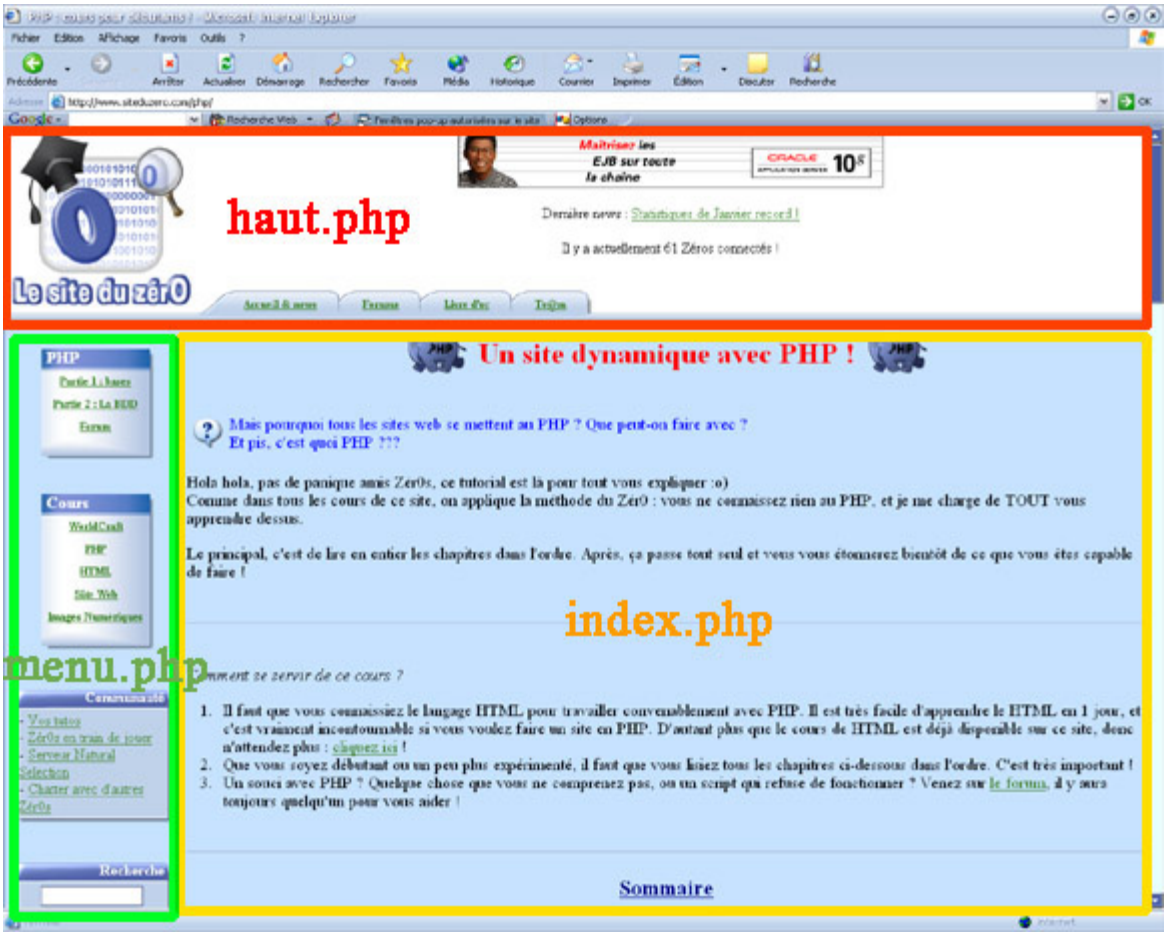
Voici comment on fait pour inclure la page menu.php :

Source 3.1.1 : inclure menu.php

```
<?
include("menu.php");
?>
```

C'est un code tout simple. PHP voit l'instruction include, il va aller chercher la page menu.php et la mettre à la place de cette instruction.

Un exemple concret ? N'allez pas chercher bien loin, regardez ce site web. Oui oui, le Site du Zér0 utilise beaucoup les includes. Voici le sommaire du cours de PHP (page index.php) :



La page index.php contient 2 includes : haut.php (pour le logo, la pub...) et menu.php (le menu du Site du Zér0). Vient ensuite le contenu proprement dit de index.php, c'est-à-dire le sommaire du cours de PHP. Le code PHP de index.php ressemble donc à cela :

Source 3.1.2 : le code de index.php

```
<table width="100%">

<tr>
<td colspan="2">
<?
// On inclue le haut de la page
include("haut.php");
?>
</td>
</tr>

<tr>
<td width="20%">
<?
// Puis on inclue le menu
include("menu.php");
?>
</td>

<td>

<?
// Maintenant on met le code de notre page (ce qu'on veut)
// Ce code peut bien entendu contenir du PHP comme du HTML
?>

<div align="center"><b><font color="red" size="6"><img src="images/elephpant.gif"
align="absmiddle"> Un site dynamique avec PHP ! <img src="images/elephpant.gif"
align="absmiddle"></font></b></div><br>

<br>
<p><table cellspacing="5">

<tr>
<td><img src="/images/qst.gif"></td>
<td><font size="4" color="blue">Mais pourquoi tous les sites web se mettent au PHP ? Que peut-
on faire avec ?<br>Et pis, c'est quoi PHP ???</font></td>
</tr>

</table><p>

Hola hola, pas de panique amis Zér0s, ce tutorial est là pour tout vous expliquer :o)...

</td>
</tr></table>
```

On a en premier les 2 includes (haut.php et menu.php), et après on a mis le code de notre page. Toutes les pages du site fonctionnent comme ça !

Vous remarquerez que j'utilise un tableau pour la présentation, mais vous faites comme vous voulez. Si vous ne comprenez pas ce que font les balises <td> <tr> etc... alors revoyez le cours de HTML.
Ca me sert juste pour organiser la présentation du site.

 Au fait, on peut sans problème mettre du code PHP dans les pages haut.php et menu.php.

Bien, et si nous passions à la pratique ?
On veut par exemple afficher le titre de notre site en haut de toutes les pages. On va créer une page titre.php, qui sera incluse dans toutes les pages. On va aussi créer une page test.php pour tester l'inclusion.

On va mettre dans titre.php ce qu'on veut (HTML, PHP etc...). Pour ma part je fais simple, j'écris juste le nom du site en centré :

Source 3.1.3 : le code de titre.php, la page incluse

```
<div align="center"><h2>Le Site du Zér0</h2></div>
<br />
```

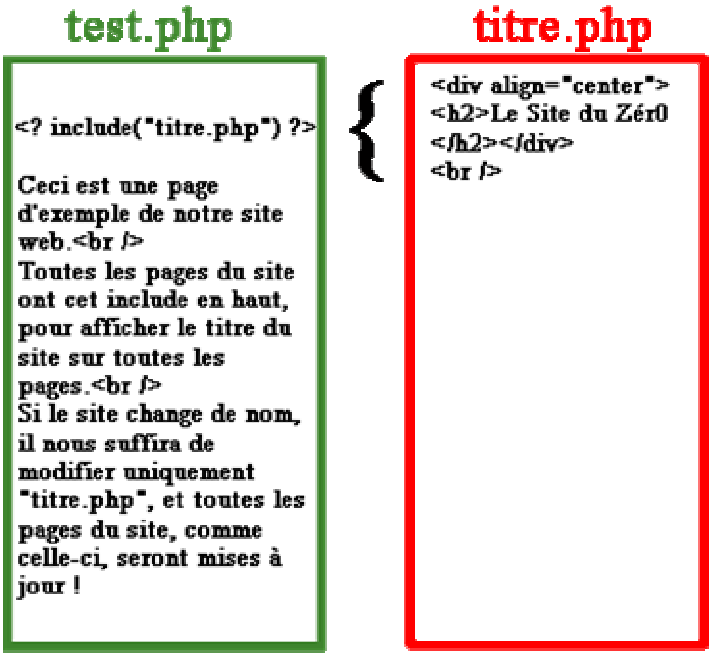
La page test.php est une page d'exemple de notre site. Toutes les pages du site ressembleront à celle-ci :

Source 3.1.4 : le code de test.php

```
<? include("titre.php") ?>

Ceci est une page d'exemple de notre site web.<br />
Toutes les pages du site ont cet include en haut, pour afficher le titre du site sur toutes les pages.<br />
Si le site change de nom, il nous suffira de modifier uniquement "titre.php", et toutes les pages du site, comme celle-ci, seront mises à jour !
```

Faites pareil chez vous, vous allez voir que c'est très facile à faire !

Un petit schéma pour résumer :



Ce schéma, bien que très moche car fait par moi, illustre bien ce qu'il se passe. Lorsque l'internaute demande à voir test.php, l'instruction include est remplacée par le contenu de titre.php.
Ce qui fait qu'à la fin, la page que l'internaute chargera contiendra ce code :

Source 3.1.5 : le code de la page test.php que l'internaute reçoit

```
<div align="center"><h2>Le Site du Zér0</h2></div>
<br />

Ceci est une page d'exemple de notre site web.<br />
Toutes les pages du site ont cet include en haut, pour afficher le titre du site sur toutes les pages.<br />
Si le site change de nom, il nous suffira de modifier uniquement "titre.php", et toutes les pages du site, comme celle-ci, seront mises à jour !
```

C'est très facile à comprendre, avouez !

Voilà, en théorie vous savez tout ce qu'il y a à savoir mais... je ne vais pas vous abandonner là, je ne suis pas comme ça.
En effet, nous allons voir dans une seconde partie de ce chapitre comment mettre en place concrètement des includes sur votre site web.

Bien Utiliser Les Includes

Grosso modo, on peut considérer qu'il y a 2 méthodes pour utiliser les includes sur son site : la brutale et la dangereuse. Je suis plutôt un adepte de la méthode brutale car je préfère être sûr de ne pas laisser une porte grande ouverte aux apprentis hackers.

QUOIIIII ?!!! Que vois-je ?! Qu'entends-je ?!
Je risque de me faire hacker mon site avec les includes et tu me le dis même pas ?

Bah non, faut pas pleurer comme ça voyons.
J'allais justement vous en parler. Suivez avec attention tout ce que je vais vous dire.

Méthode n°1 : La Brutale

On commence par ma préférée : la méthode dite "brutale" (bien entendu ce n'est pas un nom officiel, c'est moi qui l'appelle comme ça).

Cette méthode a un avantage et un défaut :

- **Avantage** : vous n'avez aucun risque de vous faire hacker avec cette méthode (c'est pour ça que je la préfère).
- **Défaut** : si vous voulez changer complètement le design de votre page web, il se peut (je dis bien "il se peut") que vous deviez tout refaire si vous avez utilisé la méthode brutale. Par ailleurs, elle fait un peu moins "pro", mais elle est tellement plus sûre...

Ne tournons pas autour du pot, cette méthode est simple : elle consiste à copier-coller l'instruction include sur toutes les pages web de votre site :

Source 3.1.6 : toutes les pages du site ressemblent à ceci (méthode brutale)

```
<? include("haut.php"); ?>

Ceci est une page X de votre site.<br />
Tout le code de vos pages ressemble à ceci : il y a un include en haut, et un include en
bas.<br />

<? include("bas.php"); ?>
```

Dans les pages haut.php et bas.php, vous mettez ce que vous voulez. Par exemple, dans haut.php je mettrais le titre du site et les premiers tags html : <html>, <head>, <title> etc... Ce qu'on trouve en haut du code d'une page web quoi...
Dans bas.php, vous inscrivez par exemple un copyright, le nom du webmaster, puis vous fermez les balises </body> et </html>

Méthode n°2 : La Dangereuse

De la même manière que la méthode précédente, il y a un avantage et un défaut :

- **Avantage** : on peut facilement changer toute l'apparence de votre site sans problème. Certains trouveront aussi peut-être ce code plus "séduisant" (si toutefois il est possible de trouver un code séduisant).
- **Défaut** : si vous ne faites pas très attention au code que vous écrivez, votre site web sera vulnérable et il sera asssez facile de récupérer votre mot de passe MySQL par exemple...

A noter que cette méthode est utilisée par un bon nombre de sites web, mais encore une fois je ne vous la conseille pas trop.

Le fonctionnement est ici complètement l'inverse : au lieu d'inclure l'en-tête de vos pages, les menus etc... Vous créez une page qui contient tout sauf le corps de votre page : vous mettez donc vos balises <html>, <head>, <title>, vos menus, votre copyright, puis vous fermez les balises </body> et </html>. Et là, vous allez inclure la page que vous souhaitez afficher. Par exemple, si vous voulez inclure la page minichat.php, vous ferez comme ceci :

Source 3.1.7 : approche de la méthode dangereuse

```
<html>

<head>
<title>Mon super site !</title>
</head>

<body>

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


Ce site a été écrit par Mateo21.

</body>

</html>
```

Et si je veux inclure une autre page, je fais comment ? Je refais une page comme celle-là et j'inclue mon autre page cette fois ?

Pas du tout, malheureux !
L'astuce utilisée ici, c'est que l'on va recevoir une variable avec l'url. Vous vous souvenez de index.php?langue=fr&truc=bidule n'est-ce pas ? Eh bien, dans toutes les pages du site, on va transmettre une information qui contiendra le nom de la page à inclure, par exemple :

index.php?page=minichat

On reprend maintenant le code 3.1.7 :

Source 3.1.8 : la méthode dangereuse

```
<html>

<head>
<title>Mon super site !</title>
</head>

<body>

<?
$page = $_GET['page'];
include("$page.php");
?>


Ce site a été écrit par Mateo21.

</body>

</html>
```

Si l'url contient index.php?page=minichat, alors on incluera minichat.php
Si l'url contient index.php?page=news, alors on incluera news.php
Si l'url contient index.php?page=forum, alors on incluera forum.php
Si l'url contient index.php?page=http://www.hacker.com/pagespeciale, alors on incluera http://www.hacker.com/pagespeciale.php

 Euh, attends une minute là, tu peux m'expliquer le dernier exemple ?

Oui, je crois que des explications s'imposent... Regardez en haut de cette page web, vous pouvez modifier facilement l'url, donc modifier facilement la page qui sera incluse !!! Et vu le code PHP qui a été utilisé, on peut très facilement inclure une page située sur un autre site ! Du coup, en modifiant juste l'url de la page, PHP va exécuter cette instruction :
<? include("http://www.hacker.com/pagespeciale.php"); ?>

Qu'est-ce qu'on risque ? C'est simple, je n'ai qu'à modifier l'url pour mettre l'adresse d'un fichier PHP sur un FTP à moi, et c'est VOTRE serveur qui exécutera le code de ma page (pagespeciale.php).

 Et alors ?

Eh bien, je n'ai qu'à dire à PHP : "Donne-moi le mot de passe de ce site"
Et hop, comme ça je peux accéder à un FTP qui ne m'appartient pas, modifier tous les fichiers que je veux, faire un bordel monstre... Que de joyeusetés illégales qui n'ont aucun intérêt, mais ça amuse

certains abrutis (et je pèse mes mots) qui veulent montrer ainsi qu'ils sont "les plus forts". Hum, je m'emporte là !!!!

Bien entendu, on ne peut pas dire à PHP de sortir tous les mots de passe du site comme ça, c'est un peu plus compliqué. Mais, réveillez-vous : vous êtes ici pour apprendre le PHP, pas pour apprendre comment hacker un site web hein ?

Moi, tout ce qui m'intéresse ici, c'est de vous sensibiliser au fait que ce que vous écrivez en PHP peut mettre en danger la sécurité de votre site. Vous venez de le voir sur un exemple concret : je viens de vous faire, sans que vous vous en rendiez compte, une ouverture aux problèmes de sécurité du PHP. Ce sont des problèmes qui ne vous préoccupent pas encore trop pour le moment, mais quand vous serez bons (et vous n'allez pas tarder à l'être, croyez-moi), vous verrez que vous ferez très attention à la sécurité sur votre site.
Pour le moment, je vous rassure, on n'en est pas encore là, alors vous pouvez continuer à lire le tuto PHP tranquillement. Vous apprendrez tout cela petit à petit.

Avant de nous quitter, voyons une des solutions possibles pour résoudre le problème de sécurité (il y en a plusieurs) :

Source 3.1.9 : méthode bien moins dangereuse

```
<html>

<head>
<title>Mon super site !</title>
</head>

<body>

<?
if ($_GET['page'] == "minichat")
{
    include("minichat.php");
}

if ($_GET['page'] == "news")
{
    include("news.php");
}

if ($_GET['page'] == "forum")
{
    include("forum.php");
}

?>


Ce site a été écrit par Mateo21.

</body>

</html>
```

Si un hacker essaie de changer l'url, aucun des if ne sera valable donc rien ne sera inclus. Ouf !
Ça implique de faire autant de if qu'il n'y a de pages sur votre site, c'est pas super pratique... Voilà pourquoi je préfère (et vous conseille d'utiliser) la première méthode !

Voilà, vous savez tout ce qu'il y a à savoir sur les includes.
Comme vous avez pu le constater ça n'est absolument pas sorcier, et pourtant grâce à ce truc on peut déjà rendre son site bien plus agréable !

Faire Joujou Avec Des Variables

Vous avez appris à manipuler des variables dès la partie I. Depuis, vous vous êtes sûrement rendus compte que vous avez besoin de variables dans la totalité de vos scripts PHP.

Le but de ce chapitre est de vous apprendre quelques "techniques" vous permettant d'encore mieux les manipuler :

1. Nous verrons dans un premier temps la concaténation : c'est juste une bonne habitude à prendre qui rendra votre code plus propre.
2. Nous verrons ensuite une série de fonctions toutes prêtes en PHP permettant de travailler sur des variables. Ce sont des outils très pratiques (et faciles à utiliser) dont vous ne pourrez bientôt plus vous passer.
3. Enfin, pour terminer le chapitre en beauté, je vais vous apprendre à faire un truc très tordu : des variables variables !

Que d'émotions en perspective !

La Concaténation

Concaténation.
Derrière ce mot barbare qui semble réservé aux gourous de l'informatique se cache en fait... un point !
Oui oui, vous avez bien entendu, un misérable petit point comme celui-ci . Voyons voir un exemple concret. Regardez-moi ce code, digne des premiers chapitres de ce cours :

Source 3.2.1 : afficher le nom

```
<?
$nom = "Mateo21";
echo "Salut $nom, comment ça va ?";
?>
```

Ca va jusque là, pas trop dur ?
Ca affichera :

Salut Mateo21, comment ça va ?

Eh bien, laissez-moi vous en apprendre une bien bonne. Le code ci-dessous produira exactement le même résultat :

Source 3.2.2 : afficher le nom avec une concaténation

```
<?
$nom = 'Mateo21';
echo 'Salut ' . $nom . ', comment ça va ?';
?>
```

Et ça, c'est ce qu'on appelle une concaténation. Vous avez remarqué le point ? C'est ce qui permet faire le lien entre votre variable et le reste de votre texte.
Autre chose qui a dû vous surprendre : j'utilise désormais des apostrophes à la place des guillemets pour délimiter du texte. On peut toujours utiliser des guillemets, mais avec des concaténations les apostrophes c'est plus propre.
A partir de cet instant, nous allons utiliser uniquement la concaténation. Nous allons utiliser la concaténation et des apostrophes pratiquement tout le temps.

Mais... mais... Tu nous prends pour des abrutis ? Tu nous as appris à faire différemment avant, et maintenant tu veux qu'on utilise ta concaténation et tes apostrophes ?
Et pis, je vois pas ce que ça a de si génial la concaténation moi...

Voilà de bonnes questions !
Il faut savoir que, dans une première version de ce cours PHP, je parlais de la concaténaton dès le début (dans le premier chapitre sur les variables). Et visiblement, ça posait problème : ça faisait trop de choses à la fois au début.
Donc, j'ai préféré vous apprendre une technique plus simple, en attendant. Mais mettez-vous ça dans la tête, la concaténation c'est **=>MIEUX<=**

Désormais, vous verrez que j'utilise le plus souvent des apostrophes à la place des guillemets. Je vous invite à faire de même, nous allons voir que ça a quelques avantages.

La Concaténation

Elle permet de faire plus de choses. Par exemple, comment mélanger deux variables ? Avec une concaténation !

Source 3.2.3 : mélanger deux variables

```
<?
$prenom = 'Jean ';
$nom = 'Dupont';

$nom_complet = $prenom . $nom;
?>
```

$nom_complet vaudra "Jean Dupont".

Ca, vous verrez que c'est très pratique. En plus, c'est facile à faire et à comprendre n'est-ce pas ? On dit à PHP : *$nom_complet vaut $prenom et $nom*
Le "et" correspond au petit point de la concaténation.

On peut aussi faire une concaténation sur une même variable. Tordu, mais pratique là encore :

Source 3.2.4 : concaténation sur une même variable

```
<?
$phrase = 'Je suis ';
$phrase = $phrase . 'un Zér0';
?>
```

A la fin, $phrase vaudra "Je suis un Zér0".

Pourquoi faire une concaténation sur une même variable me direz-vous ? Eh bien, parfois vous avez besoin de mettre beaucoup de texte dans une variable. Plutôt que de tout écrire sur une même ligne, vous passez à la ligne suivante (comme j'ai fait) et vous utilisez une concaténation pour *associer* le contenu de la variable avec la suite de la phrase.

Astuce ! Lorsque vous faites une concaténation sur une même variable, vous pouvez aussi écrire *$phrase .= "un Zér0";*
C'est un raccourci qui vous évite à avoir à écrire $phrase 2 fois sur une même ligne.

On s'arrêtera là pour la concaténation, il n'y a rien de bien compliqué.

Les Apostrophes

Concernant les apostrophes (que j'utiliserai désormais à la place des guillemets), j'imagine que ça doit vous perturber un petit peu. Voici donc quelques points à savoir à propos de ces apostrophes, car je veux que vous compreniez *pourquoi* je préfère les utiliser :

- Premier point, le plus important : si une variable est entre apostrophes, on n'affiche pas son contenu contrairement aux guillemets.
  C'est-à-dire que si on a une variable $var = 'Manger' :
  - echo "$var"; affichera *Manger*
  - echo '$var'; affichera *$var*

  Maintenant qu'on a la concaténation, ça n'est plus un problème. On n'écrira plus jamais de variables entre apostrophes, donc on ne risque pas de voir $var s'afficher.
  Pour afficher le contenu de $var on écrira donc : echo 'Texte' . $var . 'Suite du texte';

- Si vous utilisez des apostrophes, vous n'aurez plus à taper d'antislash \ devant vos guillemets. C'est très utile car en HTML on doit écrire beaucoup de guillemets, ça nous évite d'écire des tonnes d'antislashs. Par exemple, sur ce code on s'est évité 4 antislashs :
  ```
  echo '<img src="monimage.jpg" title="Ceci est une image">';
  ```
- Par contre, vous vous demandez comment on va insérer des apostrophes maintenant ? C'est le revers de la médaille, il faudra cette fois mettre des antislashs devant les apostrophes :
  ```
  echo 'Il l\'a trouvé chez son ami Paco';
  ```
  Mais bon, vu qu'en pratique on est plus souvent amenés à utiliser du HTML avec des guillemets que de longs textes littéraires avec plein d'apostrophes, c'est plus avantageux de se servir des apostrophes pour délimiter son texte.

Voilou c'est tout !
Et rappellez-vous : à partir de maintenant si j'en vois un qui n'utilise pas la concaténation ou qui se sert encore des guillemets, je l'étripe !!!!

Des Outils Très Pratiques

Maintenant, nous allons voir une série de fonctions toutes prêtes en PHP qui travaillent sur des chaînes de caractères.

Une chaîne de caractères, c'est tout simplement une variable qui contient du texte. En anglais on dit *string*
Par exemple : "Ceci est une chaîne de caractères".

Les fonctions que vous allez voir sont toutes très simples à utiliser, encore faut-il les connaître quand on en a besoin.
Je vais vous lister les plus utiles et vous expliquer rapidement leur fonctionnement. Pour connaître toutes les fonctions travaillant sur des chaînes de caractères, n'hésitez pas à consultez le manuel PHP (un peu austère certes, mais terriblement efficace !)

addslashes

Cette fonction ajoute des anti-slashs \ dans votre chaîne. Pourquoi faire ? Pour éviter d'avoir des bugs si votre chaîne contient des guillemets ou des apostrophes. Vous n'en voyez peut-être pas l'utilité maintenant, mais retenez bien que cette fonction existe car vous en aurez forcément besoin plus tard.
On l'utilise comme ceci :

Source 3.2.5 : addslashes

```
<?
$nouvelle_variable = addslashes($ancienne_variable);
?>
```

Par exemple, on a cette chaîne : *Elvis Presley était le "King", y'a aucun doute !*
Après passage à addslashes, ça deviendra : *Elvis Presley était le \"King\", y\'a aucun doute !*

Vous voyez, ça ajoute les anti-slashs juste devant les apostrophes et les guillemets !

stripslashes

Bah là, je vais pas faire long : cette fonction, c'est exactement l'inverse de addslashes. Ca enlève les anti-slashs de votre chaîne.

Source 3.2.6 : stripslashes

```
<?
$nouvelle_variable = stripslashes($ancienne_variable);
?>
```

Si on a la chaîne : *Elvis Presley était le \"King\", y\'a aucun doute !*
Après passage à stripslashes, ça redeviendra : *Elvis Presley était le "King", y'a aucun doute !*

htmlentities

Celle-là, vous l'avez déjà vue si je ne m'abuse.
Elle convertit les caractères HTML d'une chaîne en un code qui ne risque pas de s'exécuter. Très pratique par exemple si vous faites un mini-chat et que vous voulez empêcher vos visiteurs d'utiliser du HTML !

Source 3.2.7 : htmlentities

```
<?
$variable_html = '<i>Ceci est une variable qui contient du HTML</i>';
$variable_sans_html = htmlentities($variable_html);

echo 'Avant : ' . $variable_html . '<br />Après : ' . $variable_sans_html;
?>
```

Remarquez au passage la zolie concaténation.

Il existe d'autres fonctions similaires qui peuvent vous être utiles : *htmlspecialchars* bloque uniquement les caractères les plus utilisés en HTML (< > & " '), tandis que *strip_tags* supprime carrément toutes les balises HTML .

nl2br

Ultra-pratique, on s'en servira dans le prochain chapitre !
La fonction nl2br transforme toutes les "Entrées" qu'a tapé votre visiteur en code HTML "
" (qui correspond à un retour à la ligne).
En effet, comme vous le savez peut-être déjà, en HTML une "Entrée" n'a aucun effet (ça ne crée pas de retour à la ligne). Heureusement qu'il y a nl2br, moi je vous le dis !

Source 3.2.8 : nl2br

```
<?
$ma_variable = 'Ceci est la première ligne.
Ceci est la seconde ligne.
Ceci est la troisième ligne.
Bon on arrête là...';
$ma_variable = nl2br($ma_variable);

echo $ma_variable;
?>
```

Hé ho ! Une minute ! Je vois pas ce qu'il fait de si extraordinaire ton code moi ?!

On essaie sans le nl2br ? Vous allez voir :

Source 3.2.9 : sans nl2br

```
<?
$ma_variable = 'Ceci est la première ligne.
Ceci est la seconde ligne.
Ceci est la troisième ligne.
Bon on arrête là...';

echo $ma_variable;
?>
```

Comme vous pouvez le voir, sans nl2br les retours à la ligne ne se font pas tous seuls ! Eh oui c'est pas la faute à PHP, c'est le langage HTML qui est fait ainsi.

strlen

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

Source 3.2.10 : strlen

```
<?
$phrase = 'Bonjour les Zér0s ! Je suis une phrase !';
$longueur = strlen($phrase);


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

Comptez les caractères si ça vous amuse, il y en a bien 40 je vous assure !

str_replace

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

Source 3.2.11 : str_replace

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

echo $ma_variable;
?>
```

On a besoin d'indiquer 3 paramètres :

1. La chaîne qu'on recherche. Ici, on recherche les "b" (on aurait pu rechercher un mot aussi).
2. La chaîne qu'on veut mettre à la place. Ici, on met des "p" à la place des "b".
3. La chaîne dans laquelle on doit faire la recherche.

Ce qui nous donne "pim pam poum".

str_shuffle

Pour vous amuser à mélanger aléatoirement les caractères de votre chaîne !

Source 3.2.12 : str_shuffle

```
<?
$chaine = 'Cette chaîne va être mélangée !';
$chaine = str_shuffle($chaine);

echo $chaine;
?>
```

strtolower

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

Source 3.2.13 : strtolower

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

echo $chaine;
?>
```

A noter qu'il existe strtoupper qui fait la même chose en sens inverse : minuscules => majuscules

Les Variables Variables

J'en vois déjà qui ouvrent des yeux grands comme ça :
Non non je vous rassure, vous ne voyez pas double !

Il s'agit d'un truc qui va paraître bien tiré par les cheveux au premier abord. Mais, moi qui ne savais pas que ça existait, quand j'ai découvert que PHP savait faire ce genre de trucs j'étais vraiment aux anges.

On va partir d'un problème concret.
Je possède 3 variables : $ville, $pays, $continent. Je veux, en fonction de la valeur d'une autre variable ($afficher), afficher le contenu d'une de ces 3 variables (je vous avais dit que c'était tiré par les cheveux).

SI $afficher vaut "ville", ALORS afficher le contenu de $ville.
SI $afficher vaut "pays", ALORS affiche le contenu de $pays.
SI $afficher vaut "continent", ALORS affiche le contenu de $continent.

Normalement vous êtes capables d'écrire ce code non ? Je pense que vous utiliseriez des conditions :

Source 3.2.14 : des conditions ? Mouais...

```
<?
if ($afficher == 'ville')
{
    echo $ville;
}
elseif ($afficher == 'pays')
{
    echo $pays;
}
elseif ($afficher == 'continent')
{
    echo $continent;
}
?>
```

Taratata... C'est lourd, c'est répétitif, et souvent vous n'aurez pas un problème avec 3 variables mais plutôt 300. Je vous vois mal faire 300 "if" dans votre code.

La solution ? Demander à PHP d'afficher le contenu d'une variable en fonction d'une autre variable. Pour cela, on utilisera un dollar suivi d'accolades (${}). En fait, on écrira *${$afficher}*.
$afficher sera remplacé par sa valeur (par exemple "ville"), et du coup PHP comprendra qu'il s'agit de la variable $ville !

Vu que vous êtes sceptiques (on l'est forcément quand on vient de lire les atrocités tordues que je viens de vous sortir), voici un code source qui fonctionne comme le code 3.2.14 :

Source 3.2.15 : gloire aux variables variables !

```
<?
$afficher = 'ville'; // Modifiez la valeur de $afficher pour voir...

// On définit les 3 variables dont on a parlé
$ville = 'Marseille';
$pays = 'France';
$continent = 'Europe';

echo ${$afficher}; // On affiche la variable dont le nom est "ville" dans notre exemple
?>
```

Je reconnais que c'est difficile à expliquer avec des mots, et pour une fois (je dis bien pour une fois), je crois que c'est plus simple à comprendre en lisant le code source.

Donc, lisez et relisez bien ce code 3.2.15, vous allez rapidement comprendre comment ça marche. Et surtout, faites des tests chez vous : mettez par exemple $afficher = "pays" pour voir ce que ça fait.

On peut aller plus loin. Entre les accolades, vous pouvez mettre le texte que vous voulez, utilisez juste une concaténation dedans !

```
echo ${'mateo_' . $afficher};
```

Ce qui dans notre exemple afficherait le contenu de la variable $mateo_ville !

De ce chapitre, il faut retenir :

- Que la concaténation et les apostrophes c'est bien, il faut les utiliser.
- La liste des fonctions que je vous ai faite sur les chaînes de caractères. L'idéal c'est de connaître ça par coeur (c'est pas bien long) car vous vous en servirez pratiquement tout le temps !

Les variables variables, c'est quelques chose d'un peu plus particulier. Retenez juste que ça existe, et si un jour vous en avez besoin retournez lire ce chapire pour savoir comment on fait.

PHP Et Les Formulaires

Une des applications les plus intéressantes du PHP est que l'on peut travailler sur des formulaires, et de manière très puissante.

Les formulaires sont le seul et unique moyen pour vos visiteurs de rentrer des informations sur votre site, donc de produire *l'interactivité*.
Regardez par exemple, sur un forum on doit rentrer du texte puis cliquer sur un bouton pour envoyer son message. Sur un livre d'or, sur un mini-chat, pareil. On a besoin des formulaires partout.

Vous allez voir qu'il y a de nombreux rappels de HTML dans ce chapitre... Et ce n'est pas un hasard : ici le PHP et le HTML sont très liés.
A quoi sert PHP dans l'histoire ? Il va nous permettre de traiter les données qu'a rentré l'utilisateur. Cela va nous servir de base pour tous nos prochains TP (livre d'or, news...) donc soyez attentifs.

Go !

Fonctionnement Du Formulaire

En HTML, pour dire qu'on va insérer un formulaire on se sert de la balise <form>. On l'utilise de la manière suivante :

Source 3.3.1 : la balise <form> (formulaire.php)

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

On mettra ici les éléments de notre formulaire.<br />
Notez qu'il n'y a pour l'instant pas de PHP.

</form>
```

Ce qu'il faut retenir, c'est qu'on met le contenu de notre formulaire entre les balises <form> et </form>
Il y a 2 attributs intéressants à connaître pour la balise <form> :

- **method="post"** : il faut savoir qu'il y a plusieurs moyens d'envoyer le formulaire (plusieurs "méthodes"). Ne retenez que la méthode "post", c'est la seule qui nous intéressera en PHP. Vous devrez donc toujours mettre method="post" pour vos formulaires !
- **action="cible.php"** : très important. C'est le nom de la page qui sera appelée lorsque l'utilisateur aura envoyé son formulaire (lorsqu'il aura cliqué sur "Envoyer" quoi).
  Par exemple, le code 3.3.1 est situé sur la page formulaire.php ; une fois le formulaire envoyé, ça charge la page cible.php dans laquelle on traitera les informations.

Retenez donc bien que vous travaillez normalement sur 2 pages différentes : la page qui contient le formulaire (formulaire.php dans notre exemple), et celle qui reçoit les données du formulaire pour les traiter (cible.php).


Si vous ne mettez pas d'attribut *action*, alors la page du formulaire sera rechargée (formulaire.php)

Mais du coup vous devez vous demander quelque chose : pour les TP "mot de passe" et "mini-chat", n'a-t-on pas utilisé une seule et même page ? Eh oui en effet, le page cible était la même page que celle où il y avait le formulaire. Par exemple, on avait mis le formulaire dans minichat.php, *mais aussi* le traitement des données dans la même page (car on avait un action="minichat.php" qui renvoyait sur la même page).

Du coup les choses devrait devenir plus claires dans votre tête :

- La première fois que le mini-chat est chargé, on vérifie si des variables renvoyées par le formulaire commme $_POST['pseudo'] existent (c'est à ça que sert isset). Comme la première fois qu'on a chargé la page on n'a rien rentré dans le formulaire, on sait qu'on ne doit pas écrire d'informations dans la base de données (on saute le if).

- Quand on a rentré un message, la page se recharge mais cette fois avec des informations du formulaire entrées par l'utilisateur (comme le pseudo). Du coup, $_POST['pseudo'] existe, DONC on sait qu'on doit enregistrer quelque chose.

Et tout ça se fait sur une seule et même page !

Dans la pratique, vous verrez qu'on préfère utiliser deux pages distinctes car c'est plus simple à gérer (ça nous épargne le isset). Mais souvent, vous serez contraints à travailler sur une même page, donc à utiliser la même technique que dans les TP précédents.

Si vous vous souvenez bien, on avait très rapidement parlé des formulaires avec zone de texte dans le chapitre sur les variables. Je vous avais promis qu'on reviendrait dessus plus tard et qu'on verrait tous les éléments de formulaires que l'on peut traiter avec PHP.
Ce moment est enfin venu !

Les Éléments Du Formulaire

Dans un formulaire, vous le savez peut-être déjà, on peut mettre beaucoup d'éléments différents : zones de textes, boutons, cases à cocher etc etc...
Je vais ici tous les lister et vous montrer comment vous servir de chacun d'eux. Avec ça, vous devriez être parés pour partir à l'assaut des formulaires tous seuls.

Les Petites Zones De Texte

Une zone de texte ressemble à ceci : 
En HTML, on l'insère tout simplement avec la balise :

```
<input type="text">
```

Pour les mots de passe, vous pouvez utiliser type="password", ce qui aura pour effet de cacher le texte rentré par le visiteur.

Mais il y a 2 attributs que vous allez devoir rajouter qui vous seront très importants :

- **name** : c'est le nom de la zone de texte. Choisissez-le bien, car c'est lui qui va produire une variable. Par exemple :
  ```
  <input type="text" name="pseudo">
  ```
  Cela va créer dans cible.php une variable $_POST['pseudo']
- **value** : c'est ce que contient la zone de texte au départ. Par défaut, la zone de texte est vide. Mais il peut être très pratique de pré-remplir le champ : sur le TP mini-chat par exemple, on pourrait facilement écrire automatiquement le pseudo de l'utilisteur comme ça ! Exemple :
  ```
  <input type="text" name="pseudo" value="M@teo21">
  ```

Oui, je sais que vous commencez à vous inquiéter car vous n'avez pas encore vu de PHP pour le moment et vous craignez ce qui risque de vous tomber sur la tronche.
Rassurez-vous, vous ne risquez rien !

En fait, c'est tout bête : le texte que le visiteur aura rentré sera disponible dans cible.php sous la forme d'une variable appelée $_POST['pseudo'].

Et je ne vous apprends rien d'extraordinaire, on avait déjà vu ça dans le chapitre sur les variables. D'ailleurs, pour l'exemple, je ne vais pas m'amuser à en faire un nouveau, je vous ressors celui que je vous avais fait (comment ça "feignasse" ?).

Source 1.4.10 : code de appel.php

```
Cette page, elle aussi, ne contient que du HTML.<br>
Veuillez taper votre prénom :<p>

<form action="cible.php" method="post">
<center>
<input type="text" name="prenom"> <input type="submit" value="Valider">
</center>
</form>
```

Source 1.4.11 : code de cible.php

```
Bonjour !<p>

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

Si tu veux changer de prénom, <a href="appel.php">clique ici</a> pour revenir à appel.php
```

Dans cible.php on a affiché une variable $_POST['prenom'] qui contient ce que l'utilisateur a rentré dans le formulaire.

Comme il s'agissait d'un exemple des premiers chapitres du cours, je n'y avais pas encore parlé de htmlentities (pour rendre le html inoffensif). Mais, par sécurité, vous DEVEZ appliquer un htmlentities à la variable $_POST['prenom']. D'ailleurs, si vous testez mon exemple, vous verrez que moi j'ai pris la précaution d'appliquer un htmlentities !

De même, pour en revenir au mini-chat, on peut sans problème réécrire le pseudo dans le champ "pseudo" si la variable $_POST['pseudo'] existe. Ce qui nous donnerait quelque chose du genre :

Source 3.3.2 : une idée pour le mini-chat

```
<input type="text" name="pseudo"
<?
if (isset($_POST['pseudo'])) // Si on a le pseudo rentré par le visiteur
{
    echo 'value="' . $_POST['pseudo'] . '"'; // On pré-remplit le champ avec le pseudo du
visiteur
}

// Et on n'oublie pas de fermer la balise <input> tout en bas :
?>

>
```

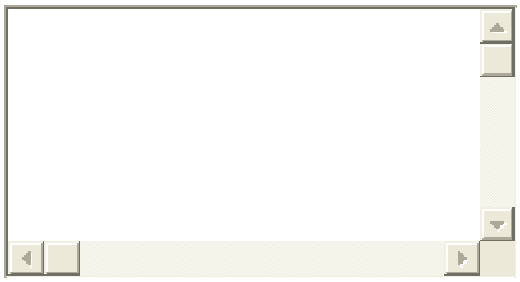
En gros, on a mis du PHP en plein milieu de la balise <input> (oui oui on a tout à fait le droit).
SI on a le pseudo du visiteur, ALORS on rajoute l'attribut *value="Le pseudo"*, ce qui fera que la zone de texte sera pré-remplie.

N'oubliez surtout pas de mettre un petit > tout à la fin pour fermer la balise <input>

Vous reconnaîtrez entre autres l'intérêt des apostrophes et de la concaténation pour séparer le texte des variables. Grâce à ça on n'a pas eu à mettre d'antislash devant les guillemets.

Les Grosses Zones De Texte

La grosse zone de texte (qu'on appelle aussi "Zone de saisie multiligne"), ça ressemble à ceci :



On peut y écrire autant de lignes que l'on veut. C'est plus adapté si le visiteur doit écrire un long message par exemple.

On va utiliser le code HTML suivant pour insérer cette grosse zone de texte :

Source 3.3.3 : une grosse zone de texte

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

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

Chose plus particulière : il n'y a pas d'attribut *value*. En fait, le texte par défaut est ici écrit entre le <textarea> et le </textarea>. C'est plus pratique du coup pour faire un echo au milieu.
Si vous ne voulez rien mettre par défaut, alors n'écrivez rien entre <textarea> et </textarea>

La Liste Déroulante

La liste déroulante, c'est ça : 

On utilise le code HTML suivant :

Source 3.3.4 : la liste déroulante

```
<select name="choix">
    <option value="choix1">Choix 1</option>
    <option value="choix2">Choix 2</option>
    <option value="choix3">Choix 3</option>
    <option value="choix4">Choix 4</option>
</select>
```

Tout bêtement, on utilise la balise <select> à laquelle on donne un nom (ici : "choix").
On écrit les différentes options disponibles...
Puis on referme la balise avec </select>.

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

Un truc important qu'il peut être utile de savoir faire, c'est de définir le choix par défaut. Normalement c'est le premier, mais si vous rajoutez l'attribut *selected* à une balise <option>, alors ce sera le choix par défaut. On pourrait par exemple écrire :

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

Les Cases À Cocher

Une case à cocher ressemble à ceci :

 Case à cocher

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

Source 3.3.6 : les cases à cocher

```
<input type="checkbox" name="case"> Ma case à cocher
```

Là encore, on donne un nom à la case à cocher (ici : "case"). Ce nom va générer une variable dans la page cible, par exemple $_POST['case'].

- Si la case est cochée, alors $_POST['case'] aura pour valeur "on".
- Si elle n'est pas cochée, alors $_POST['case'] ne contiendra rien (NULL).

Si vous voulez que la case soit cochée par défaut, il faudra lui rajouter l'attribut *checked*. Par exemple :

```
<input type="checkbox" name="case" checked>
```

On aura du coup une case déjà cochée, comme celle-ci : 

Les Boutons D'Option

Les boutons d'option fonctionnent par groupes de 2 minimum. Par exemple :

Aimez-vous les frites ?  *Oui*  *Non*

Le code correspondant à cet exemple est le suivant :

Source 3.3.7 : les boutons d'option

```
Aimez-vous les frites ?
<input type="radio" name="frites" value="oui" checked> Oui
<input type="radio" name="frites" value="non"> Non
```

Comme vous pouvez le voir, les deux boutons d'option ont le même nom ("frites"). C'est très important, car les boutons d'options fonctionnent par "groupes" : tous les boutons d'option d'un même groupe ont le même nom.
Cela permet au navigateur de savoir quels boutons d'option désactiver quand on active un autre bouton d'option du groupe. Il serait bête en effet de pouvoir sélectionner "Oui" et "Non" à la fois !

Pour pré-cocher l'un de ces boutons d'option, vous faites pareil que pour les cases à cocher : vous rajoutez un checked. Ici, commme vous pouvez le voir, "Oui" est sélectionné par défaut.

Dans la page cible, une variable $_POST['frites'] sera créée. Elle aura la valeur du bouton d'option choisi par le visiteur. Si on aime les frites, alors on aura $_POST['frites'] = 'oui'.
Il faut bien penser à remplir l'attribut "value" du bouton d'option car c'est lui qui va déterminer la valeur de la variable.

Les Champs Cachés

Si y'a un truc bien pratique avec les formulaires et php, ce sont les champs cachés.

En quoi ça consiste ? C'est un code dans votre formulaire qui n'apparaîtra pas aux yeux du visiteur, mais qui va quand même créer une variable avec une valeur.
Je m'explique : supposons que vous ayez besoin de "retenir" que le pseudo du visiteur est "Mateo21".
Vous allez taper ce code :

Source 3.3.8 : les champs cachés

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

A l'écran, vous ne verrez rien.
Mais dans la page cible, une variable $_POST['pseudo'] sera créée (correspondant à *name*), et elle aura la valeur "Mateo21" (correspondant à *value*) !

C'est apparemment inutile, mais vous verrez que lorsque vous commencerez à créer des formulaires vous en aurez vite besoin.

Petit Exercice

Pour s'entraîner, on va travailler sur une liste déroulante. On va demander au visiteur quelle est sa couleur préférée. Lorsqu'il aura choisi, on doit arriver à faire 2 choses :

1. On va écrire sa couleur préférée (à l'aide d'un *echo* tout bête)
2. On va faire de cette couleur le choix par défaut de la liste déroulante

On va faire appel à un vieux truc qu'on n'a pas utilisé depuis la partie I : les fonctions que vous pouvez définir vous-même. On va définir une fonction appelée choixParDefaut qui va renvoyer 'selected' si la couleur qu'on lui donne est bien le choix de l'utilisateur, ou qui ne renvoie rien si ce n'est pas sa couleur préférée.

Allez, on arrête de bavarder et on code !

Source 3.3.9 : la liste déroulante optimisée

```
<?
function choixParDefaut($couleur) // Création de la fonction
{
$par_defaut = ''; // On crée une variable (vide par défaut) que l'on retournera à la fin

    if (isset($_POST['couleur'])) // Si le visiteur a choisi une couleur
    {
        if ($_POST['couleur'] == $couleur) // Si cette couleur correspond à la couleur que
l'on est en train de traiter
        {
            $par_defaut='selected'; // Alors on modifie la variable que l'on retournera et on
lui met selected
        }
    }

return $par_defaut; // On ne retourne rien si ce n'était pas la couleur choisie, selected si
c'était la bonne couleur
}


// -------------- Fin de la fonction ---------------


if (isset($_POST['couleur'])) // On vérifie si le visiteur a déjà choisi une couleur
{
    echo 'Votre couleur préférée est le : ' . htmlentities($_POST['couleur']) . '<br />';
}
?>


Quelle est votre couleur préférée ?<br />

<form method="post">
<select name="couleur">
    <option value="Bleu" <? echo choixParDefaut('Bleu'); ?>>Le Bleu</option>
    <option value="Marron" <? echo choixParDefaut('Marron'); ?>>Le Marron</option>
    <option value="Vert" <? echo choixParDefaut('Vert'); ?>>Le Vert</option>
    <option value="Rose" <? echo choixParDefaut('Rose'); ?>>Le Rose</option>
</select>
<input type="submit" value="OK">
</form>
```

J'imagine votre tête lorsque vous avez regardé ce code :
D'un côté, ça ne m'étonne pas que ce code 3.3.9 vous ait fait un peu peur. D'un autre, évitez de me le dire parce que je risquerais de ma fâcher tout rouge !

Pourquoi ? Parce que j'ai "juste" utilisé une fonction, et c'est quelque chose que je vous ai appris dans un des premiers chapitres de ce cours de PHP !!!
Donc, en toute logique, vous *devriez* être capables de comprendre tout cela.

Je vais quand même vous expliquer rapidement comment ça marche, j'ai pas envie que vous me jetiez des tomates pour ça !
En gros, oubliez le haut de ce code. Regardez après le commentaire "Fin de la fonction". Si l'utilisateur a fait un choix, alors on affiche quelle est sa couleur préférée.

Alerte rouge : quand vous affichez (ou enregistrez) les résultats d'un formulaire, prenez l'habitude de TOUJOURS appliquer un htmlentities. Et quand je dis toujours, c'est tout le temps : même sur une liste déroulante ou un champ caché, un visiteur peut modifier la source pour aller mettre du html ou du javascript !

Ensuite, on affiche notre formulaire tout bête. Comme on veut recharger la même page, je n'ai pas mis d'attribut *action*.
Pour chaque option, j'affiche ce que me renvoie ma fonction choixParDefaut quand je lui donne la couleur 'Vert' par exemple. Soit la fonction renvoie 'selected', dans ce cas on affiche selected et ce sera le choix par défaut, soit la fonction ne renvoie rien, donc ce ne sera pas le choix par défaut.

Si vous ne comprenez pas bien comment marche ma fonction, je vous conseille vivement de relire le chapitre sur les fonctions.
N'hésitez pas à passer un peu de temps sur ce code pour comprendre comment il marche, car je le trouve très intéressant. Si vous avez compris ça, vous avez tout compris !

Les Dates

Après ce TP de livre d'or, nous allons faire une pause en passant à quelque chose pas prise de tête : les dates avec PHP.

Ce chapitre est constitué de 2 parties :

- On verra d'abord la liste des possibilités de la fonction date, pour récupérer le jour, le mois etc... Oui je sais, on en avait déjà un peu parlé dans la partie I, c'était pour vous montrer un exemple de fonction toute prête en PHP. Mais là, on va vraiment détailler toutes les possibilités.
- Ensuite, on verra ce qu'on appelle les *timestamp*, qui nous seront bien utiles pour faire des calculs sur des dates !

En plus, je vais vous dire quelque chose qui va vous motiver : en maîtrisant les dates, vous aurez le niveau pour attaquer un TP "Système de news" !

La Fonction Date

Quand on veut utiliser une date en PHP, c'est une fonction presque incontournable.
En fait, *date* est une fonction à tout faire : si vous savez bien l'utiliser, vous pourrez afficher n'importe quel élément de date, dans tous les calendriers du moooonde !

Pour voir comment elle fonctionne, basons-nous sur cet exemple :

Source 3.4.1 : afficher le jour du mois

```
<?
$jour = date('d');
echo 'Aujourd\'hui, nous sommes le : ' . $jour;
?>
```

On donne une lettre à la fonction *date*. Le résultat renvoyé dépend de la lettre que vous donnez à *date*. Ici, j'ai mis "d" : ça signifie "Renvoyer le numéro du jour".

 Mais... Je pouvais pas deviner moi que "d" signifiait "Renvoyer le numéro du jour" ?!

Non en effet, vous ne pouviez pas deviner.

date peut renvoyer beaucoup de valeur différentes, je ne vais pas toutes vous les lister car certaines ne vous seront vraiment pas utiles.
En voici déjà un bon paquet, et la signification qui va avec :

Lettre	Signification	Valeurs possibles	Valeur actuelle
s	Secondes	00 à 59	33
i	Minutes	00 à 59	05
H	Heure	00 à 23	01
I	Indique si l'heure d'été est activée (1 = oui, 0 = non)	0 ou 1	1
O	Différence d'heures avec l'heure GMT (Greenwich)	-1200 à +1200	+0200
d	Jour du mois	01 à 31	04
m	Mois de l'année	01 à 12	06
Y	Année, sur 4 chiffres	Beaucoup de possibilités	2004
y	Année, sur 2 chiffres	Beaucoup de possibilités	04
L	Indique si l'année est bissextile (1 = oui, 0 = non)	0 ou 1	1
l	Jour de la semaine écrit en anglais	Sunday à Saturday	Friday
F	Mois écrit en anglais	January à December	June
t	Nombre de jours dans le mois	28 à 31	30
w	Numéro du jour de la semaine	0 (dimanche) à 6 (samedi)	5
W	Numéro de la semaine dans l'année	01 à 52	23
z	Numéro du jour de l'année	0 à 366	155

Comme vous pouvez le constater, le plus embêtant avec *date* c'est que la fonction est faite pour... des anglais. Il n'y a pas moyen qu'elle affiche les jours de la semaine en français.
A partir de là, ou vous vous contentez des mots anglais, ou vous créez vous-même une fonction qui transforme les dates anglaises en français.

Je précise aussi que c'est l'heure du serveur qui est renvoyée, et non pas celle du client. Le serveur du Site du Zér0 étant basé à Paris, si vous habitez ailleurs dans le monde il ne faut pas vous étonner si les valeurs ne sont pas les mêmes que chez vous.

Bon, avec ce tableau vous avez toutes les informations nécessaires pour travailler sur des dates.
Mais vous n'avez pas encore tout vu : on peut donner plusieurs lettres à la fois à *date*, comme ceci :

Source 3.4.2 : plusieurs lettres à la fois

```
<?
echo 'Aujourd\'hui, nous sommes le : ' . date('d/m/Y');
?>
```

Vous avez peut-être été surpris de voir que je n'ai pas utilisé de variable cette fois pour la concaténation. C'est normal : j'ai tout à fait le droit d'utiliser directement une fonction au beau milieu d'une concaténation comme je viens de faire.

date a créé une chaîne de caractères qui contient jour/mois/année. En fait, vous pouvez mettre ce que vous voulez dans *date*, dès que la fonction rencontre une lettre qu'elle connaît elle la remplace par la valeur correspondante.
Cela veut dire que vous pouvez mettre des espaces, des tirets, ou des slashs comme j'ai fait pour séparer les éléments de date. C'est pratique, ça nous permet de n'avoir à appeler la fonction qu'une seule fois.

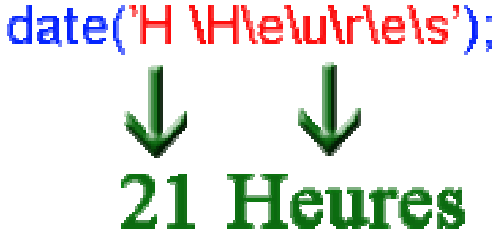
Et si je veux écrire la lettre H dans *date* sans que cette lettre soit remplacée par l'heure ?

Bah là, c'est délicat : il va falloir mettre un antislash \ devant chaque lettre que vous ne voulez pas voir remplacée.
Par exemple :

Source 3.4.3 : des lettres non remplacées

```
<?
echo 'Il est ' . date('H \H\e\u\r\e\s');
?>
```

Dans cet exemple 3.4.3, la fonction *date* est interprétée comme ceci :



Comme vous pouvez le constater, il est plus simple de faire un echo pour écrire "Heures", mais cette technique pourra vous être utile à l'occasion, donc je vous l'apprends.

C'est tout pour *date*, il n'y a pas grand chose d'autre à redire dessus, c'est tout simple à utiliser !

Le timestamp

Bon, jusqu'ici il n'y avait rien de bien extraordinaire. Vous avez juste vu que PHP savait donner tout ce dont vous avez besoin pour afficher la date.
Mais vous n'avez pas vu le plus intéressant : le **timestamp**.

Récupérer Le timestamp

 C'est quoi un timestamp ?

Un timestamp, c'est un nombre.
C'est le nombre de secondes écoulées depuis le 1er Janvier 1970 à Minuit.

Pourquoi depuis le 1er Janvier 1970 à Minuit ? C'est symbolique, il fallait bien prendre un point de départ.
En fait, ça représente le début de l'époque où le système d'exploitation Unix a été créé. Peut-être avez-vous déjà entendu parler de Unix ? Il est à la base de Linux, un système d'exploitation comme Windows (sauf qu'il est gratuit, mais ne nous égarons pas).

Bref, le 1er Janvier 1970 à Minuit, le timestamp avait pour valeur 0. Aujourd'hui, beaucoup beaucoup de secondes se sont écoulées, vous devez vous en douter.
Pour connaître le timestamp actuel en PHP, on utilise la fonction *time* (qui n'a besoin d'aucun paramètre) :

Source 3.4.4 : le timestamp actuel

```
<?
echo 'Le timestamp actuel est : ' . time();
?>
```

Si vous vous amusez à recharger la page chaque seconde, vous allez voir que le timestamp n'arrête pas d'augmenter. Eh oui, le timestamp est un nombre qui devient chaque seconde plus gros !

Bon, à partir de là vous devez vous dire que le timestamp, c'est bien rigolo, mais ça sert pas à grand chose.
Faux : au contraire ça va vous être très utile, nous allons voir pourquoi...

Le timestamp Avec La Fonction date

Il est possible de fournir une second paramètre à *date* (après les lettres) : le timestamp sur lequel vous voulez obtenir des informations.

Par défaut, *date* utilise le timestamp actuel : elle renvoie donc l'heure actuelle, le jour actuel etc... Mais si vous lui donnez un timestamp, elle fera des calculs sur ce moment-là.

Allez, pour faire un test grandeur nature, je vous donne en =>**exclusivité mondiale**<= le timestamp qu'il était au moment où j'ai écrit ces lignes.

```
1080513608
```

On a donc un timestamp, et on va extraire toutes les informations qu'on veut dessus :

Source 3.4.5 : tout savoir sur mon timestamp

```
<?
$timestamp = 1080513608; // C'est l'heure qu'il était quand j'écrivais le tutorial !
?>

Voici plein d'infos sur mon timestamp :<br />

<ul>
<li>M@teo a écrit ces lignes le <? echo date('d/m/Y', $timestamp); ?></li>
<li>Ce jour-là était un <? echo date('l', $timestamp); ?> (désolé, c'est en anglais ;o)</li>
<li>Il était exactement : <? echo date('H\h i\m\i\n s\s', $timestamp); ?> (rhoo l'insomniaque
!)</li>
<li>Il y avait <? echo date('t', $timestamp); ?> jours ce mois-ci.</li>
<li>C'était le <? echo date('z', $timestamp); ?>ème jour de l'année !</li>
</ul>
```

Pour l'affichage du jour en anglais, ne confondez pas : il s'agit d'un L minscule et non pas du chiffre 1 !

La différence par rapport à ce qu'on a vu au début de ce chapitre, c'est qu'on a donné un deuxième paramètre à la fonction date : ce paramètre, c'est un timestamp. En gros, quand on fait :
`echo date('d');`
... on dit à PHP :

"Affiche-moi le numéro du jour actuel"

Mais si on rajoute un timestamp en deuxième paramètre :
`echo date('d', $timestamp);`
... alors là on dit à PHP :

"Affiche-moi le numéro du jour qu'il était au moment de ce timestamp"

Ca, ça va être un truc très très pratique !
Par exemple, lorsque vous écrirez une news, il vous suffira d'enregistrer juste le timestamp, et vous serez capables grâce à ce nombre de ressortir toutes les infos possibles et imaginables dessus : le jour où la news a été postée, l'heure qu'il était etc... C'est donc très très puissant !

Récupérer Le timestamp À Partir D'Une Date

Enfin, une dernière chose qu'il peut être très utile de savoir faire : vous aimeriez connaître le timestamp qu'il était le 5 Février 1998 à 13h 45min 26s (très précisément).

Pour récupérer le timestamp correspondant, on va utiliser la fonction *mktime*. On va lui donner en paramètre une date, et elle va nous ressortir le timestamp correspondant.
Cette fonction peut prendre pas mal de paramètres, en voici la liste dans l'ordre :

```
$timestamp = mktime(heure, minutes, secondes, mois, jour, an, heure d'hiver);
```

Dans la pratique, vous pouvez oublier le dernier paramètre (heure d'hiver) qui ne vous sera pas utile en général. Sachez qu'il faut mettre 1 si l'heure d'hiver est activée, 0 si elle ne l'est pas.
Mais passons. Si on enlève l'heure d'hiver pour éviter de s'embrouiller, il reste 6 paramètres possibles qu'on retiendra :

```
$timestamp = mktime(heure, minutes, secondes, mois, jour, an);
```

Pour bien comprendre, voici un exemple. Je veux toujours le timestamp du 5 Février 1998 à 13h 45min 26s (oui je suis têtu), je vais écrire le code suivant :

Source 3.4.6 : le timestamp du 5 Février 1998 à 13h 45min 26s

```
<?
$vieux_timestamp = mktime(13, 45, 26, 2, 5, 1998);
echo 'Le timestamp du 05/02/1998 à 13h 45min 26s était : ' . $vieux_timestamp;
?>
```

Attention, les anglais n'écrivent pas leurs dates au format jour/mois/an mais au format mois/jour/an !
C'est source de confusion : si vous regardez bien l'exemple ci-dessus, j'ai mis le mois avant le jour dans la liste des paramètres !

Si vous ne voulez pas renseigner l'heure, mais que vous voulez juste obtenir le timestamp d'une date... ce n'est pas possible (il y a plein de timestamp différents dans une journée je vous rappelle !)
La solution communément employée, c'est de se baser sur Minuit :

```
mktime(0, 0, 0, 2, 5, 1998);
```

C'est le timestamp du 5 Février 1998 à Minuit. Ce n'est peut-être pas aussi "précis" que l'exemple du dessus me direz-vous, mais on n'a parfois pas besoin d'autant de précision.

Un Petit Exemple Pratique ?

On veut retrouver le nom du jour où vous êtes né. On va pour cela créer un formulaire pour récupérer 3 variables : le jour, le mois, et l'année de naissance.
Avec mktime, on récupère le timestamp correspondant.
A partir de là, on a plusieurs possibilités pour la fonction *date* :

- Soit on utilise la lettre l (L minuscule) et on obtient le nom du jour en anglais.
- Soit on est têtus et on veut l'obtenir en français, dans ce cas on récupère juste le numéro du jour de la semaine avec la lettre w (0 = Dimanche, 1 = Lundi...)

Vous vous en doutez, on va utiliser la seconde solution (en plus comme ça vous allez comprendre comment on fait pour "traduire" la date en français.

L'idée est la suivante : on va créer un array qui associera le numéro 0 au texte "Dimanche", le numéro 1 au texte "Lundi" etc... Revoyez le début du chapitre sur les array si vous avez oublié, c'est vraiment tout simple.
Après, vu qu'on a le numéro du jour, on donne ce numéro à l'array et on récupère le nom du jour en français qui correspond !

Regardez bien ce code :

Source 3.4.7 : le jour de votre naissance

```
<?
if (isset($_POST['jour']) AND isset($_POST['mois']) AND isset($_POST['an']))
{
    // Le visiteur vient d'entrer sa date de naissance, on va calculer le jour qu'il était.

    // On calcule le timestamp correspondant à la date entrée
    $timestamp_naissance = mktime(0, 0, 0, $_POST['mois'], $_POST['jour'], $_POST['an']);
    // On récupère le numéro du jour correspondant au timestamp (0, 1, 2, 3...)
    $numero_jour = date('w', $timestamp_naissance);

    // On crée un array pour numéroter les jours (0 => Dimanche, 1 => Lundi...)
    $jours = array('Dimanche', 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi');
    // On récupère le nom du jour en français grâce à l'array qu'on vient de créer
    $jour_naissance = $jours[$numero_jour];

    // Puis on affiche le résultat
    echo 'Vous êtes né un ' . $jour_naissance;
}
else // Sinon, c'est que le visiteur n'a pas encore entré sa date de naissance, on affiche le
formulaire
{
?>
Indiquez votre date de naissance (jj/mm/aaaa) :<br />
<form method="post">
<input type="text" name="jour" size="2" maxlength="2"> /
<input type="text" name="mois" size="2" maxlength="2"> /
<input type="text" name="an" size="4" maxlength="4"><p>
<input type="submit" value="Envoyer">
</form>
<?
}
?>
```

Avec un peu de logique, vous allez vite comprendre comment ça marche.
Si vous avez revu le début du chapitre sur les array, vous avez donc bien compris qu'on a associé le numéro 0 à "Dimanche", 1 à "Lundi" etc...

La ligne vraiment intéressante dans notre code, c'est celle-ci :
`$jour_naissance = $jours[$numero_jour];`
On a $numero_jour. On donne ce numéro à notre array $jours, et ça nous ressort le texte qui correspond à ce numéro. Par exemple, si le numéro du jour est 2, alors $jour_naissance aura pour valeur "Mardi" !

Si ce petit code arrive à faire quelque chose d'assez impressionnant, il a un sacré défaut : il ne fonctionne pas si vous êtes né avant 1970 !
En effet, le timestamp n'existe que depuis le 1er janvier 1970, donc si vous tapez 1969 ça ne marchera pas.

Par ailleurs, il est bon de savoir que le timestamp devient de plus en plus gros, et ce nombre sera tellement gros en 2037 que ça ne marchera plus. En clair, le timestamp fonctionne donc entre 1970 et 2037 !

Voilà tout à fait le genre de chapitre que je prends plaisir à rédiger : c'est assez facile à comprendre et pourtant riche en informations.

Vous en apprenez beaucoup, et c'est d'autant plus intéressant quand on sait que ça va nous être utile tout de suite.
Eh oui, ladies and gentlemen, nous allons maintenant passer au TP que vous attendiez tous : un système de news pour votre site !

Les Variables Superglobales

Dans ce chapitre, nous allons travailler sur ce qu'on appelle les variables "superglobales".
A votre niveau, il devient important de savoir qui elles sont et comment on peut les utiliser. Vos scripts vont énormément gagner en puissance grâce à ces variables.

Nous allons travailler dans cet ordre :

- Je vais vous présenter les superglobales, pour que vous sachiez ce que c'est exactement.
- Ensuite, je vous montrerai le fonctionnement de 2 variables superglobales (les plus intéressantes) :
  - Nous étudierons les **sessions**, un système puissant et facile à utiliser dont vous allez sûrement vous servir sur votre site.
  - Enfin, nous étudierons **les cookies** qui, pour ceux qui ne le savent pas déjà, permettent de conserver des informations sur un visiteur même lorsqu'il a quitté votre site.

Bonne lecture !

Présentation Des Superglobales

Je vais vous en apprendre une bien bonne : vous avez déjà manipulé des variables superglobales... sans le savoir.

Ces variables un peu "spéciales" sont faciles à reconnaître. Voici 3 points pour les identifier :

- Elles sont écrites en majuscules et commencent toutes par un underscore _ (le trait de soulignement).
  $_GET et $_POST ça vous dit quelque chose ? Eh oui, ce sont ce qu'on appelle des variables superglobales, et vous les avez déjà utilisé de nombreuses fois.
- Autre point important : les superglobales sont toutes des array.
  Pour ceux qui auraient un petit trou de mémoire, les array sont des variables sous forme de "tableau", facilement reconnaissables grâce aux crochets (ex : $_GET['page']).
  Revoyez le chapitre sur les array de la partie I si vous en avez vraiment besoin.
- Enfin, ces variables sont automatiquement créées par PHP à chaque fois qu'une page est chargée. Ces variables existent donc sur toutes les pages et sont accessibles partout : au milieu de votre code, au début, dans les fonctions etc…

Le mot "global" en PHP signifie que la variable est accessible partout.
Une variable créée dans une fonction par exemple n'est pas accessible partout. Elle n'existe que dans la fonction. Une variable dite globale est accessible partout dans votre code (au début, à la fin...)

Bien, maintenant que vous savez les repérer, que diriez-vous si je vous listais toutes les superglobales qui existent en PHP ?
Il existe d'autres superglobales en PHP, en plus de $_GET et $_POST. Soyons francs : il y en a qui vont beaucoup nous intéresser, mais il y en a aussi qu'on ne touchera jamais.

- **$_SERVER** : ce sont des valeurs utiles que nous donne le serveur.
  Pour accéder à ces informations, il faut indiquer ce qu'on demande exactement entre crochets (vu que c'est un array). Il y a plein de choses disponibles qui ne nous intéressent pas... par contre en voici quelques-unes dont vous aurez peut-être à vous servir :
  - $_SERVER['PHP_SELF'] : c'est le chemin de la page que vous êtes en train d'exécuter, par rapport à la racine de votre site web.
    Exemple : si vous êtes sur la page http://www.monsite.com/scripts/monscript.php, alors $_SERVER['PHP_SELF'] aura pour valeur : /scripts/monscript.php
  - $_SERVER['HTTP_REFERER'] : c'est l'url de la page qui a amené le visiteur sur la page courante. Cela peut être utile notamment pour faire des statistiques : vous saurez par exemple que le site "supersite.com" a fait un lien vers votre site et vous amène des visiteurs.

    
    La documentation de PHP nous avertit clairement que vous ne pouvez pas vous fier à 100% à $_SERVER['HTTP_REFERER'] car le client peut très facilement refuser d'envoyer cette information ou même la modifier.
    Bref, n'ayez pas trop confiance en elle.

  - $_SERVER['REMOTE_ADDR'] : sans aucun doute l'information la plus intéressante de $_SERVER. Elle nous donne l'adresse IP du client qui a demandé à voir la page.
    On se servira de cette variable plusieurs fois dans les prochains TP (pour repérer un même visiteur), donc souvenez-vous qu'elle existe !
- **$_ENV** : ce sont des variables d'environnement, toujours données par le serveur. Plus précisément, le système d'exploitation (Linux) donne ces informations.
  Mais bon, il n'y a rien de vraiment bien utile et de toute façon je suis incapable de vous donner une liste de ce que renvoit cette superglobale. Donc on l'oublie !
- **$_GET** : vous la connaissez bien, c'est elle qui vous donne les valeurs des informations indiquées dans l'url.
  Par exemple, si la page appelée est :
  `http://www.site.com/mapage.php?jour=18&mois=avril&annee=2000`
  ... on aura une superglobale $_GET découpée en 3 parties :
  - $_GET['jour'] = "18"
  - $_GET['mois'] = "avril"
  - $_GET['annee'] = "2000"
- **$_POST** : c'est là-dedans que vous venez récupérer les informations issues d'un formulaire.
  Bon je passe, on y a déjà assez travaillé comme ça dans les chapitres et TP précédents.
- **$_FILES** : cette superglobale est utilisée lorsqu'on envoie des fichiers sur le serveur à partir d'un formulaire. Oui oui, c'est possible, mais on l'étudiera plus tard.
- **$_SESSION** : c'est là-dedans que l'on retrouve les variables de session. Nous allons voir ce qu'est une session en PHP plus loin dans ce chapitre.
- **$_COOKIE** : de même, c'est là-dedans que l'on retrouve les valeurs des cookies enregistrés sur l'ordinateur du visiteur. Nous étudierons les cookies dans ce chapitre là-aussi.

En clair, si je résume : on connaît déjà $_GET et $_POST, on retient que $_SERVER['REMOTE_ADDR'] donne l'adresse IP du visiteur, et on se prépare à étudier $_SESSION et $_COOKIE dans la suite de ce chapitre.

Les Sessions

Les sessions sont un moyen de conserver des variables sur toutes les pages de votre site.
Jusqu'ici, on était arrivés à passer des variables de page en page via la méthode GET (en modifiant l'url : page.php?variable=valeur) et via la méthode POST (un formulaire quoi).

Mais si on veut transmettre une ou plusieurs variables sur TOUTES les pages de son site, c'est vraiment la galère avec GET et POST... D'où l'invention des sessions.

Comment ça marche ?

1. Un visiteur se connecte. On demande à créer une session pour lui : PHP génère un numéro. Pour cela, on utilise la fonction *session_start()*. Ce numéro est souvent très gros et écrit en hexadécimal. Par exemple : a02bbffc6198e6e0cc2715047bc3766f

   
   Ce numéro sert d'identifiant et est appelé "ID de session" (ou PHPSESSID). PHP transmet automatiquement cet ID de page en page en utilisant un cookie ou via l'url (ex : mapage.php?PHPSESSID=a02bbffc6198e6e0cc2715047bc3766f).

2. A partir de là, c'est du tout bon : on peut créer une infinité de variables de session. Par exemple : $_SESSION['login'] contient le login du visiteur, $_SESSION['password'] contient le mot de passe etc...
   L'avantage, c'est que **le serveur conserve ces variables même lorsque la page PHP a fini d'être générée**. Ce qui veut dire que, quelle que soit la page de votre site, vous pourrez récupérer par exemple le login et le mot de passe du visiteur !
3. Lorsque le visiteur se déconnecte (il a cliqué sur un bouton "Déconnecter" ou est resté inactif trop longtemps), alors la session est fermée avec *session_destroy()*

Ca a l'air compliqué comme ça, mais en fait c'est d'une simplicité à en pleurer.

Le seul truc qu'il ne faut pas oublier de faire, c'est d'appeler *session_start()* sur chacune de vos pages AVANT d'écrire le moindre code HTML.
Si vous oubliez *session_start()*, vous ne pourrez pas accéder aux variables superglobales $_SESSION.


Faites très attention : appelez *session_start* tout au début de vos pages PHP. Ne mettez la balise <html> qu'après, sinon vous aurez des problèmes avec votre session.

Après, vous pouvez utiliser les variables $_SESSION comme des variables normales.
Un petit exemple ?

Source 3.5.1 : création de variables de session (index.php)

```
<?
session_start(); // On démarre la session AVANT toute chose

// On s'amuse à créer quelques variables de session :
$_SESSION['prenom'] = 'Jean';
$_SESSION['nom'] = 'Dupont';
$_SESSION['age'] = 24;

// Maintenant que le session_start est fait, on peut taper du code HTML
?>

<html>

<head>
    <title>Titre de ma page</title>
</head>

<body>
Salut <? echo $_SESSION['prenom']; ?> !<br />
Tu es à l'accueil de mon site (index.php). Tu veux aller sur une autre page ?<p>

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

</html>
```

Ne vous y trompez pas : on peut créer des variables de session n'importe où dans le code. Ici je les ai créé en haut de la page, mais j'aurais pu le faire ailleurs.
La seule chose qui importe, c'est que le *session_start()* soit fait au tout début de la page.

Notez quelque chose de très important : mes liens sont tous simples. Je ne m'occupe de rien : ni de transmettre le nom, prénom, âge du visiteur, ni de transmettre l'ID de session.

Et ça, croyez-moi, c'est génial !
Quand votre site sera un peu gros et qu'il y aura plein de liens partout, vous apprécierez de savoir que PHP s'occupe tout seul de transmettre les variables !

En effet, sur chacune des pages "mapage.php", "monscript.php", "informations.php" (et n'importe quelle autre page de votre site), vous retrouverez les variables $_SESSION['prenom'], $_SESSION['nom'] et $_SESSION['age'] !

Voici par exemple le code source de la page informations.php :

Source 3.5.2 : récupérer les variables de session (informations.php)

```
<?
session_start(); // On démarre la session AVANT toute chose
?>

<html>

<head>
    <title>Titre de ma page</title>
</head>

<body>
Re-bonjour !<p>

Je me souviens de toi ! Tu t'appelles <? echo $_SESSION['prenom'] . ' ' . $_SESSION['nom']; ?>
!<br />
Et ton âge hummm... Tu as <? echo $_SESSION['age']; ?> ans, c'est ça ? :-D
</body>

</html>
```

Vous voyez ? On a juste fait un *session_start()*, puis on s'est amusés à afficher les valeurs des variables de session.
Et là, magie !
Les valeurs des variables avaient été conservées, on n'a rien eu à faire !

En résumé, vous créez des variables de session comme vous créeriez des variables normales, sauf que vous mettez le préfixe $_SESSION devant pour que PHP sache qu'il doit "retenir" ces variables sur toutes les pages.

Enfin, si vous voulez détruire la session du visiteur, vous pouvez faire un lien "Déconnexion" qui amène vers une page qui fait appel à la fonction *session_destroy()*
Quoiqu'il en soit, la session sera automatiquement détruite au bout d'un certain temps d'inactivité.

Et voilà, vous savez tout ce qu'il faut, ce n'est vraiment pas bien compliqué.

Concrètement, les sessions peuvent servir pour :

- Un script qui demande un login / mot de passe pour qu'un visiteur puisse se "connecter" (s'authentifier). Ainsi, on peut enregistrer des variables de session et se souvenir du login du visiteur sur toutes les pages du site !
- ... Ce qui permet d'ailleurs de créer une zone d'administration sécurisée sur plusieurs fichiers SANS utiliser de .htaccess. Les variables de sessions sont suffisantes pour vérifier si le mot de passe est le bon.
- Un dernier exemple : on se sert 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 !
  Si je vous dis ça, c'est en connaissance de cause, parce que j'ai déjà réalisé un site de vente en ligne. En utilisant les sessions, c'est vraiment super simple et vous avez maintenant le niveau.

Je vais m'arrêter là pour les explications sur les sessions... En effet, avec ça vous savez tout ce qu'il faut.

Toutefois, il manque quelque chose : un exemple d'utilisation des sessions (un TP quoi).
Pour ça, ne vous en faites pas : je vous ferai utiliser les sessions un peu plus tard dans ce cours, mais il faut d'abord que l'on voie d'autres choses.

Bien, on passe aux cookies maintenant !

Les Cookies

Travailler avec des cookies est quasiment aussi simple que de travailler avec des sessions.
Il faut dire que PHP fait fort grâce aux superglobales, vous allez le voir une fois de plus.

Voici le plan que nous allons suivre :

1. On va voir ce que c'est un cookie exactement... parce que je sais pas vous mais moi j'ai horreur de travailler sur des choses abstraites.
2. Ensuite, nous verrons comment **écrire un cookie**. C'est facile à faire, si on respecte un ou deux points.
3. Enfin, nous verrons comment **afficher le contenu d'un cookie**. Ca c'est super facile à faire.

Qu'Est-Ce Qu'Un Cookie ?

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

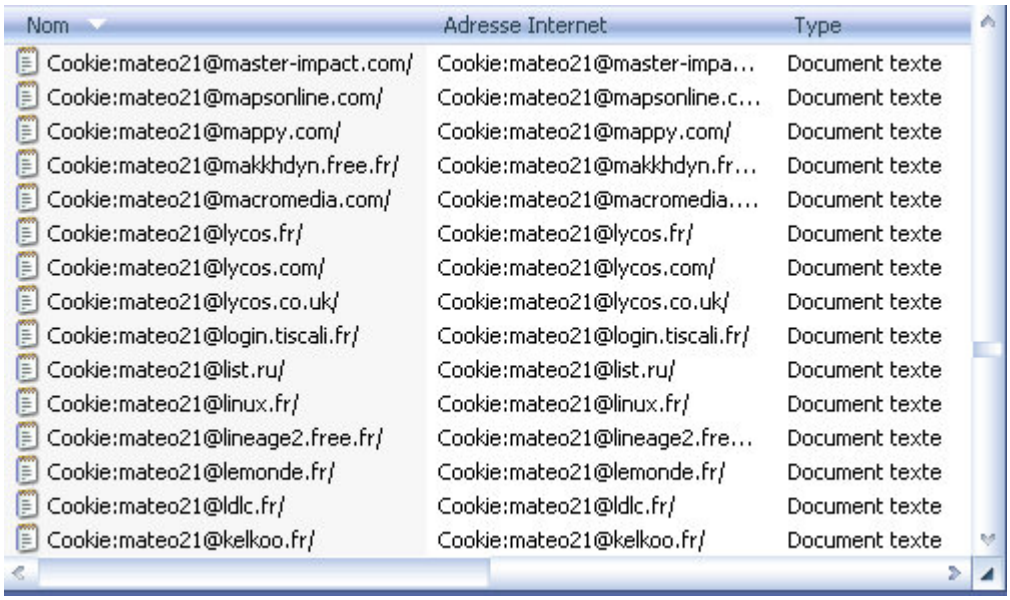
On fait souvent l'erreur de penser que les cookies sont "dangereux". Non, ce ne sont pas des virus, juste des petits fichiers textes qui permettent de retenir des informations.
Au pire, un site marchand peut retenir que vous aimez les appareils photos numériques et vous afficher uniquement des pubs pour des appareils photos, mais en aucun cas un cookie peut scanner votre disque dur ou le formater, rassurez-vous ils sont inoffensifs.

Un cookie est créé pour chaque nouveau site web qui le demande. Ainsi, si vous êtes allés voir 3 sites web, vous pouvez avoir jusqu'à 3 cookies.

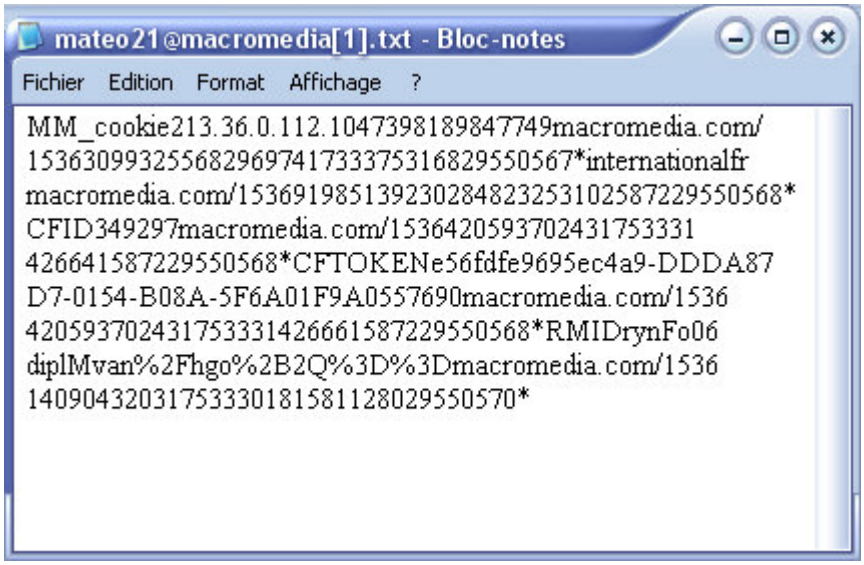
Chaque cookie peut contenir **plusieurs** informations.

 Où sont stockés les cookies sur mon disque dur ?

Ca dépend de votre navigateur. Par exemple, Internet Explorer les stocke dans le dossier "Temporary Internet Files" :



Si vous vous amusez à en ouvrir un, vous verrez probablement quelque chose d'incompréhensible :



Eh bah vous savez quoi ? On s'en fout !
On n'a pas à se préoccuper de tout ça, je vous le montrais juste pour que vous sachiez à quoi vous avez affaire. Comme d'habitude, PHP se charge de tout.

Écrire Un Cookie

Comme une variable, on écrit un cookie en donnant son nom et sa valeur.
Par exemple, le cookie "pseudo" aurait chez moi la valeur "M@teo21".

Pour écrire un cookie, on utilise la fonction PHP *setcookie* (qui signifie "Placer un cookie" en anglais).
On lui donne en général 3 paramètres, dans l'ordre suivant :

1.Le nom du cookie (ex : "pseudo")

2.La valeur du cookie (ex : "M@teo21")

3.La date d'expiration du cookie, sous forme de timestamp (ex : 1090521508)
Si vous ne savez pas ce qu'est un timestamp, c'est que vous n'avez pas lu le chapitre sur les dates.

Comme vous pouvez le voir, un cookie a une durée de vie limitée.
Il est automatiquement "supprimé" au bout d'un certain temps.

Si vous voulez supprimer le cookie dans un an, il vous faudra faire :
`time() + 365*24*3600`
Cela veut dire : timestamp actuel + nombre de secondes dans une année. Cela aura pour effet de voir votre cookie disparaître dans exactement un an.

Vous pouvez aussi utiliser la fonction *mktime* comme on l'a vu dans le chapitre sur les dates pour effacer le cookie à une date précise.

Toutefois, il y a un petit problème avec *setcookie*... Comme pour *session_start*, cette fonction ne marche QUE si vous la mettez avant tout code HTML (y compris la balise <html>)
Ca peut paraître bizarre, je le reconnais. Ce n'est pas du tout la faute à PHP, c'est comme ça que les cookies fonctionnent.

Ne placez donc JAMAIS le moindre code HTML avant d'utiliser *setcookie*.
La plupart des gens qui ont des problèmes avec *setcookie* ont fait cette erreur, donc souvenez-vous en !

Voyons maintenant comment je ferais pour inscrire 2 cookies : un qui retient mon pseudo pendant un an, et un autre qui retient le nom de mon pays :

Source 3.5.3 : écrire des cookies

```
<?
$timestamp_expire = time() + 365*24*3600; // Le cookie expirera dans un an
setcookie('pseudo', 'M@teo21', $timestamp_expire); // On écrit un cookie
setcookie('pays', 'France', $timestamp_expire); // On écrit un autre cookie...

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

<html>

<head>
    <title>Ma super page PHP</title>
</head>


<body>
etc... etc...
```

Et voilà, les cookies sont écrits !
Il m'a donc fallu faire 2 *setcookie* pour écrire ces 2 informations.

Afficher Un Cookie

Ca, c'est vraiment le plus simple.
Avant de commencer à travailler sur une page, PHP lit les cookies du client pour récupérer toutes les informations qu'il y a dedans. Ces informations sont placées dans la superglobale $_COOKIE, sous forme d'array (tableau) comme d'habitude.

Ce qui fait que, si je veux ressortir le pseudo du visiteur que j'avais inscrit dans un cookie, il suffirait d'écrire :
`$_COOKIE['pseudo']`

Ce qui nous donne un code PHP tout bête pour réafficher le pseudo du visiteur :

Source 3.5.4 : lire un cookie

```
Hé ! Je me souviens de toi !<br />
Tu t'appelles <? echo $_COOKIE['pseudo']; ?> et tu viens de <? echo $_COOKIE['pays']; ?> c'est bien ça ?
```

Comme vous le voyez encore une fois, le gros avantage c'est que les superglobales sont accessibles partout.
Vous avez besoin de savoir ce que contient le cookie "pseudo" ? Affichez donc le contenu de la superglobale $_COOKIE['pseudo'] !

A noter que si le cookie n'existe pas, la variable superglobale n'existe pas.
Plus précisément, si vous faites un *isset* sur $_COOKIE['pseudo'] comme on a appris à le faire jusqu'ici, la condition (if isset($_COOKIE['machintruc'])...) vous répondra que la variable n'existe pas... ce qui veut donc dire que le cookie n'existe pas.

Enfin, vous vous demandez peut-être comment modifier un cookie déjà existant ?
C'est là encore très simple : il faut refaire un *setcookie* en gardant le même nom de cookie. Cela "écrasera" l'ancien cookie. Par exemple, si j'habite maintenant en Chine, je ferai :
`setcookie('pays', 'Chine', $timestamp_expire);`

Notez qu'alors le temps d'expiration du cookie est remis à zéro pour un an. On aurait donc encore un an avant que le cookie disparaisse.

PHP vous offre beaucoup de puissance avec les superglobales, et vous allez voir qu'on ne va pas se faire prier pour s'en servir.

On va réutiliser dans le TP qui suit ce que vous venez d'apprendre (récupérer l'IP, manipuler des cookies etc...).
Ce chapitre était donc plutôt théorique... que diriez-vous de passer à la pratique ?

Lire et Écrire Dans Un Fichier

On ne le dira jamais assez : MySQL c'est bien !
Mais parfois, je dis bien parfois, MySQL est "trop compliqué" et pas assez rapide pour ce qu'on veut faire.

Je m'explique : supposons que vous vouliez compter le nombre de pages qui ont été vues sur votre site. Vous auriez juste besoin d'enregistrer un nombre, et de le faire augmenter à chaque fois qu'une page est chargée sur votre site.
56
57
58...

Bref, si c'était par exemple juste pour stocker UN nombre, il serait franchement débile de faire appel à la base de données.
Pourquoi ? Parce que MySQL, mine de rien, ça se révèle assez lent. Il faut s'y connecter, donner son login / mot de passe, et PHP fait l'intermédiaire entre vous et MySQL... Pas toujours pratique...

La solution ? Créer un fichier tout bête et lire et écrire dedans.
Vous allez voir que c'est très pratique, du moins tant que vous n'avez pas à stocker beaucoup de choses... Sinon MySQL redevient alors plus adapté.

Le CHMOD

Avant de commencer quoi que ce soit sur les fichiers, il faut que je vous parle de quelque chose d'un peu particulier : le CHMOD.
Derrière ce nom mystérieux se cache en fait une série de "droits", qui déterminent si oui ou non vous avez le droit de modifier un fichier.

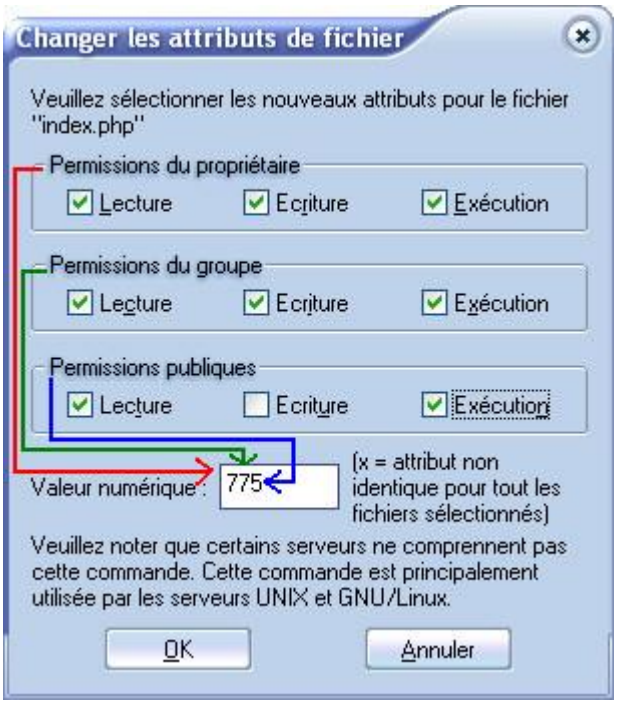
Sous Windows, vous n'en avez probablement jamais entendu parler, tout simplement parce que ça n'existe pas comme ça.
Mais le serveur de votre site lui, il est sous Linux. Et sous Linux, on utilise ce qu'on appelle le CHMOD.

Le CHMOD est un nombre à 3 chiffres que l'on attribue à un fichier (par exemple 777). Selon la valeur de ce nombre, Linux autorisera (ou non) la modification du fichier.
Le problème, c'est qu'en général Linux n'autorise pas les modifications de fichiers par un script PHP. Or, c'est justement ce qu'on veut faire. Alors, comment on va faire pour s'en sortir ? En modifiant le CHMOD pardi !

Il va falloir passer par... votre logiciel FTP ! Oui, celui-là même qui vous sert à envoyer vos pages sur le web.
En ce qui me concerne, j'utilise Filezilla (vous utilisez celui que vous voulez, la manipulation est quasiment la même).

Connectez-vous à votre serveur, et faites un clic-droit sur l'un des fichiers du serveur :

En général, vous devriez avoir un menu "CHMOD" ou "Attributs du fichier" (comme moi).
Cela devrait ouvrir une fenêtre qui ressemble à peu près à ceci :



Et c'est là que se trouve la solution à tous nos problèmes !
Bon, sans rentrer dans les détails parce qu'il n'est pas question de faire un cours de Linux ici, voilà comment ça fonctionne : il y a 3 types de personnes qui ont le droit de lire/modifier des fichiers.

- Le propriétaire : c'est l'utilisateur sous Linux qui a créé le fichier. Lui, il a en général tous les droits : lire, écrire, exécuter.
  Selon les droits qu'il possède, le premier chiffre du CHMOD change. Ici, c'est 7 : ça veut dire qu'il a tous les droits.
- Le groupe : bon ça ne nous concerne pas trop là non plus. Ce sont les droits du groupe d'utilisateurs auquel appartient le propriétaire du fichier. Cela correspond au 2° chiffre du CHMOD (ici : 7).
- Permissions publiques : ah ! Là ça devient intéressant. Les permissions publiques concernent tout le monde, c'est-à-dire même vos fichiers PHP. C'est le troisième chiffre du CHMOD (ici : 5).
  Regardez ! Il n'y a pas tous les droits ici ! En effet, nos scripts PHP n'ont pas le droit de modifier les fichiers. Allez, hop : soit vous cochez la case "Ecriture", soit vous tapez "777" pour le CHMOD. Ca voudra dire : tous les droits pour tout le monde.

Ouf ! Donc en gros, quand on met un CHMOD à 777, ça veut dire que tout le monde a le droit de modifier le fichier ! Vos visiteurs ne le peuvent pas bien entendu, c'est uniquement sur le serveur que ça se passe.

Vous pouvez aussi modifier le CHMOD d'un dossier. Cela déterminera si on a le droit de lire/écrire dans ce dossier.
Cela vous sera notamment utile si vous avez besoin d'écrire des fichiers dans un dossier en PHP.

Ouvrir Et Fermer Le Fichier

Avant de lire/écrire dans un fichier, il faut d'abord l'ouvrir.
C'est un peu comme pour la connexion à MySQL si vous voulez : on dit à PHP qu'il va falloir travailler sur ce fichier.

Commencez par créer un fichier compteur.txt (par exemple). Envoyez-le sur votre serveur avec votre logiciel FTP, et appliquez-lui un CHMOD à 777 comme on vient d'apprendre à le faire.

Maintenant, on va créer le fichier PHP qui va travailler sur compteur.txt.
Votre mission, si vous l'acceptez : compter le nombre de pages qui ont été vues sur votre site, et l'enregistrer dans ce fichier.

Fastoche !

Source 3.6.1 : ouvrir et fermer un fichier

```
<?
// 1 : on ouvre le fichier
$monfichier = fopen("compteur.txt", "r+");

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

// 3 : quand on a fini de l'utiliser, on ferme le fichier
fclose($monfichier);
?>
```

Il y a 3 étapes à respecter :

1. On ouvre le fichier avec fopen. Cette fonction renvoie une information que vous devez mettre dans une variable (ici : $monfichier). Cela nous sera utile tout à l'heure pour fermer le fichier. On indique à fopen tout d'abord le fichier qu'on veut ouvrir ("compteur.txt"), et *comment* on veut l'ouvrir (ici j'ai mis "r+"). Voici les principales possibilités qu'on a :

Mode	Explication
r	Cela ouvre le fichier en lecture seule. Cela veut dire que vous ne pourrez que lire le fichier.
r+	Cela ouvre le fichier en lecture / écriture. Vous pourrez non seulement lire le fichier, mais aussi écrire dedans (on l'utilisera assez souvent en pratique).
a	Comme "r", ça ouvre le fichier en lecture seule. Mais il y a un avantage : si le fichier n'existe pas, ça le crée automatiquement.
a+	Comme "r+", et si le fichier n'existe pas il est créé automatiquement. Attention : le répertoire doit avoir un CHMOD à 777 dans ce cas !

2. Ici, on a créé le fichier avant, donc pas besoin d'utiliser a+.
3. On fait nos opérations de lecture / écriture sur le fichier. Je n'ai encore rien mis, on va voir ça juste après.
4. Enfin, quand on a fini d'utiliser le fichier, on fait un *fclose* pour le fermer. On doit préciser quel fichier on doit fermer : mettez-y la variable $monfichier pour que PHP sache duquel il s'agit, et c'est bon.

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

Lire Et Écrire Dans Le Fichier

Maintenant que vous savez ouvrir et fermer votre fichier, on va apprendre à lire et écrire dedans.

Lire

Pour lire, on a 2 possibilités :

- Lire caractère par caractère avec la fonction *fgetc*
- Lire ligne par ligne avec *fgets*

En général, on se débrouillera pour mettre une information par ligne dans notre fichier. On utilise donc assez peu *fgetc* qui est assez lourd à utiliser (il faudrait faire une boucle pour lire caractère par caractère).

Dans notre cas, on va supposer que notre fichier ne contient qu'une ligne : le nombre de pages qui ont été vues sur le site.
Pour récupérer ce nombre, il faudra donc faire comme ceci :

Source 3.6.2 : lire dans le fichier

```
<?
// 1 : on ouvre le fichier
$monfichier = fopen("compteur.txt", "r+");

// 2 : on lit la première ligne du fichier
$ligne = fgets($monfichier);

// 3 : quand on a fini de l'utiliser, on ferme le fichier
fclose($monfichier);
?>
```

Il faut indiquer à *fgets* le fichier à lire. On lui donne notre variable $monfichier qui lui permettra de l'identifier.
fgets renvoie toute la ligne (la fonction arrête la lecture à la première Entrée). Donc, notre variable $ligne devrait contenir la première ligne du fichier.

Et si mon fichier fait 15 lignes, comment je fais pour toutes les lire ?

Il faut faire une boucle. Un premier *fgets* vous donnera la première ligne, ensuite si vous refaites *fgets* vous obtiendrez la deuxième ligne etc...

Pas très pratique hein ? Ce n'est pas pour rien qu'on a inventé la base de données.
Mais bon, comme ici on n'a à stocker qu'un seul nombre, le choix d'utiliser un fichier est justifié.

Écrire

Pour l'écriture, on n'a qu'une seule possibilité : utiliser *fputs*.
Cette fonction va écrire la ligne que vous voulez dans le fichier.

Elle s'utilise comme ceci :
```
fputs($monfichier, "Texte à écrire");
```

Toutefois, il faut savoir où on écrit le texte. En effet, le fonctionnement d'un fichier est assez bizarre :

1. Vous l'ouvrez avec *fopen*
2. Vous lisez par exemple la première ligne avec *fgets*.
3. Oui mais voilà, maintenant le "curseur" de PHP se trouve à la fin de la première ligne (vu qu'il vient de lire la première ligne).

Si vous faites un *fputs* juste après, il va écrire à la suite ! Pour éviter ça, on va utiliser la fonction *fseek* qui va replacer le curseur où on veut dans le fichier. En l'occurence, on va replacer le curseur au début du fichier en faisant :
```
fseek($monfichier, 0);
```
Notre curseur sera alors repositionné au début :

4. Ouf, notre curseur est au début du fichier, on peut faire un *fputs*. La ligne va s'écrire par-dessus l'ancienne, ce qui fait que l'ancien texte sera écrasé (remplacé par le nouveau).

Allez, pour y voir plus clair, voici le code pour réaliser notre compteur de pages vues :

Source 3.6.3 : le compteur de pages vues

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

$pages_vues = fgets($monfichier); // On lit la première ligne (nombre de pages vues)
$pages_vues++; // On augmente de 1 ce nombre de pages vues
fseek($monfichier, 0); // On remet le curseur au début du fichier
fputs($monfichier, $pages_vues); // On écrit le nouveau nombre de pages vues

fclose($monfichier);

echo 'Cette page a été vue ' . $pages_vues . ' fois !';
?>
```

Avouez que c'était pas si dur hein !
Voici la description des 4 lignes du milieu (les plus importantes) :

1. On récupère la première ligne du fichier, qui est le nombre de pages qui ont été vues pour le moment sur le site.
2. On ajoute 1 à la variable $pages_vues. Si elle valait 15, elle vaudra désormais 16.
3. On replace notre fameux "curseur" au début du fichier (parce que sinon il se trouvait à la fin de la première ligne et on aurait écrit à la suite).
4. On écrit notre nouveau nombre de pages vues dans le fichier, en écrasant l'ancien nombre.

Si vous avez oublié de mettre un CHMOD à 777 sur le fichier compteur.txt, vous aurez l'erreur suivante :
Warning: fopen(compteur.txt): failed to open stream: Permission denied
Ici, PHP essaie de vous dire qu'il n'a pas réussi à ouvrir le fichier car il n'a pas le droit d'écrire dedans. Il faut donc absoluement faire ce CHMOD si vous voulez pouvoir toucher au fichier !

Voilà, vous venez de voir comment on se sert d'un fichier : ouverture, lecture, écriture, fermeture.
Pour un gros fichier c'est vite la prise de tête, mais pour un petit fichier comme celui-ci, avouez que c'était pas long ni compliqué à faire, et en plus ça marche très bien.

Et voilà, vous savez désormais travailler avec des fichiers !
Comme vous avez pu le voir, c'est pratique et rapide du temps qu'on ne stocke pas grand chose dans le fichier. Le reste du temps, utiliser MySQL est quand même ce qu'il y a de plus pratique.

On peut faire beaucoup d'autres choses avec les fichiers, mais il serait trop long de tout vous lister ici.
Je vous invite à aller consulter la documentation PHP sur les fichiers : c'est un peu austère, mais il y a tout.
Les fonctions listées y sont assez simples à utiliser : vous verrez qu'on peut copier des fichiers, supprimer des fichiers, créer des dossiers, supprimer des dossiers etc etc...

Et n'oubliez pas qu'en cas de problème, le forum est là pour vous aider.

Note du Webmaster : bien entendu, le tutorial ne s'arrête pas là !
La rédaction d'un tel tutorial me prend énormément de temps, donc il faut patienter un peu pendant que je rédige les chapitres qui suivent. Mais ne vous inquiétez pas, je travaille dessus à fond.

ANNEXES

Dans les annexes, vous trouverez plusieurs choses intéressantes en rapport avec le PHP que je n'ai pas pu mettre dans le cours.
Ne regardez pas les annexes à la fin, mais plutôt pendant de la lecture du cours, histoire de souffler entre 2 chapitres.

Protéger Un Dossier Avec Un .htaccess

Lorsque vous réalisez votre site en PHP, vous êtes souvent amenés à créer une zone "Admin" où l'accès est limité... Et il vaut mieux, vu que les personnes qui ont accès à la zone Admin peuvent en général tout supprimer si elles le désirent.

Supposons que vous avez créé un dossier "Admin" dans lequel il y a tous les fichiers d'administration de votre site. Comment empêcher que n'importe qui accède à ces pages ?
C'est là que les fichiers .htaccess vont bien nous aider : on peut très facilement créer une protection par Login / Mot de passe qui empêche l'accès à tous les fichiers du dossier.

Il va falloir créer 2 fichiers :

- .htaccess : ce fichier contiendra l'adresse du .htpasswd et quelques autres options que vous pourrez définir.
- .htpasswd : ce fichier contiendra une liste de logins / mots de passe, pour chaque personne autorisée à accéder aux pages !

Créer Le .htaccess

La première étape est de créer sur votre disque dur un fichier appelé ".htaccess". Mais là, vous allez certainement avoir un problème (ça commence fort).
En effet, Windows n'aime pas les fichiers qui commencent par un point. Pour tous les autres systèmes d'exploitation (Mac OS, Linux) vous n'aurez aucun problème. Mais Windows lui il veut pas, allez savoir pourquoi.

On va utiliser une astuce : on va dans un premier temps créer un fichier appelé htaccess.txt, et plus tard avec notre logiciel FTP on le renommera en .htaccess (et là ça marchera !).

Commencez donc par ouvrir Bloc-Notes par exemple :

Là dedans, on va rentrer des informations qui n'ont rien à voir avec du HTML ou du PHP : ce sont des instructions pour le serveur. Elles vont expliquer au serveur que seules certaines personnes sont autorisées à accéder au dossier. Copiez-y ce code :

Source : le code du .htaccess

```
AuthName "Page d'administration protégée"
AuthType Basic
AuthUserFile "/home/sdz/www/gestion/admin/.htpasswd"
Require valid-user
```

Parmi ces 4 lignes, il y en a 2 que vous allez devoir changer :

- AuthName : c'est le texte qui invitera l'utilisateur à inscrire son login / mot de passe. Vous pouvez personnaliser ce texte comme bon vous semble.
- AuthUserFile : là c'est plus délicat, c'est le chemin absolu vers le fichier .htpasswd (que vous mettrez dans le même répertoire que le .htaccess).

Mais comment je trouve ce chemin absolu moi ?

En effet, c'est la plupart du temps délicat à trouver. Heureusement, il existe une fonction PHP qui va beaucoup nous aider : *realpath*.
Cette fonction donne le chemin absolu vers le fichier que vous indiquez. Vous allez donc faire comme ceci pour trouver le chemin absolu :

1. Créez un fichier appelé "chemin.php".
2. Mettez juste cette ligne de code dedans :
   `<? echo realpath('chemin.php'); ?>`

3. Envoyez ce fichier sur votre serveur avec votre logiciel FTP. Placez-le dans le dossier que vous voulez protéger.
4. Ouvrez votre navigateur et allez voir ce fichier PHP. Il vous donne le chemin absolu, par exemple dans mon cas :
`/home/sdz/www/gestion/admin/chemin.php`
5. Copiez ce chemin dans votre .htaccess, et remplacez le "chemin.php" par ".htpasswd", ce qui nous donne au final par exemple :
`/home/sdz/www/gestion/admin/.htpasswd`
6. Supprimez le fichier "chemin.php" de votre serveur, il ne nous sert plus à rien maintenant qu'il nous a donné le chemin absolu.

Notez que chez certains hébergeurs, la commande *realpath* est désactivée, donc vous ne pourrez peut-être pas l'utiliser. Renseignez-vous auprès de votre hébergeur pour savoir comment faire !

La ligne AuthUserFile indique donc où se trouve le fichier .htpasswd qui contient les mots de passe.

Enregistrez le fichier avec le nom "htaccess.txt" pour le moment, on le renommera en ".htaccess" plus tard.
Voilà, on a fini de créer le .htaccess, on peut maintenant passer au .htpasswd.

Créer Le .htpasswd

Créez maintenant un nouveau fichier avec Bloc-Notes.

Le .htpasswd contient la liste des personnes autorisées à accéder aux pages du dossier.
On met une personne par ligne, sous cette forme :
`login:mot_de_passe_crypté`

Au final, votre fichier .htpasswd devrait ressembler à ceci :

Source : un exemple de .htpasswd

```
mateo21:$1$MEqT//cb$hAVid.qmmSGFW/wDlIfQ81
darkeden:$1$/lgP8dYa$sQNXcCP47KhP1sneRIZoO0
IAN:$1$lT7nqnsg$cVtoPfe0IgrjES7Ushmoy.
Leon:$1$h4oVHp3O$X7Ejpn.uuOhJRkT3qnw3i0
```

Dans cet exemple, il y a 4 personnes autorisées à accéder au dossier : ce sont mateo21, darkeden, IAN, et Leon.

S'il n'y a qu'une personne autorisée à accéder au dossier, vous n'avez qu'à mettre qu'une ligne. Mais si vous êtes plusieurs admins, il est très pratique de pouvoir créer plusieurs "comptes" avec login / mot de passe.

Hé ho ?! Comment je les crypte les mots de passe moi ?

Bonne question !
Encore une fois, il y a une super fonction PHP qui va nous tirer d'affaire : *crypt*. Vous lui donnez un mot de passe et, ne cherchez pas à savoir comment, ça vous le crypte.

Par exemple, si mon mot de passe est "kangourou", voici le code PHP que je devrai écrire pour l'obtenir en version cryptée :

```
<? echo crypt('kangourou'); ?>
```

Crypter ses mots de passe est très utile : en effet, si quelqu'un vient un jour à lire votre fichier .htpasswd (quelqu'un qui utilise le même PC que vous par exemple), il ne verra que le mot de passe crypté.
Et là, aucun risque qu'il ne retrouve votre mot de passe : ce cryptage est indéchiffrable. C'est donc très pratique.

Bon, en pourrait en théorie s'arrêter là pour le .htpasswd, mais mon âme de codeur PHP me commande de créer un petit script qui va bien vous être utile. Si vous avez lu le cours PHP jusqu'à la fin de la partie I, vous devriez être capables de comprendre ce script :

Code : une page pratique pour le .htpasswd

```
<?
if (isset($_POST['login']) AND isset($_POST['pass']))
{
    $login = $_POST['login'];
    $pass_crypte = crypt($_POST['pass']); // On crypte le mot de passe

    echo 'Ligne à copier dans le .htpasswd :<br />' . $login . ':' . $pass_crypte;
}

else // On n'a pas encore rempli le formulaire
{
?>

Entrez votre login et votre mot de passe pour le crypter.

<form method="post">

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

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

</form>

<?
}
?>
```

Il y a 2 parties dans ce code, dont la forme est similaire aux TP "Page protégée par mot de passe", "Mini-Chat", etc...

1. SI les variables $_POST['login'] et $_POST['pass'] existent, alors c'est qu'on vient de valider le formulaire.
   On crypte le mot de passe qu'on a rentré, et on affiche $login:$pass_crypte pour que vous n'ayez plus qu'à copier la ligne dans le .htpasswd.
2. SINON, si les variables $_POST['login'] et $_POST['pass'] n'existent pas, donc on affiche le formulaire pour demander d'entrer un login et un mot de passe.
   Le formulaire recharge la même page, car il n'y a pas d'attribut *action* dans la balise <form> comme on l'a vu dans le chapitre sur les formulaires. Lors du rechargement de la page, les variables $_POST['login'] et $_POST['pass'] existeront puisque vous venez d'entrer le login et le mot de passe. Le mot de passe sera alors crypté !

Je vous conseille de créer cette page quelque part sur votre disque dur (ou sur votre serveur peu importe), pour que vous puissiez crypter rapidement vos mots de passe pour le .htpasswd.
Si vous avez la flême de le créer, pas de souci, vous n'avez qu'à venir sur cette page et cliquer sur le bouton "Essayer !"

Si vous êtes hébergés chez Free, vous ne DEVEZ PAS crypter vos mots de passe. En effet, Free demande à ce que les mots de passe ne soient pas cryptés (ce qui est complètement nul pour la sécurité, mais bon...). Vous devrez donc taper le mot de passe directement. Par exemple :
`mateo21:superpass`

Envoyer Les Fichiers Sur Le Serveur

Vous avez maintenant 2 fichiers sur votre disque dur : htaccess.txt et htpasswd.txt.

Lancez votre logiciel FTP.
Transférez les fichiers htaccess.txt et htpasswd.txt dans le dossier que vous voulez protéger par mot de passe.
Vous devriez voir ceci dans votre logiciel FTP :

Site Distant : /www/admin/

Nom	Taille	Date	Heure	Permissions
..				
admin_commentaires.php	681	08/04/2004	17:08	-rw-r--r--
admin_livreor.php	1161	08/04/2004	17:07	-rw-r--r--
admin_news.php	681	08/04/2004	17:07	-rw-r--r--
admin_sondage.php	681	08/04/2004	17:08	-rw-r--r--
htaccess.txt	72	08/04/2004	17:08	-rw-r--r--
htpasswd.txt	72	08/04/2004	17:09	-rw-r--r--

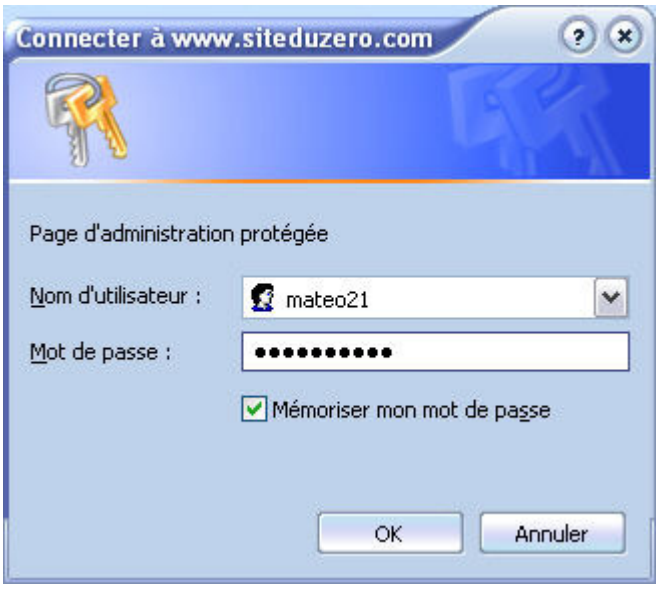
Maintenant que ces fichiers sont sur le serveur, renommez-les (Bouton droit / "Renommer" ça doit marcher). Appelez-les respectivement ".htaccess" et ".htpasswd".
Vous devriez voir ceci au final :

Site Distant : /www/admin/

Nom	Taille	Date	Heure	Permissions
..				
.htaccess	72	08/04/2004	17:09	-rw-r--r--
.htpasswd	72	08/04/2004	17:08	-rw-r--r--
admin_commentaires.php	681	08/04/2004	17:08	-rw-r--r--
admin_livreor.php	1161	08/04/2004	17:07	-rw-r--r--
admin_news.php	681	08/04/2004	17:07	-rw-r--r--
admin_sondage.php	681	08/04/2004	17:08	-rw-r--r--

Voilà, désormais le dossier est protégé !

Vous n'avez pas besoin de faire un lien vers le .htaccess pour demander de se logger : ça se fera tout seul si vous essayez d'accéder à l'une des pages du dossier (admin_commentaires.php, admin_livreor.php...)
Par exemple, si j'essaie d'accéder à la page "http://www.siteduzero.com/admin/admin_livreor.php", cette fenêtre apparaît et me demande de m'authentifier :



Si vous rentrez le bon login avec le bon mot de passe, vous serez alors autorisé à accéder aux pages !