



Concevez votre site web avec PHP et MySQL

Par M@teo21



Sommaire

Sommaire	1
Informations sur le tutoriel	5
Concevez votre site web avec PHP et MySQL	7
Informations sur le tutoriel	7
Partie 1 : Les bases de PHP	8
Introduction à PHP	8
Les sites statiques et dynamiques	8
Comment fonctionne un site web ?	9
Cas d'un site statique	10
Cas d'un site dynamique	11
Les langages du Web	11
Pour un site statique : XHTML et CSS	12
Pour un site dynamique : ajoutez PHP et MySQL	13
PHP génère du XHTML	13
Et la concurrence ?	14
Les concurrents de PHP	15
Les concurrents de MySQL	15
Plusieurs combinaisons sont possibles	16
Préparer son ordinateur	16
De quels programmes a-t-on besoin ?	17
Avec un site statique	17
Avec un site dynamique	17
Sous Windows : WAMP	17
Sous Mac OS X : MAMP	22
Sous Linux : XAMPP	26
Utiliser un bon éditeur de fichiers	30
Sous Windows	31
Sous Mac OS X	35
Sous Linux	36
Premiers pas avec PHP	37
Les balises PHP	37
La forme d'une balise PHP	38
Insérer une balise PHP au milieu du code XHTML	39
Afficher du texte	41
L'instruction echo	41
Enregistrer une page PHP	42
Tester la page PHP	43
Comment PHP génère du code XHTML	44
Les commentaires	44
Les commentaires monolignes	45
Les commentaires multilignes	45
Inclure des portions de page	46
Le principe	47
Le problème	47
La solution	49
La pratique	50
Les variables	52
Qu'est-ce qu'une variable ?	52
Un nom et une valeur	53
Les différents types de variables	53
Affecter une valeur à une variable	55
Premières manipulations de variables	55
Utiliser les types de données	56
Afficher et concaténer des variables	58
Afficher le contenu d'une variable	58
La concaténation	58
Faire des calculs simples	61
Les opérations de base : addition, soustraction	61
Le modulo	62
Et les autres opérations ?	62
Les conditions	62
La structure de base : If... Else	63
Les symboles à connaître	63
La structure If... Else	63
Le cas des booléens	65
Des conditions multiples	66

L'astuce bonus	67
Une alternative pratique : Switch	69
Les ternaires : des conditions condensées	72
Les boucles	73
Une boucle simple : While	74
Une boucle plus complexe : For	77
Les fonctions	78
Qu'est-ce qu'une fonction ?	78
Dialogue avec une fonction	79
Les fonctions en PHP	80
Les fonctions prêtes à l'emploi de PHP	81
Traitement des chaînes de caractères	82
Récupérer la date	84
Créer ses propres fonctions	86
1er exemple : dis bonjour au Monsieur	86
2ème exemple : calculer le volume d'un cône	87
Les tableaux	89
Les deux types de tableaux	89
Les tableaux numérotés	90
Les tableaux associatifs	92
Parcourir un tableau	93
La boucle for	94
La boucle foreach	94
Afficher rapidement un array avec print_r	96
Rechercher dans un tableau	97
Vérifier si une clé existe dans l'array : array_key_exists	98
Vérifier si une valeur existe dans l'array : in_array	98
Récupérer la clé d'une valeur dans l'array : array_search	99
Partie 2 : Transmettre des données de page en page	100
Transmettre des données avec l'URL	101
Envoyer des paramètres dans l'URL	101
Former une URL pour envoyer des paramètres	102
Créer un lien avec des paramètres	102
Récupérer les paramètres en PHP	103
Ne faites jamais confiance aux données reçues !	106
Tous les visiteurs peuvent trafiquer les URL	106
Tester la présence d'un paramètre	106
Contrôler la valeur des paramètres	108
Transmettre des données avec les formulaires	111
Créer la base du formulaire	111
La méthode	112
La cible	112
Les éléments du formulaire	114
Les petites zones de texte	114
Les grandes zones de texte	115
La liste déroulante	116
Les cases à cocher	117
Les boutons d'option	117
Les champs cachés	118
Ne faites jamais confiance aux données reçues : la faille XSS	119
Pourquoi les formulaires ne sont pas sûrs	119
La faille XSS : attention au code HTML que vous recevez !	121
L'envoi de fichiers	123
Le formulaire d'envoi de fichier	124
Le traitement de l'envoi en PHP	125
TP : page protégée par mot de passe	129
Instructions pour réaliser le TP	130
Les prérequis	130
Votre objectif	130
Comment procéder ?	130
A vous de jouer !	131
Correction	131
Aller plus loin	133
Variables superglobales, sessions et cookies	135
Les variables superglobales	136
Les sessions	136
Fonctionnement des sessions	137
Exemple d'utilisation des sessions	137
L'utilité des sessions en pratique	139
Les cookies	141
Qu'est-ce qu'un cookie ?	141
Ecrire un cookie	142
Afficher un cookie	144

Modifier un cookie existant	144
Lire et écrire dans un fichier	145
Autoriser l'écriture de fichiers (chmod)	145
Ouvrir et fermer le fichier	147
Lire et écrire dans le fichier	148
Lire	149
Ecrire	149
Partie 3 : Stocker des informations dans une base de données	152
Présentation des bases de données	152
Le langage SQL et les bases de données	152
Les SGBD s'occupent du stockage	153
Vous donnez les ordres au SGBD en langage SQL	153
PHP fait la jonction entre vous et MySQL	154
Structure d'une base de données	155
Mais où sont enregistrées les données ?	156
phpMyAdmin	157
Créer une table	158
Les types de champs MySQL	160
Les clés primaires	161
Modifier une table	163
Autres opérations	166
SQL	166
Importer	167
Exporter	168
Opérations	170
Vider	170
Supprimer	170
Lire des données	171
Se connecter à la base de données en PHP	172
Comment se connecte-t-on à la base de données en PHP ?	172
Activer PDO	172
Se connecter à MySQL avec PDO	173
Tester la présence d'erreurs	175
Récupérer les données	175
Faire une requête	176
Votre première requête SQL	177
Afficher le résultat d'une requête	178
Afficher seulement le contenu de quelques champs	180
Les critères de sélection	181
WHERE	181
ORDER BY	182
LIMIT	183
Construire des requêtes en fonction de variables	186
La mauvaise idée : concaténer une variable dans une requête	186
La solution : les requêtes préparées	186
Ecrire des données	189
INSERT : ajouter des données	189
La requête INSERT INTO permet d'ajouter une entrée	190
Application en PHP	191
Insertion de données variables grâce à une requête préparée	192
UPDATE : modifier des données	192
La requête UPDATE permet de modifier une entrée	193
Application en PHP	194
Avec une requête préparée	194
DELETE : supprimer des données	195
TP : un Mini-Chat	195
Instructions pour réaliser le TP	195
Prérequis	196
Objectifs	196
Structure de la table MySQL	197
Structure des pages PHP	198
Rappels sur les consignes de sécurité	198
A vous de jouer !	199
Correction	200
minichat.php : formulaire et liste des derniers messages	200
minichat_post.php : enregistrement et redirection	201
Aller plus loin	203
Les fonctions SQL	203
Les fonctions scalaires	204
Utiliser une fonction scalaire SQL	205
Présentation de quelques fonctions scalaires utiles	207
Les fonctions d'agrégat	208
Utiliser une fonction d'agrégat SQL	209

Présentation de quelques fonctions d'agrégat utiles	210
GROUP BY et HAVING : le groupement de données	214
GROUP BY : grouper des données	214
HAVING : filtrer les données regroupées	215
Les dates en SQL	216
Les champs de type date	217
Les différents types de dates	217
Utilisation des champs de date en SQL	218
Les fonctions de gestion des dates	219
NOW() : obtenir la date et l'heure actuelles	220
DAY(), MONTH(), YEAR() : extraire le jour, le mois ou l'année	220
HOUR(), MINUTE(), SECOND() : extraire les heures, minutes, secondes	220
DATE_FORMAT : formater une date	220
DATE_ADD et DATE_SUB : ajouter ou soustraire des dates	221
TP : un blog avec des commentaires	222
Instructions pour réaliser le TP	222
Prérequis	223
Objectifs	223
Structure des tables MySQL	225
Structure des pages PHP	226
A vous de jouer !	226
Correction	228
index.php : la liste des derniers billets	228
commentaires.php : affichage d'un billet et de ses commentaires	229
Aller plus loin	231
Un formulaire d'ajout de commentaire	232
Utiliser les includes	232
Vérifier si le billet existe sur la page des commentaires	232
Paginer les billets et commentaires	232
Réaliser une interface d'administration du blog	233
Les jointures entre tables	234
Modélisation d'une relation	234
Qu'est-ce qu'une jointure ?	236
Les jointures internes	239
Jointure interne avec WHERE (ancienne syntaxe)	239
Jointure interne avec JOIN (nouvelle syntaxe)	241
Les jointures externes	242
LEFT JOIN : récupérer toute la table de gauche	243
RIGHT JOIN : récupérer toute la table de droite	243
Partie 4 : Utilisation avancée de PHP	244
Créer des images en PHP	245
Activer la librairie GD	245
Les bases de la création d'image	247
Le header	247
Créer l'image de base	247
Quand on a terminé : on affiche l'image	249
Texte et couleur	252
Manipuler les couleurs	252
Ecrire du texte	254
Dessiner une forme	255
Des fonctions encore plus puissantes	258
Rendre une image transparente	259
Mélanger deux images	260
Redimensionner une image	262
Les expressions régulières (Partie 1/2)	265
Où utiliser une Regex ?	265
POSIX ou PCRE ?	266
Les fonctions qui nous intéressent	266
preg_match	266
Des recherches simples	267
Et tu casses, tu casses, tu casses..	269
Le symbole OU	269
Début et fin de chaîne	270
Les classes de caractères	270
Des classes simples	271
Les intervalles de classe	271
Et pour dire que j'en veux pas ?	272
Les quantificateurs	273
Les symboles les plus courants	274
Etre plus précis grâce aux accolades	275
Les expressions régulières (Partie 2/2)	276
Une histoire de métacaractères	276
Alerte mon Général ! Les métacaractères s'échappent !	277
Le clas cas des classes	278

Les classes abrégées	278
Construire une Regex complète	279
Un numéro de téléphone	280
Une adresse e-mail	282
Des Regex... avec MySQL !	283
Capture et remplacement	284
Les parenthèses capturantes	285
Créez votre bbCode	286
La programmation orientée objet	290
Qu'est-ce qu'un objet ?	290
Ils sont beaux, ils sont frais mes objets	291
Imaginez... un objet	291
Vous avez déjà utilisé des objets !	293
Créer une classe	295
Les variables membres	296
Les fonctions membres	297
Créer un objet à partir de la classe	300
Constructeur, destructeur et autres fonctions spéciales	303
Le constructeur : __construct	303
Le destructeur : __destruct	304
Les autres fonctions magiques	304
L'héritage	306
Comment reconnaître un héritage ?	306
Réaliser un héritage en PHP	306
Les droits d'accès et l'encapsulation	309
Les droits d'accès	309
L'encapsulation	310
Partie 5 : Annexes	312
Envoyez votre site sur le web	312
Le nom de domaine	312
Réserver un nom de domaine	313
L'hébergeur	315
Le rôle de l'hébergeur	316
Trouver un hébergeur	316
Commander un hébergement pour votre site web	318
Utiliser un client FTP	320
Installer un client FTP	321
Configurer le client FTP	322
Transférer les fichiers	323
Accéder à la base de données de l'hébergeur	324
Codez proprement	327
Des noms clairs	329
Des noms de variables peu clairs	330
Des noms de variables beaucoup plus clairs	330
Indentez votre code	331
Avec un code non indenté	331
Avec un code indenté	331
Un code correctement commenté	332
Utilisez la documentation PHP !	334
Accéder à la doc	334
Liste des fonctions classées par thème	335
Accès direct à une fonction	336
Présentation d'une fonction	337
Apprendre à lire un mode d'emploi	339
Un autre exemple : date	340
Lisez les exemples !	340
Au secours ! Mon script plante !	341
Les erreurs les plus courantes	342
Parse error	342
Undefined function	343
Wrong parameter count	343
Traiter les erreurs SQL	345
Repérer l'erreur SQL en PHP	345
Allez ! Crache le morceau !	345
Quelques erreurs plus rares	346
Headers already sent by...	347
L'image contient des erreurs	348
Maximum execution time exceeded	348
Protéger un dossier avec un .htaccess	349
Créer le .htaccess	349
Créer le .htpasswd	351
Envoyer les fichiers sur le serveur	354
Memento des expressions régulières	354

Structure d'une Regex	354
Classes de caractères	355
Quantificateurs	357
Métacaractères	357
Classes abrégées	358
Capture et remplacement	360
Options	360

Concevez votre site web avec PHP et MySQL

Blogs, réseaux sociaux, pages d'accueil personnalisables... Depuis quelques années, les sites web ont gagné en fonctionnalités et sont devenus dans le même temps de plus en plus complexes.

Que le temps de la "page web perso" est loin ! Il y a une époque où l'on pouvait se contenter de créer un site basique. Un peu de texte, quelques images : hop là, notre site perso était prêt. 😊

Aujourd'hui, c'est différent : *il faut que ça bouge* ! On s'attend à ce qu'un site soit régulièrement mis à jour : on veut voir des actualités sur la page d'accueil, on veut pouvoir les commenter, discuter sur des forums, bref, participer à la vie du site.

Le **langage PHP** a justement été conçu pour créer des sites "vivants" (on parle de sites dynamiques). Et si vous voulez apprendre à créer vous aussi des sites web dynamiques, c'est votre jour de chance : vous êtes sur un cours pour vrais **débutants** en PHP !

L'essentiel, 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 ! 😊

Informations sur le tutoriel

Auteur : M@teo21

Difficulté :

Temps d'étude estimé : 2 mois

Licence :



Pour utiliser PHP, il faut connaître au préalable les langages XHTML et CSS.

Comment ça, ces langages ne vous disent rien ? Vous ne savez même pas ce qu'est un "langage" ? Il faut donc que vous lisiez d'abord mon [cours de création de site web en XHTML et CSS](#) avant de revenir ici !



Ce cours vous plaît ?

Si vous avez aimé ce cours, vous pouvez retrouver le livre "[Concevez votre site web avec PHP et MySQL](#)" du même auteur, en vente [sur le Site du Zéro](#), [en librairie](#) et [dans les boutiques en ligne](#). Vous y trouverez ce cours adapté au format papier avec une série de chapitres inédits.

[Plus d'informations](#)

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

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

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

Ce premier chapitre est là pour répondre à toutes ces questions et vous montrer que vous êtes capables d'apprendre à créer des sites web dynamiques. Tous les lecteurs seront à la fin rassurés de savoir qu'ils commencent au même niveau !

Les sites statiques et dynamiques

On considère qu'il existe 2 types de sites web : les sites **statiques** et les sites **dynamiques**.



- **Les sites statiques** : ce sont des sites réalisés uniquement à l'aide des langages (X)HTML et CSS. Ils fonctionnent très bien mais leur contenu ne peut pas être mis à jour automatiquement : il faut que le propriétaire du site (le webmaster) modifie le code source pour y ajouter des nouveautés. Ce n'est pas très pratique quand on doit mettre à jour son site plusieurs fois dans la même journée ! Les sites statiques sont donc bien adaptés pour réaliser des sites "vitrine", pour présenter par exemple son entreprise, mais sans aller plus loin. Ce type de site se fait de plus en plus rare aujourd'hui, car dès que l'on rajoute un élément d'interaction (comme un formulaire de contact), on ne parle plus de site statique mais de site dynamique.
- **Les sites dynamiques** : plus complexes, ils utilisent d'autres langages en plus de (X)HTML et CSS, tels que PHP et MySQL. Le contenu de ces sites web est dit "dynamique" parce qu'il peut changer sans l'intervention du webmaster ! La plupart des sites web que vous visitez aujourd'hui, y compris le Site du Zéro, sont des sites dynamiques. Le seul prérequis pour apprendre à créer ce type de site est de déjà savoir réaliser des sites statiques en XHTML et CSS (vous pouvez lire [mon cours sur XHTML et CSS](#) pour vous mettre à niveau).

L'objectif de ce cours est de vous rendre capables de réaliser des sites web dynamiques entièrement par vous-mêmes, pas à pas. En effet, ceux-ci peuvent proposer des fonctionnalités bien plus excitantes que les sites statiques. Voici quelques éléments que vous serez en mesure de réaliser :

- **Un espace membres** : vos visiteurs peuvent s'inscrire à votre site et avoir accès à des sections qui leur sont réservées.
- **Un forum** : il est courant aujourd'hui de voir les sites web proposer un forum de discussion pour s'entraider ou simplement pour passer le temps.
- **Un compteur de visiteurs** : vous pouvez facilement compter le nombre de visiteurs qui se sont connectés dans la journée sur votre site, ou même connaître le nombre de visiteurs en train de naviguer dessus !
- **Des actualités** : vous pouvez automatiser l'écriture d'actualités, en offrant à vos visiteurs la possibilité d'en rédiger, de les commenter, etc.
- **Une newsletter** : vous pouvez envoyer un e-mail à tous vos membres régulièrement pour leur présenter les nouveautés et les inciter ainsi à revenir sur votre site.

Bien entendu, ce ne sont là que des exemples. Il est possible d'aller encore plus loin, tout dépend de vos besoins. Sachez par exemple que la quasi-totalité des sites de jeux en ligne sont dynamiques. On retrouve notamment des sites d'élevage virtuel d'animaux, des jeux de conquête spatiale, etc.

Mais... Ne nous emportons pas. Avant de pouvoir en arriver là, vous avez de la lecture et bien des choses à apprendre ! Commençons par la base : savez-vous ce qui se passe lorsque vous consultez une page web ? 😊

Comment fonctionne un site web ?

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

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

- **Les clients** : ce sont les ordinateurs des internautes comme vous. Votre ordinateur fait donc partie de la catégorie des clients. Chaque client représente un visiteur d'un site web. Dans les schémas qui vont suivre, l'ordinateur d'un client sera représenté par cette image :



- **Les serveurs** : ce sont des ordinateurs puissants qui stockent et délivrent des sites web aux internautes, c'est-à-dire aux clients. La plupart des internautes n'ont jamais vu un serveur de leur vie. Pourtant, les serveurs sont indispensables au bon fonctionnement du web. L'image ci-dessous représentera un serveur dans les schémas suivants :



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

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

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

Cas d'un site statique

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

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



La communication est donc plutôt basique :

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

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

Cas d'un site dynamique

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

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



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

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

Les langages du Web

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



Certains programmes, appelés WYSIWYG (What You See Is What You Get), permettent d'aider les plus novices à créer un site web statique sans connaître les langages informatiques qui se cachent derrière... Mais pour réaliser un site dynamique comme nous le souhaitons, nous devrons absolument mettre les mains dans le cambouis.

Pour un site statique : XHTML et CSS

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

- **XHTML** : c'est le langage à la base des sites web. Il ressemble beaucoup au HTML mais impose quelques règles un peu plus strictes. Dans la mesure du possible je recommande d'utiliser XHTML plutôt que HTML car cela vous force à soigner le code source de vos sites.

XHTML est un langage simple à apprendre qui fonctionne à partir de balises. Voici un exemple de code XHTML :

Code : HTML

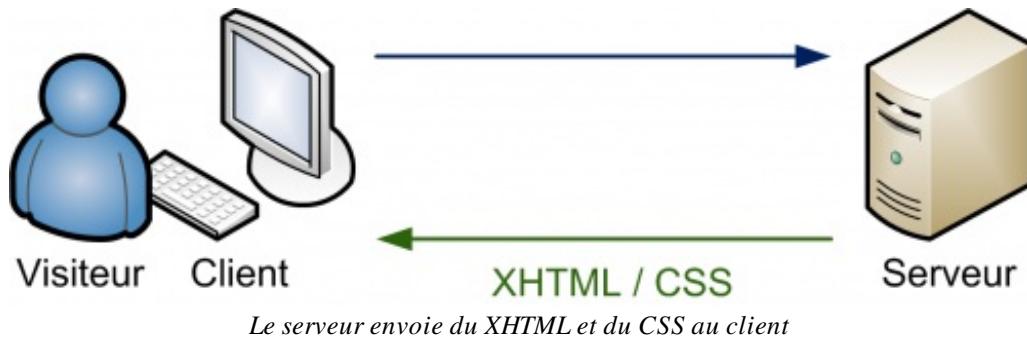
```
<p>Bonjour, je suis un <em>paragraphe</em> de texte !</p>
```

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

Code : CSS

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

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



Le problème, c'est que lorsqu'on connaît seulement XHTML et CSS, on ne peut produire que des sites statiques... et pas des

sites dynamiques ! Pour ces derniers, il est nécessaire de manipuler d'autres langages en plus de XHTML et CSS.

La question qu'il faut vous poser est donc : connaissez-vous XHTML et CSS ?

Si oui, c'est parfait, vous pouvez continuer car nous en aurons besoin par la suite. Si la réponse est non, pas de panique. Ces langages ne sont pas bien difficiles, ils sont à la portée de tous. Vous pouvez les apprendre en lisant mon [cours sur XHTML et CSS](#).

Sachez qu'apprendre ces langages n'est l'affaire que de quelques petites semaines, voire même moins si vous avez suffisamment de temps libre. 😊

Pour un site dynamique : ajoutez PHP et MySQL

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

C'est justement tout l'objet de ce cours : vous allez apprendre à manipuler PHP et MySQL pour réaliser un site web dynamique.

- **PHP** : c'est un langage que seuls les serveurs comprennent et qui permet de rendre votre site dynamique. C'est PHP qui "génère" la page web comme on l'a vu sur un des schémas précédents.
Ce sera le premier langage que nous découvrirons dans ce cours. Il peut fonctionner seul, mais il ne prend vraiment de l'intérêt que s'il est combiné à un outil tel que MySQL. Voici un code PHP :

Code : PHP

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

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

Code : SQL

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

PHP et MySQL sont ce qu'on appelle des logiciels libres. Entre autres choses, cela vous donne des garanties de pérennité : tout le monde peut contribuer à leur développement, vous ne risquez donc pas de voir tous les webmasters se désintéresser du PHP et de MySQL du jour au lendemain, et ça c'est très important !

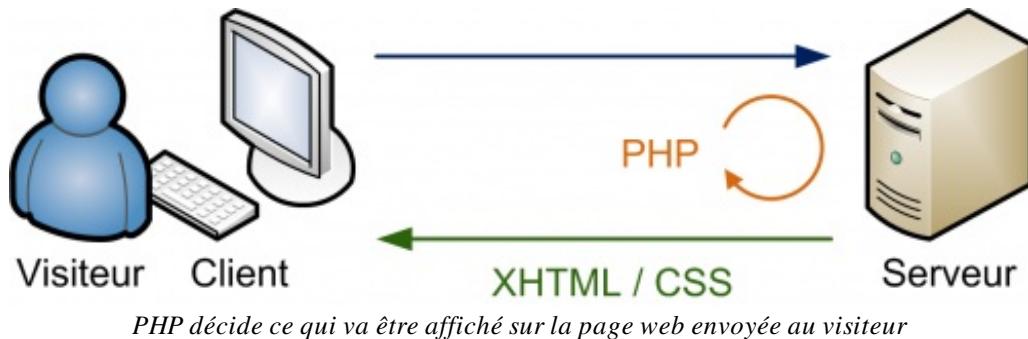
D'autre part, PHP et MySQL sont disponibles gratuitement. Cela signifie une chose essentielle : vous n'aurez pas à débourser un centime pour construire votre site web ! 😊

PHP peut fonctionner seul et suffit à créer un site dynamique, mais les choses deviennent réellement intéressantes lorsqu'on le combine à un SGBD tel que MySQL. Cependant pour simplifier, oublions pour le moment MySQL et concentrons-nous sur PHP. 😊

PHP génère du XHTML

Les clients sont incapables de comprendre le code PHP, ils ne connaissent que le XHTML et le CSS. Seul le serveur est capable de lire du PHP.

Le rôle de PHP est justement de générer du code XHTML (on peut aussi générer du CSS, mais c'est plus rare), code qui est ensuite envoyé au client de la même manière qu'un site statique :



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

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

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

Et la concurrence ?

XHTML et CSS n'ont pas de concurrents car ce sont des standards. Tout le monde est censé les connaître et les utiliser sur tous les sites web.

En revanche, pour ce qui est des sites dynamiques, PHP et MySQL sont loin d'être les seuls sur le coup. Je ne peux pas vous faire une liste complète de leurs concurrents, ce serait bien trop long (et ennuyeux !). Cependant, pour votre culture générale il faut au moins connaître quelques autres grands noms.

Tout d'abord, si on a souvent tendance à combiner PHP et MySQL pour réaliser de puissants sites dynamiques, il ne faut pas mélanger les deux. Le premier a des concurrents différents du second.

Les concurrents de PHP

Parmi les concurrents de PHP, on peut citer :

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



Je ne peux pas présenter ici tous les concurrents, mais ceci devrait déjà vous donner une bonne idée. Pour information, il est aussi possible d'utiliser par exemple le langage C ou le C++, bien que ce soit plus complexe et pas forcément toujours très adapté (en clair, je ne le recommande pas du tout 😊).



Lequel choisir dans le lot ? Lequel est le *meilleur* ?

Étant donné l'objet de ce cours, vous vous attendez à ce que je vous réponde instantanément "PHP !". Mais non. En fait, tout dépend de vos connaissances en programmation. Si vous avez déjà manipulé le Java, vous serez plus rapidement à l'aise avec les JSP. Si vous connaissez Python, Django semble tout indiqué.

Quant à PHP, il se démarque de ses concurrents par une importante communauté qui peut vous aider rapidement sur internet si vous avez des problèmes. C'est un langage facile à utiliser, idéal pour les débutants comme pour les professionnels : Wikipédia et Facebook sont des exemples de sites célèbres et très fréquentés qui fonctionnent grâce à PHP.

Bref, il n'y a pas de meilleur choix. Je vous recommande le langage pour lequel vous serez le plus certain d'avoir quelqu'un pour vous aider. Le PHP en ce sens est souvent un très bon choix.

Les concurrents de MySQL

En ce qui concerne les bases de données, le choix est là encore très vaste.

Cependant, alors que PHP et ses concurrents sont la plupart du temps libres et gratuits, ce n'est pas le cas de la plupart des SGBD.

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

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



Là encore, cette liste est loin d'être exhaustive mais vous présente au moins quelques grands noms.

Pour information, MySQL reste de loin le SGBD libre et gratuit le plus utilisé. Parmi les solutions professionnelles payantes, Oracle est le plus avancé et le plus répandu mais son utilisation est surtout réservée aux grosses entreprises.

En fin de compte, si vos moyens sont limités, vous n'avez pas beaucoup de choix pour le SGBD. MySQL est le plus indiqué car il est libre, gratuit, performant et utilisé par de nombreuses personnes qui sont susceptibles de vous aider.

Plusieurs combinaisons sont possibles

Comme vous avez pu le constater, vous avez le choix entre de nombreux outils pour réaliser un site dynamique. La plupart d'entre eux sont gratuits.

Sachez que vous pouvez a priori combiner ces outils comme bon vous semble. Par exemple, on peut très bien utiliser PHP avec une autre base de données que MySQL, telle que Oracle ou PostgreSQL. De même, MySQL peut être utilisé avec n'importe quel autre langage : Java, Python, Ruby, etc.

Cependant, la combinaison "**PHP + MySQL**" est probablement la plus courante. Ce n'est pas par hasard si ce cours traite de ces deux outils qui ont fait leurs preuves. 😊

Vous devriez maintenant avoir une bonne idée de ce que permettent de faire PHP et MySQL. Si vous retenez que PHP génère du XHTML personnalisé pour chaque visiteur et que MySQL sert à stocker les données, vous savez déjà le principal.

Dans la première partie de ce cours nous allons découvrir PHP en douceur. Nous mettons donc de côté MySQL et ne ferons pas de stockage de données dans un premier temps. Lorsque vous aurez acquis un certain niveau en PHP, je vous reparlerai de MySQL et nous pourrons alors commencer à réaliser des fonctionnalités très intéressantes pour votre futur site. 😊



Préparer son ordinateur

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

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

De quels programmes a-t-on besoin ?

Selon que l'on crée un site statique ou un site dynamique, on a besoin de logiciels différents. En fait, faire un site dynamique nécessite hélas pour nous un peu plus de logiciels ! 😊

Avec un site statique

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

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



Mozilla Firefox

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

Avec un site dynamique

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

Ces programmes dont nous allons avoir besoin, quels sont-ils ?

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



Logo d'Apache

Tous ces éléments qui vont nous aider à créer notre site dynamique sont libres et gratuits. Certes, il en existe d'autres (parfois payants), mais la combinaison Apache + PHP + MySQL est la plus courante sur les serveurs web, à tel point qu'on a créé des "packs" tous prêts qui contiennent tous ces éléments. Il est possible de les installer un à un mais cela prend plus de temps et vous n'allez rien y gagner (sauf si vous êtes administrateur de serveur, ce qui ne devrait pas être votre cas 😊).

Nous allons voir comment installer le "pack" qui convient en fonction de votre système d'exploitation dans la suite de ce chapitre.

Sous Windows : WAMP

Il existe plusieurs paquetages tous prêts pour Windows. Je vous propose d'utiliser WAMP Server qui a l'avantage d'être régulièrement mis à jour et disponible en français.

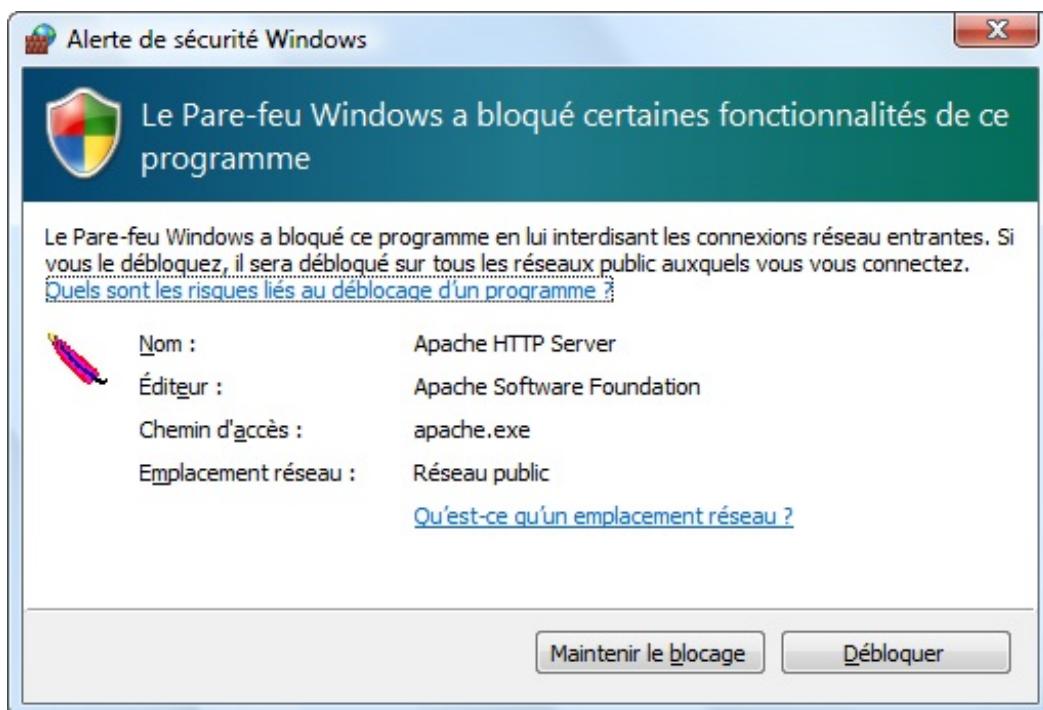
Commencez par télécharger WAMP sur son [site web officiel](#). Rendez-vous sur la page "Téléchargement". Vous n'êtes pas obligé de remplir le formulaire, il vous suffit de descendre tout en bas de la page et de cliquer sur "Télécharger WampServer".

Une fois téléchargé, installez-le en laissant toutes les options par défaut. Il devrait s'installer dans un répertoire comme C:\wamp et créer un raccourci dans le menu Démarrer.

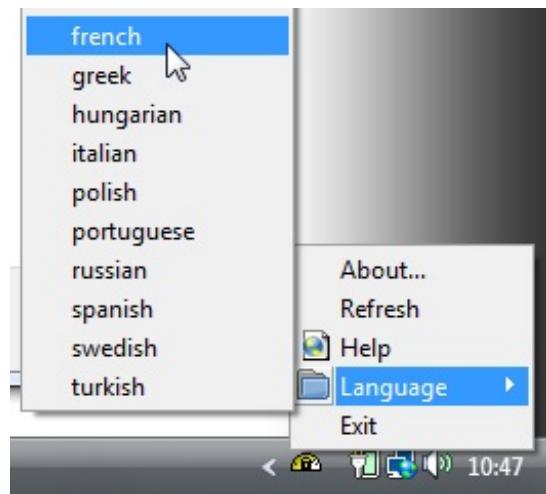
Lorsque vous lancez WAMP, une icône doit apparaître en bas à droite de la barre des tâches, à côté de l'horloge :



Si une fenêtre apparaît pour vous indiquer que le pare-feu bloque Apache, cliquez sur Débloquer. Vous n'avez aucune raison de vous inquiéter, c'est parfaitement normal. 😊

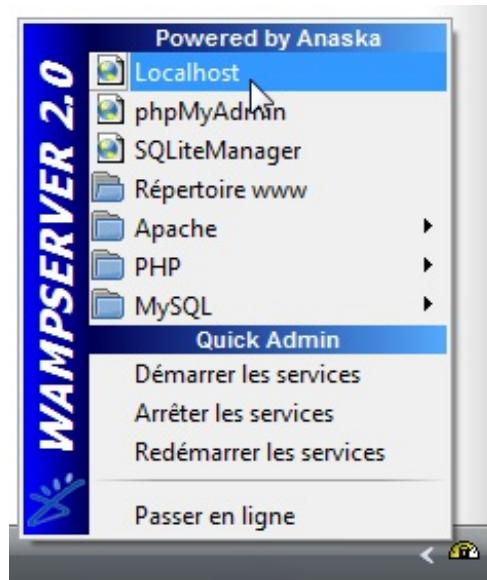


Par défaut, WAMP est en anglais. Vous pouvez facilement le passer en français en faisant un clic droit sur l'icône de WAMP dans la barre des tâches, puis en allant dans le menu Language / french.

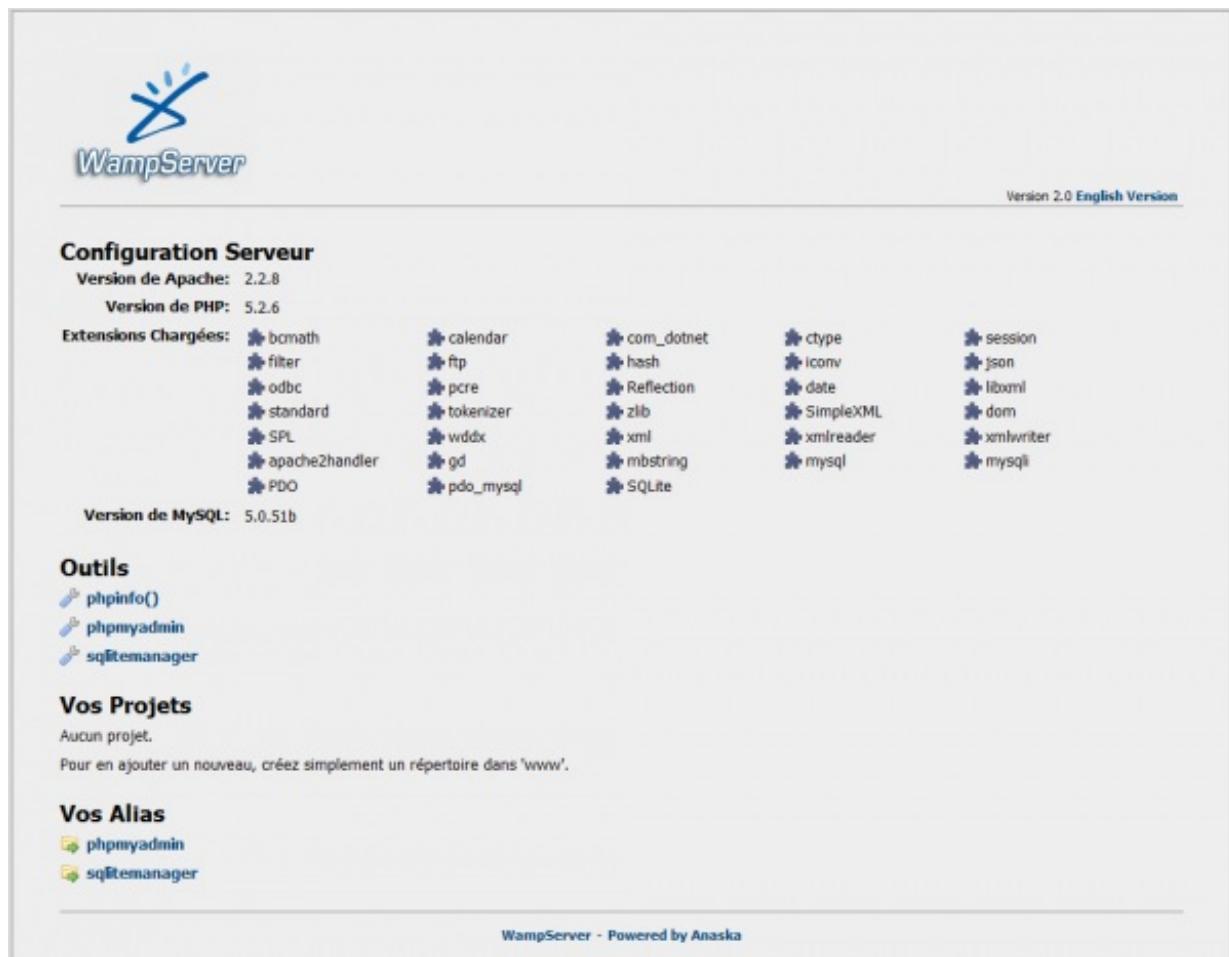


WAMP est maintenant en français ! 😊

Vous pouvez maintenant lancer la page d'accueil de WAMP. Faites un clic gauche sur l'icône de WAMP (attention, j'ai bien dit un **clic gauche** cette fois), puis cliquez sur Localhost.



Une page web similaire à la capture ci-dessous devrait s'ouvrir dans votre navigateur favori (Firefox par exemple). Si la page s'affiche chez vous, cela signifie qu'Apache fonctionne.



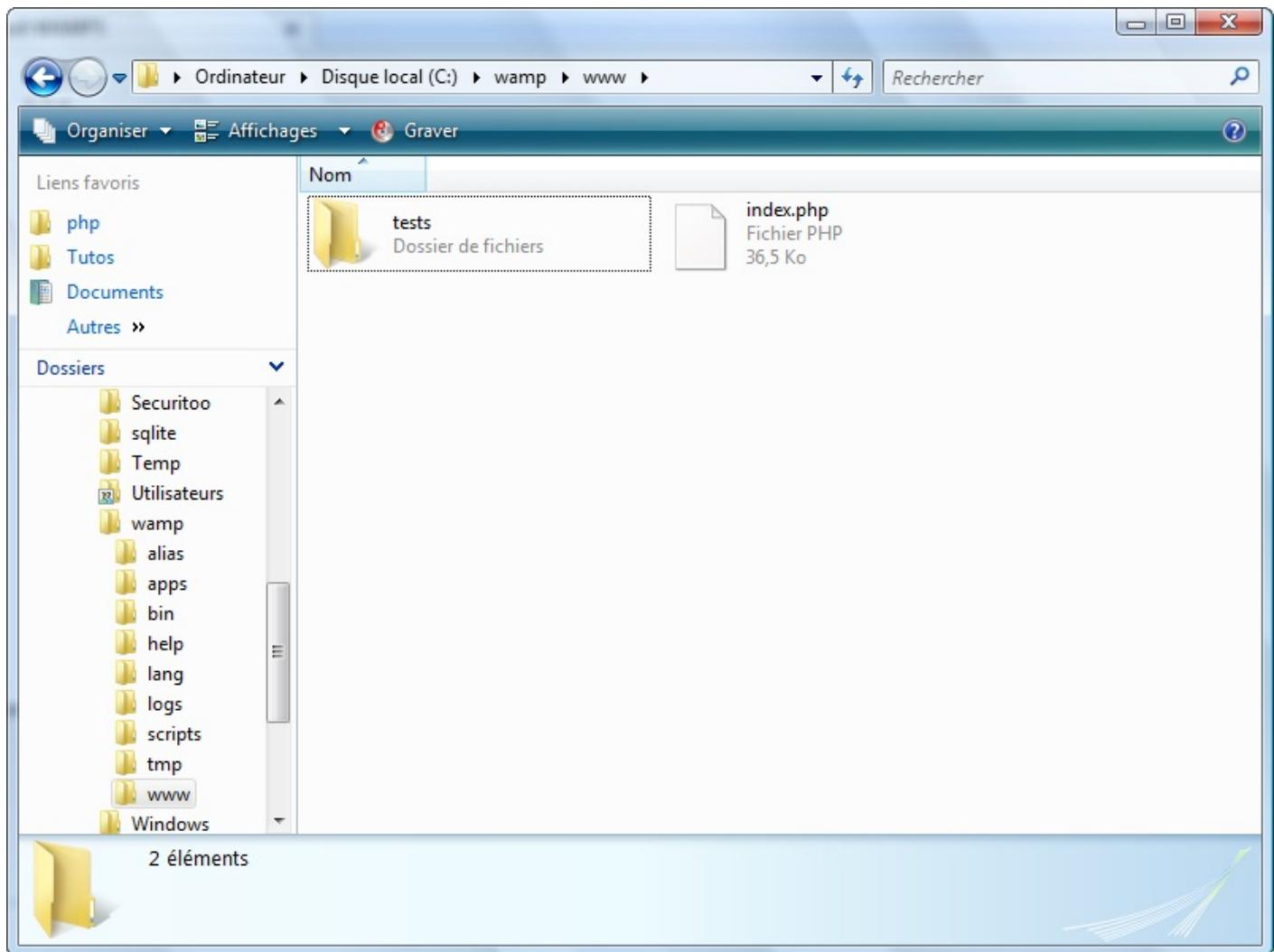
The screenshot shows the WampServer control panel. At the top left is the WampServer logo. To its right is a link to "Version 2.0 English Version". Below the logo, the title "Configuration Serveur" is displayed. Under this, it shows "Version de Apache: 2.2.8" and "Version de PHP: 5.2.6". A section titled "Extensions Chargées:" lists various PHP extensions: bcmath, calendar, com_dotnet, ctype, filter, ftp, hash, iconv, odbc, pcntl, Reflection, date, standard, tokenizer, zlib, SimpleXML, SPL, wddx, xml, mbstring, dom, apache2handler, gd, mysql, SQLite, PDO, pdo_mysql, session, json, libxml, and mysqli. Below this, "Version de MySQL: 5.0.51b" is shown. A "Tools" section contains links to "phpinfo()", "phpmyadmin", and "sqitemanager". A "Vos Projets" section states "Aucun projet." and "Pour en ajouter un nouveau, créez simplement un répertoire dans 'www'." A "Vos Alias" section contains links to "phpmyadmin" and "sqitemanager". At the bottom, a footer bar reads "WampServer - Powered by Anaska".

 La page web que vous voyez à l'écran vous a été envoyée par votre propre serveur Apache que vous avez installé en même temps que WAMP. Vous êtes en train de simuler le fonctionnement d'un serveur web sur votre propre machine. Pour le moment, vous êtes le seul internaute à pouvoir y accéder. On dit qu'on travaille "*en local*". Notez que l'URL affichée par le navigateur dans la barre d'adresse est `http://localhost/`, ce qui signifie que vous naviguez sur un site web situé sur votre propre ordinateur.

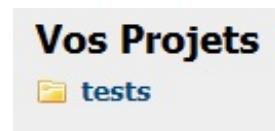
La section "Vos projets" de la page d'accueil de WAMP doit indiquer qu'aucun projet n'existe pour le moment. Considérez que chaque site web que vous entrepenez de faire est un nouveau projet.

Nous allons créer un projet de test que nous appellerons `tests`. Pour ce faire, ouvrez l'explorateur Windows et rendez-vous dans le dossier où WAMP a été installé, puis dans le sous-dossier intitulé `www`. Par exemple : `c:\wamp\www`. Vous pouvez aussi faire clic gauche / répertoire `www` sur l'icône de WAMP.

Une fois dans ce dossier, créez un nouveau sous-dossier que vous appellerez `tests` :



Retournez sur la page d'accueil de WAMP et actualisez la page (vous pouvez appuyer sur la touche F5). La section "Vos projets" devrait maintenant afficher "tests" car WAMP a détecté que vous avez créé un nouveau dossier :



Vous créerez là-dedans vos premières pages web en PHP.

Vous pouvez cliquer sur le lien "tests". Comme vous n'avez pas encore créé de fichier PHP, vous devriez voir une page vide comme ceci :

The screenshot shows a Mozilla Firefox window displaying a directory index for the URL <http://localhost/tests/>. The title bar reads "Index of /tests - Mozilla Firefox". The menu bar includes "Fichier", "Édition", "Affichage", "Historique", "Delicious", "Outils", and a question mark icon. The toolbar includes back, forward, stop, search, and other standard browser icons. The address bar shows the URL. The search bar contains "Google". The download bar has one item. The main content area displays the following table:

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
Parent Directory	-		

In the bottom left corner of the Firefox window, there is a status bar with the text "Terminé".

Si vous avez le même résultat, cela signifie que tout fonctionne. Bravo, vous avez installé WAMP et il fonctionne correctement. Vous êtes prêt à programmer en PHP !

Vous pouvez passer les sections suivantes qui ne concernent que les utilisateurs sous Mac OS X et Linux.

Sous Mac OS X : MAMP

Pour ceux qui ont un Mac sous Mac OS X, je vous conseille le programme MAMP (Mac Apache MySQL PHP). Il est vraiment très simple à installer et à utiliser.

Rendez-vous sur le [site officiel de MAMP](#) et cliquez sur "Download Now" sur la page d'accueil :



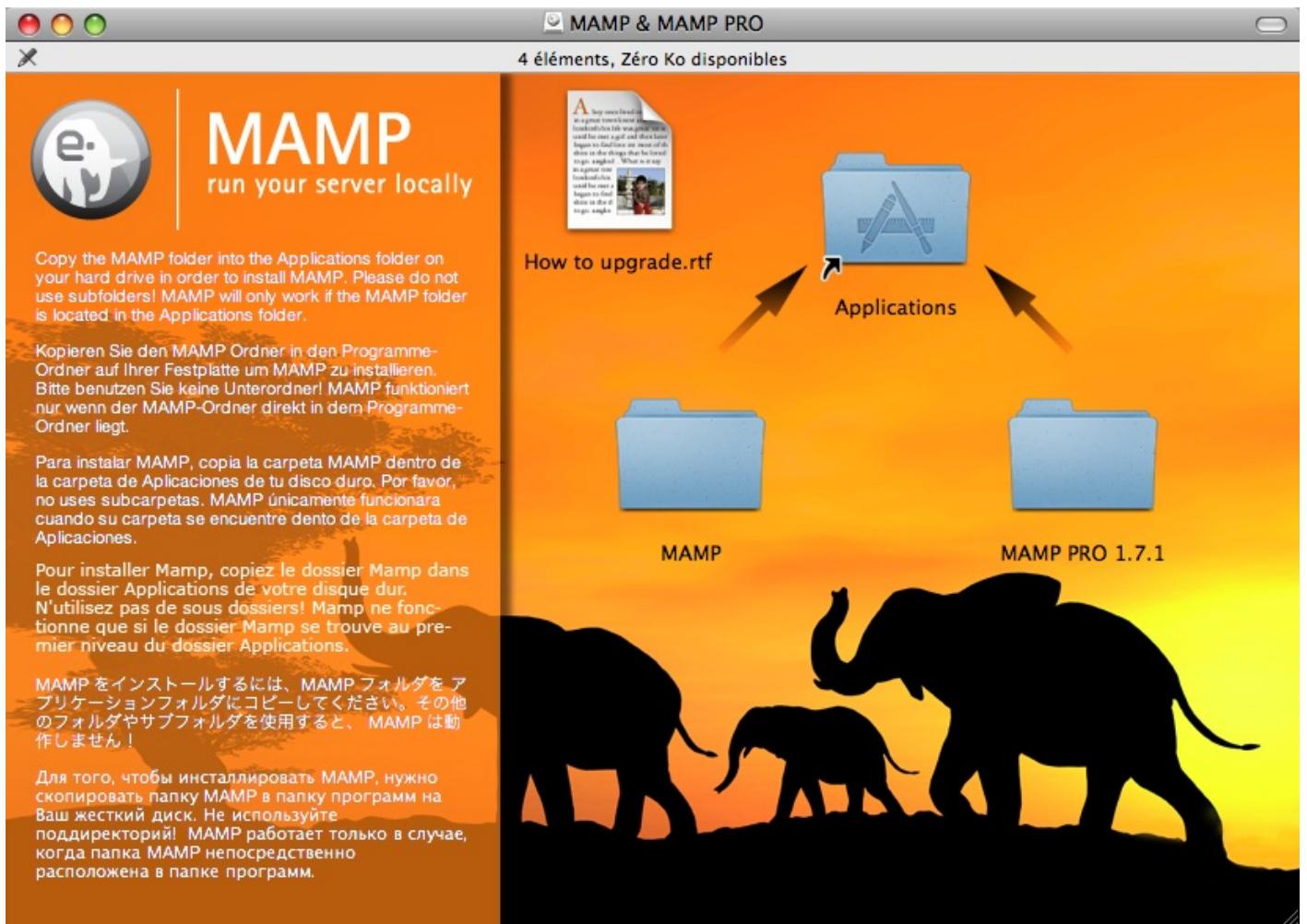
**MAMP: One-click-solution for
setting up your personal webserver**

[Download now](#)



Si vous avez une version de Mac OS X antérieure à Mac OS X 10.4, vous devrez télécharger une ancienne version de MAMP grâce aux liens présents un peu plus bas sur la même page.

Vous devriez avoir téléchargé une archive au format .dmg qui contient MAMP. Lorsque vous l'ouvrez, la fenêtre ci-dessous apparaît :

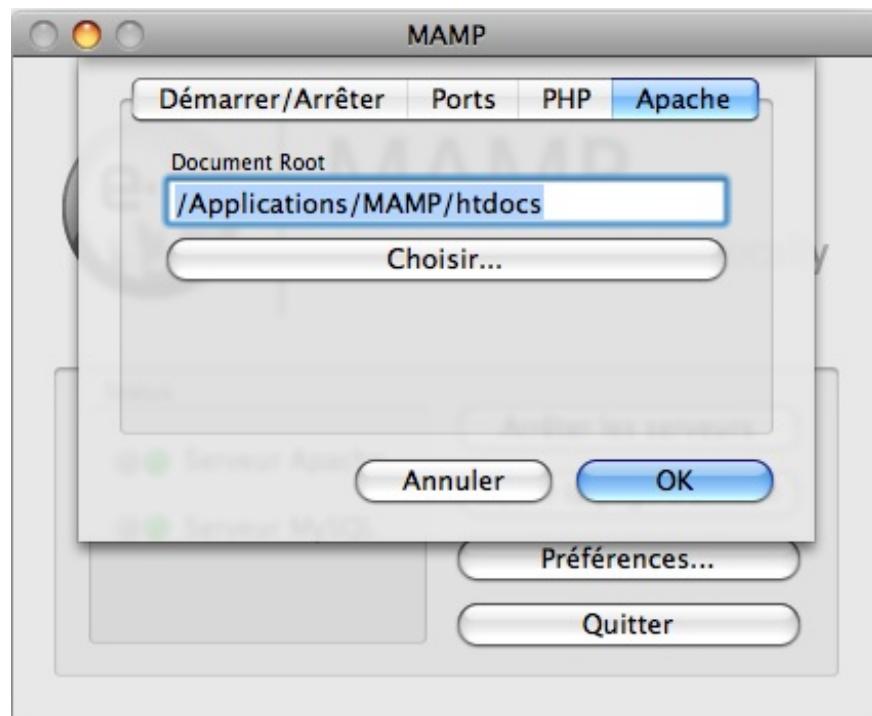


Vous devez tout simplement faire glisser le dossier MAMP en bas à gauche vers le dossier Applications au-dessus. 😊

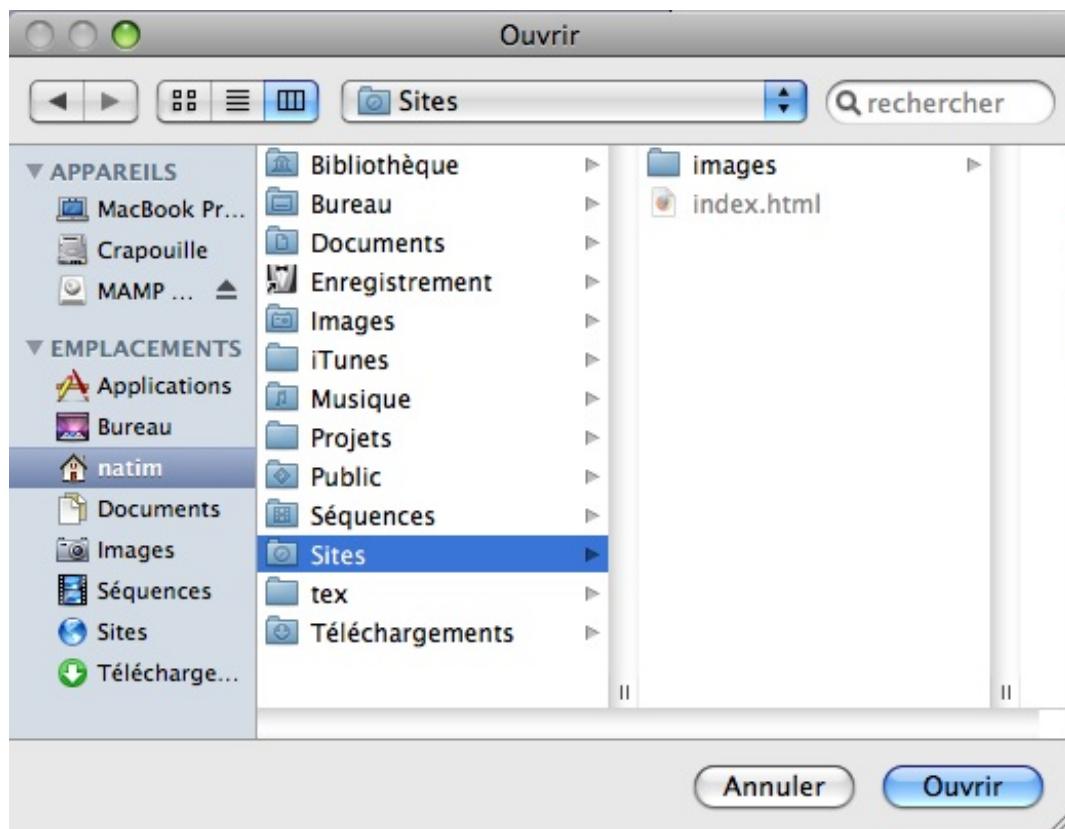
MAMP est maintenant installé. Vous le trouverez dans votre dossier "Applications". La fenêtre principale de MAMP indique que les serveurs Apache et MySQL ont été correctement démarrés. L'icône de chacun de ces éléments doit être verte :



Je vous invite à configurer le répertoire dans lequel Apache ira chercher les fichiers PHP de votre site web. Pour cela, cliquez sur le bouton Préférences de la fenêtre principale. Une boîte de dialogue de configuration s'ouvre. Cliquez sur l'onglet Apache en haut :



Cliquez sur le bouton "Choisir" pour sélectionner le dossier dans lequel vous placerez les fichiers de votre site web. Sous Mac OS, un dossier est déjà créé : il s'agit de "Sites" dans votre répertoire personnel.



Sélectionnez ce répertoire (notez que ce n'est pas une obligation : vous pouvez utiliser n'importe quel autre répertoire si vous

le désirez), qui devrait être de la forme /Users/pseudo/Sites.

Validez les changements et retournez sur la fenêtre principale de MAMP. Là, cliquez sur "Ouvrir la page d'accueil". Votre navigateur (Firefox ou Safari par exemple) devrait alors s'ouvrir et afficher une page web.

Pour vous préparer pour la suite, je vous invite à créer un dossier "tests" dans votre répertoire "Sites". Nous placerons nos premiers fichiers PHP de test à l'intérieur.

Si le dossier "tests" a été correctement créé, vous pouvez visualiser son contenu en allant avec votre navigateur à l'adresse :
`http://localhost:8888/tests/`

Si tout va bien, une page vide devrait s'afficher :



Index of /tests

Name	Last modified	Size	Description
Parent Directory	-	-	

Apache/2.0.59 (Unix) PHP/5.2.5 DAV/2 Server at localhost Port 8888

MAMP est correctement installé et configuré. Vous êtes maintenant prêt à travailler en PHP pour le chapitre suivant ! 😊

Sous Linux : XAMPP

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

Ce pack est plus complet que WAMP pour Windows ou MAMP pour Mac OS X. Nous n'utiliserons toutefois qu'une partie des éléments installés.

Sur le site officiel de [XAMPP](#), recherchez le lien XAMPP pour Linux :

 [XAMPP pour Linux](#) 

Ce kit pour les systèmes Linux (testé sous SuSE, RedHat, Mandrake and Debian) comprend : Apache, MySQL, PHP & PEAR, Perl, ProFTPD, phpMyAdmin, OpenSSL, GD, Freetype2, libjpeg, libpng, gdbname, zlib, expat, Sablotron, libxml, Ming, Webalizer, pdf class, ncurses, mod_perl, FreeTDS, gettext, mcrypt, mhash, eAccelerator, SQLite et IMAP C-Client.



XAMPP est aussi disponible pour Windows et Mac OS X comme vous pourrez le constater sur le site. La méthode d'installation est sensiblement différente, mais vous pouvez l'essayer si vous avez déjà testé WAMP (pour Windows) ou MAMP (pour Mac OS X) et qu'il ne vous convient pas.

Sur la page qui s'affiche, recherchez le lien de téléchargement de XAMPP pour Linux un peu plus bas :

Version	Taille	Notes
 XAMPP Linux 1.6.7	59 Mo	Apache 2.2.9, MySQL 5.0.51b, PHP 5.2.6 & 4.4.8 & PEAR + SQLite 2.8.17/3.3.17 + multibyte (mbstring) support, Perl 5.10.0, ProFTPD 1.3.1, phpMyAdmin 2.11.7, OpenSSL 0.9.8h, GD 2.0.1, Freetype2 2.1.7, libjpeg 6b, libpng 1.2.12, gdbname 1.8.0, zlib 1.2.3, expat 1.2, Sablotron 1.0, libxml 2.6.31, Ming 0.3, Webalizer 2.01, pdf class 009e, ncurses 5.8, mod_perl 2.0.4, FreeTDS 0.63, gettext 0.11.5, IMAP C-Client 2004e, OpenLDAP (client) 2.3.11, mcrypt 2.5.7, mhash 0.8.18, eAccelerator 0.9.5.3, cURL 7.18.2, libxslt 1.1.8, phpsQLiteAdmin 0.2, libapreq 2.08, FPDF 1.53, XAMPP Control Panel 0.6 Vérification MD5: 84c9e3543e5367cbe40bffa18f0e82d9

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

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

Code : Console

```
cd ~/Desktop
```

Vous devez passer root pour installer et lancer XAMPP.

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

Si comme moi vous utilisez Ubuntu, tapez donc :

Code : Console

```
sudo su
```

Vous devez maintenant extraire le dossier compressé dans /opt. Pour ce faire, recopiez simplement la commande suivante :

Code : Console

```
tar xvfz xampp-linux-1.6.7.tar.gz -C /opt
```



Il se peut que le nom du fichier soit légèrement différent si le numéro de version a changé. Adaptez le nom du fichier en le complétant automatiquement à l'aide de la touche Tabulation.

Lorsque la décompression des fichiers est terminée, c'est fait ! XAMPP est maintenant installé.

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

Code : Console

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

Pour arrêter XAMPP plus tard si vous le désirez, tapez :

Code : Console

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



N'oubliez pas que vous devez être root lorsque vous démarrez ou arrêtez XAMPP.

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

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

Vous devriez voir la page de sélection de la langue de XAMPP. Cliquez sur "Français" :



[English](#) / [Deutsch](#) / [Français](#) / [Nederlands](#) / [Polski](#) / [Italiano](#) / [Norsk](#) / [Español](#) / [中文](#) / [Português \(Brasil\)](#) / [日本語](#)

La page principale de configuration de XAMPP s'affiche ensuite. Elle est plus complète que ses homologues WAMP et MAMP, notamment parce que XAMPP contient plus de logiciels et propose donc plus de fonctionnalités (beaucoup plus !).

Vous pouvez vérifier si tout fonctionne correctement en allant dans le menu Statut :

The screenshot shows the XAMPP for Linux status page. On the left, there's a sidebar with links like Bienvenue, Statut, Sécurité, Documentation, Composants, Démos, and Outils. The main content area has a title "Statut XAMPP" and a sub-section "Composant" with a table. The table lists various components and their status:

Composant	Statut	Conseil
Base de Données MySQL	ACTIVE	
PHP	ACTIVE	
Perl avec mod_perl	ACTIVE	
Common Gateway Interface (CGI)	ACTIVE	
Server Side Includes (SSI)	ACTIVE	
PHP extension "eAccelerator"	DESACTIVE	voir FAQ
PHP extension "OCI8/Oracle"	DESACTIVE	voir FAQ

Below the table, a note says: "Certains changements dans la configuration peuvent causer de faux rapports de statuts erronés. Avec SSI". At the bottom left, it says "©2002-2008 ...APACHE FRIENDS...".

Au minimum, les modules MySQL et PHP doivent être en vert. Quant aux autres, nous ne les utiliserons pas donc peu importe.



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

Code : Console

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

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

Vous devriez voir une page similaire à ceci :

The screenshot shows a Mozilla Firefox browser window. The title bar reads "Index of /tests - Mozilla Firefox". The menu bar includes "Fichier", "Édition", "Affichage", "Historique", "Delicious", "Marque-pages", "Outils", and "Aide". The toolbar includes icons for back, forward, search, and refresh. The address bar shows "http://localhost/tests/" and has a "Google" search button. The main content area displays the text "Index of /tests" followed by a table header and one row. The footer of the browser window says "Terminé".

Name	Last modified	Size	Description
Parent Directory	-	-	-

Apache/2.2.9 (Unix) DAV/2 mod_ssl/2.2.9 OpenSSL/0.9.8h PHP/5.2.6
mod_apreq2-20051231/2.6.0 mod_perl/2.0.4 Perl/v5.10.0 Server at localhost Port 80

Vous êtes prêts à travailler en PHP ! 😊



Si vous rencontrez des problèmes concernant les droits des fichiers du répertoire `/opt/lampp/htdocs`, dirigez-vous vers la documentation Ubuntu sur XAMPP (section *Démarrer XAMPP*).

Utiliser un bon éditeur de fichiers

Comme vous devez déjà le savoir, 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 écrire des pages web en XHTML et même en PHP mais...
- Le mieux reste d'utiliser un **logiciel spécialisé** qui colore votre code (très pratique) et qui numérote vos lignes (très pratique aussi). Il existe des centaines et des centaines de logiciels gratuits faits pour les développeurs comme vous.

Je vous propose donc d'installer un logiciel qui va vous permettre d'éditer vos fichiers source de manière efficace. Vous en avez probablement déjà installé un si vous avez appris à [programmer en XHTML / CSS à l'aide de mon cours](#), mais comme on n'est jamais trop prudent, je vais rapidement vous en présenter quelques-uns en fonction de votre système d'exploitation.

Voici le code source XHTML que nous allons utiliser pour commencer en terrain connu. Copiez-collez ce code dans l'éditeur de texte que je vais vous faire installer :

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



Il n'y a pas de PHP pour l'instant afin de commencer en douceur. Nous allons simplement essayer d'enregistrer un fichier XHTML avec ce code pour nous échauffer.

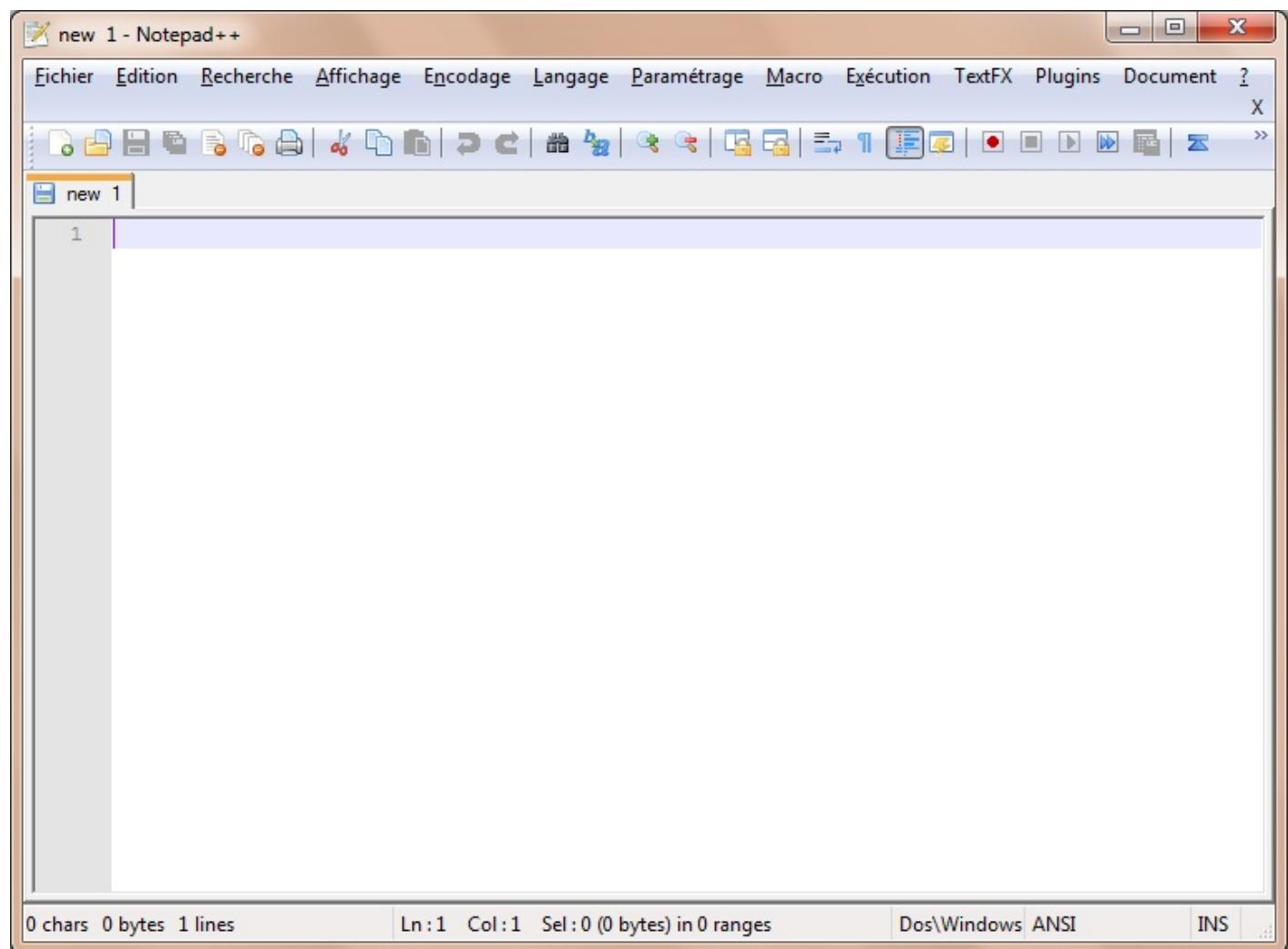
Sous Windows

Il existe beaucoup de logiciels gratuits à télécharger pour éditer du texte sous Windows. Il m'est impossible de tous vous les présenter donc je vais vous en recommander un qui est très utilisé et en lequel vous pouvez avoir confiance : Notepad++.

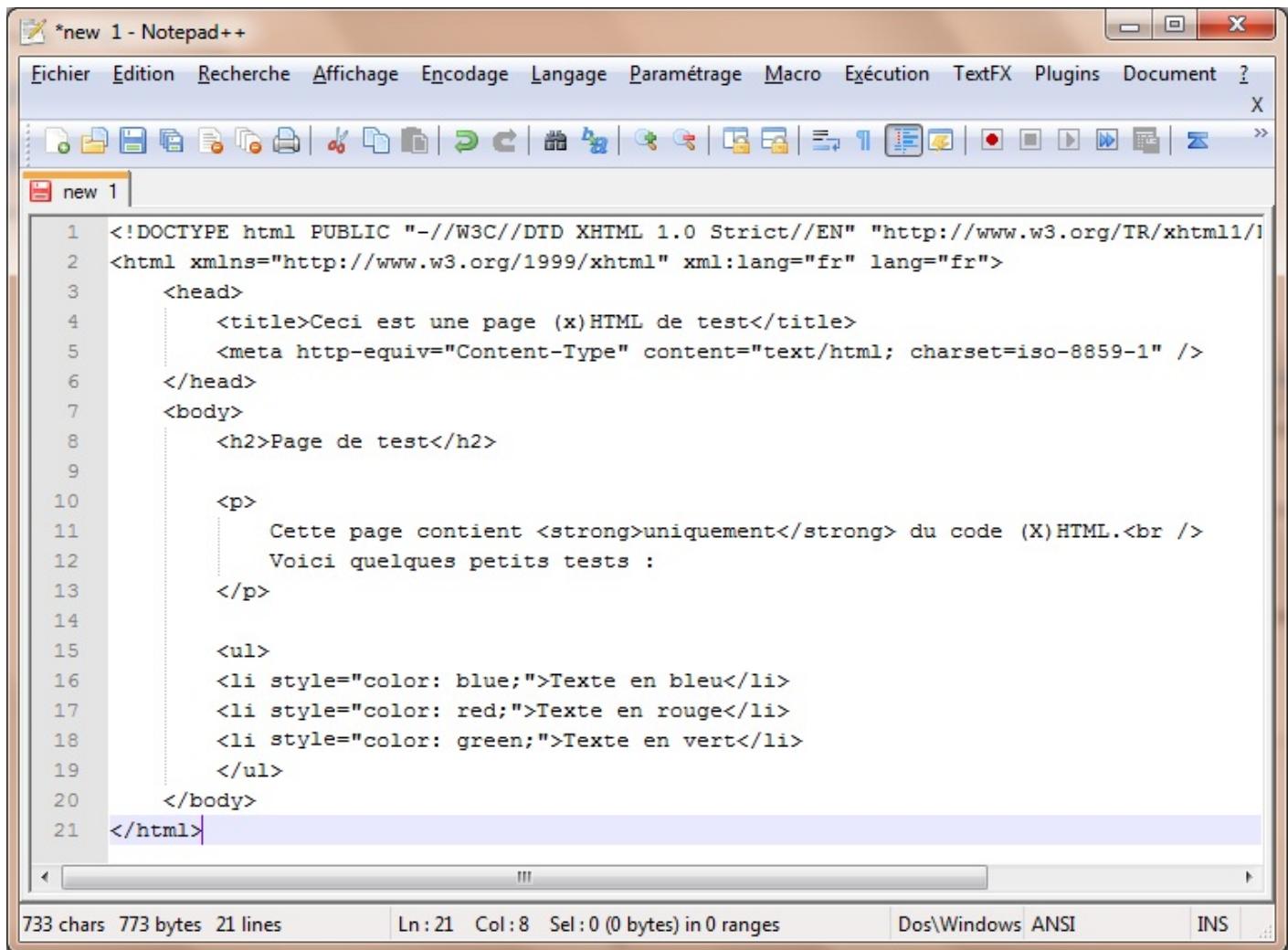
Ce logiciel est petit et rapide à télécharger. N'hésitez pas à l'essayer :

[Page de téléchargement de Notepad++](#)

Lorsque Notepad++ s'ouvre, il présente généralement un fichier vide (vous pouvez en créer un nouveau au besoin en allant dans le menu Fichier / Nouveau) :



Copiez-collez le code XHTML que je viens de vous donner dans Notepad++. Vous devriez voir l'écran suivant :



The screenshot shows the Notepad++ application window with the title bar "*new 1 - Notepad++". The menu bar includes Fichier, Edition, Recherche, Affichage, Encodage, Langage, Paramétrage, Macro, Exécution, TextFX, Plugins, Document, and Help. The toolbar below has various icons for file operations like Open, Save, Print, and Find. The main editor window contains the following HTML code:

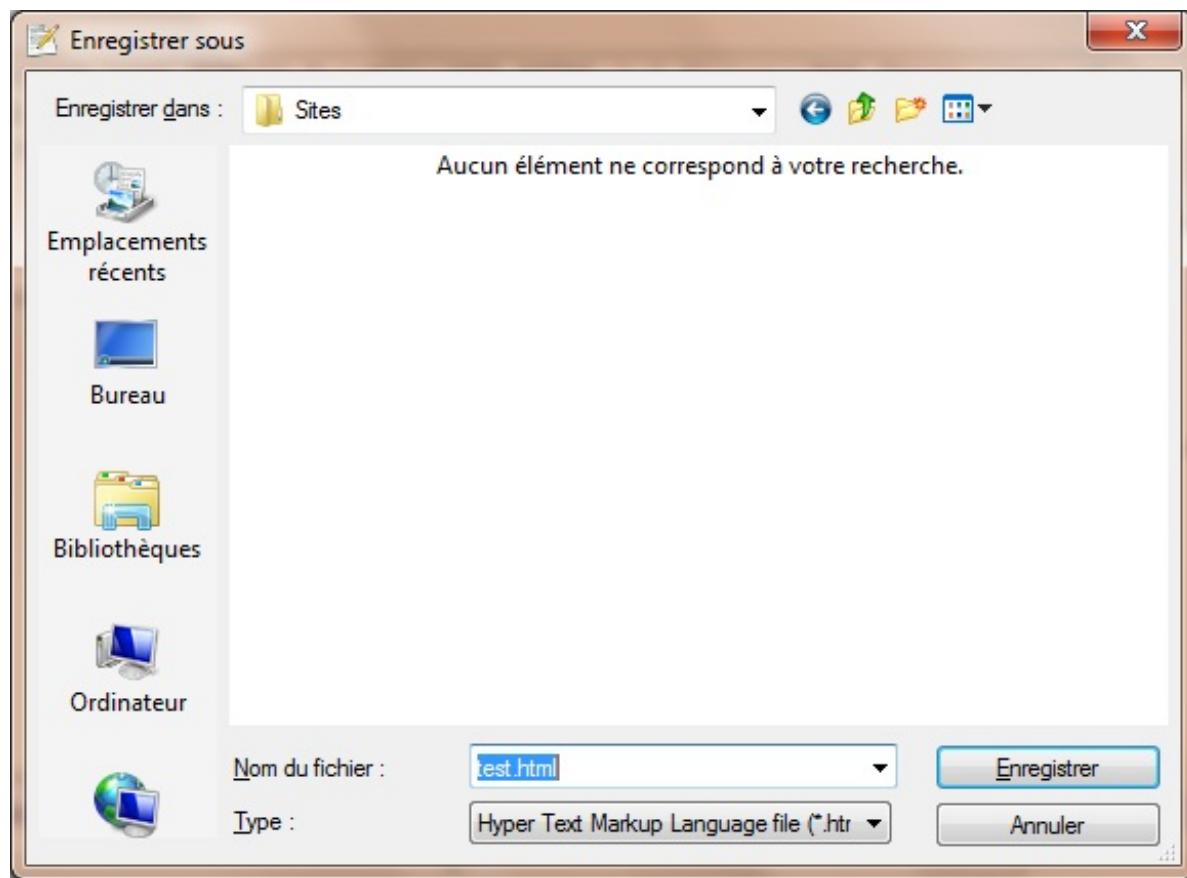
```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/1
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
3     <head>
4         <title>Ceci est une page (x)HTML de test</title>
5         <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
6     </head>
7     <body>
8         <h2>Page de test</h2>
9
10        <p>
11            Cette page contient <strong>uniquement</strong> du code (X)HTML.<br />
12            Voici quelques petits tests :
13        </p>
14
15        <ul>
16            <li style="color: blue;">Texte en bleu</li>
17            <li style="color: red;">Texte en rouge</li>
18            <li style="color: green;">Texte en vert</li>
19        </ul>
20    </body>
21 </html>
```

At the bottom, status bars show "733 chars 773 bytes 21 lines", "Ln: 21 Col: 8 Sel: 0 (0 bytes) in 0 ranges", "Dos\Windows ANSI", and "INS".

Comme vous pouvez le voir, le code n'est pas coloré. La raison vient du fait que Notepad++ ne sait pas de quel type de code source il s'agit. Vous devez au préalable enregistrer le fichier.

Allez dans Fichier / Enregistrer, puis choisissez le dossier où vous souhaitez enregistrer le fichier. Je vous conseille d'aller dans le dossier C:\wamp\www\tests que vous avez créé à l'installation de WAMP.

Choisissez le type de fichier .html (Hyper Text Markup Language File) puis donnez un nom à votre fichier :



Une fois le fichier enregistré, le code source apparaît coloré :

```
C:\Users\Mateo21\Documents\Sites\test.html - Notepad++
Fichier Edition Recherche Affichage Encodage Langage Paramétrage Macro Exécution TextFX Plugins Document ?
X
test.html
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml11
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
3      <head>
4          <title>Ceci est une page (x)HTML de test</title>
5          <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
6      </head>
7      <body>
8          <h2>Page de test</h2>
9
10         <p>
11             Cette page contient <strong>uniquement</strong> du code (X)HTML.<br />
12             Voici quelques petits tests :
13         </p>
14
15         <ul>
16             <li style="color: blue;">Texte en bleu</li>
17             <li style="color: red;">Texte en rouge</li>
18             <li style="color: green;">Texte en vert</li>
19         </ul>
20     </body>
21 </html>

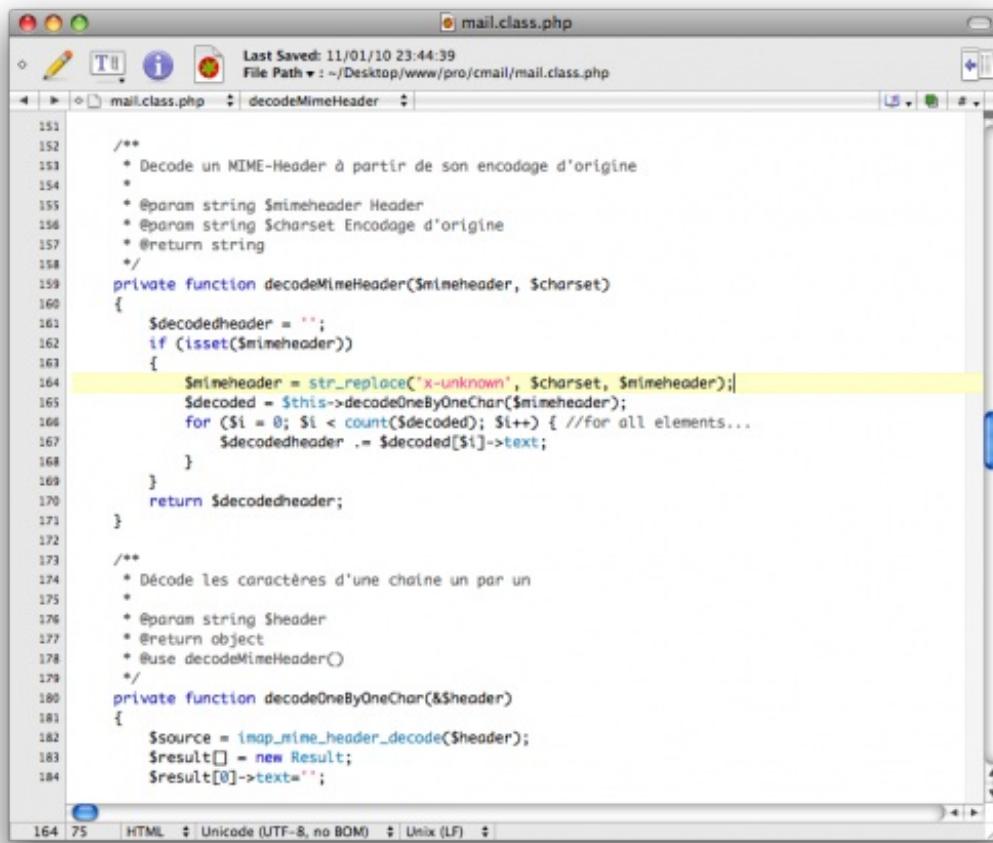
733 chars 773 bytes 21 lines Ln: 21 Col: 8 Sel: 0 (0 bytes) in 0 ranges Dos\Windows ANSI INS
```

Vous pourrez suivre la même procédure avec les fichiers PHP tout à l'heure, à condition d'enregistrer le fichier en .php. Ca vous entraînera vous verrez. 😊

Sous Mac OS X

Si vous êtes sous Mac, je peux vous recommander l'éditeur [TextWrangler](#) qui est gratuit. Il existe aussi [Fraise](#) que vous pouvez essayer.

D'autres éditeurs de texte payants de qualité existent, notamment l'excellent [TextMate](#).



The screenshot shows the TextWrangler application window on a Mac OS X desktop. The title bar reads "mail.class.php". The main pane displays a block of PHP code. A yellow highlight covers several lines of code starting from line 164, which contains a call to `str_replace('x-unknown', $charset, $mimeheader);`. The code is a class method named `decodeMimeHeader` that takes a `$mimeheader` string and a `$charset` string as parameters, returning a decoded string. Below this, another method `decodeOneByOneChar` is partially visible, utilizing the `imap_mime_header_decode` function. The status bar at the bottom indicates the file is saved on 11/01/10 at 23:44:39, has a file path of ~/Desktop/www/pro/cmail/mail.class.php, and shows encoding as HTML, Unicode (UTF-8, no BOM), and line endings as Unix (LF).

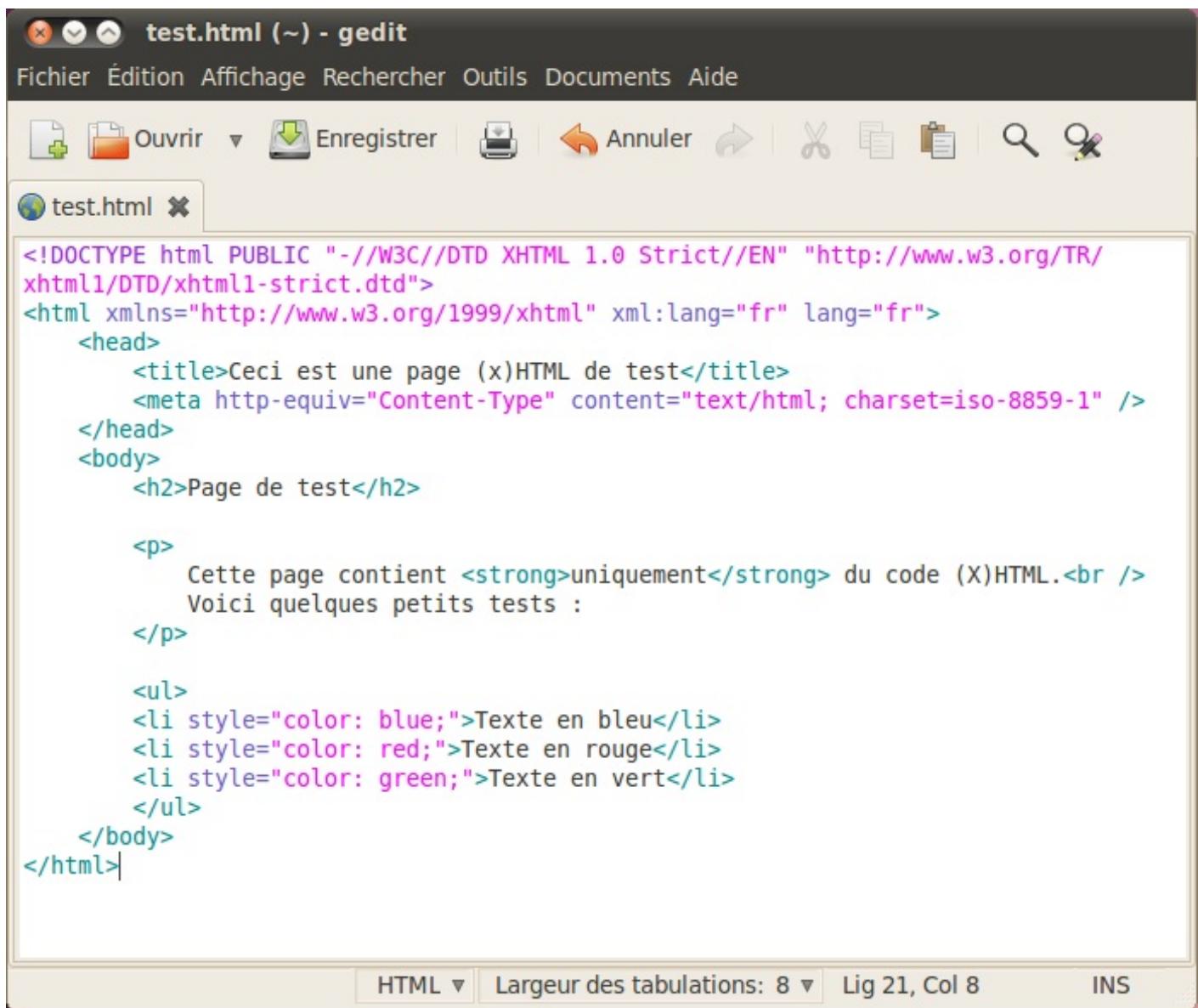
```
151
152 /**
153 * Décode un MIME-Header à partir de son encodage d'origine
154 *
155 * @param string $mimeheader Header
156 * @param string $charset Encodage d'origine
157 * @return string
158 */
159 private function decodeMimeHeader($mimeheader, $charset)
160 {
161     $decodedheader = '';
162     if (isset($mimeheader))
163     {
164         $mimeheader = str_replace('x-unknown', $charset, $mimeheader);
165         $decoded = $this->decodeOneByOneChar($mimeheader);
166         for ($i = 0; $i < count($decoded); $i++) { //for all elements...
167             $decodedheader .= $decoded[$i]->text;
168         }
169     }
170     return $decodedheader;
171 }
172
173 /**
174 * Décode les caractères d'une chaîne un par un
175 *
176 * @param string $header
177 * @return object
178 * @use decodeMimeHeader()
179 */
180 private function decodeOneByOneChar(&$header)
181 {
182     $source = imap_mime_header_decode($header);
183     $result[] = new Result;
184     $result[0]->text='';
```

L'éditeur de texte TextWrangler sous Mac OS X

Sous Linux

Sous Linux, les bons éditeurs ne manquent pas. Si vous êtes un habitué de la console, vous travaillerez sûrement avec plaisir avec vim ou emacs.

Si vous recherchez un éditeur graphique plus facile à utiliser, je vous recommande gedit ou tout autre logiciel installé avec votre distribution Linux, cela fera l'affaire.



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

HTML ▾ Largeur des tabulations: 8 ▾ Lig 21, Col 8 INS

L'éditeur de texte gedit sous Linux

Quel que soit le logiciel que vous utilisez, assurez-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.

Vous devriez maintenant avoir installé un pack type WAMP, MAMP ou XAMPP pour pouvoir exécuter du PHP ainsi qu'un éditeur de texte comme Notepad++ ou TextWrangler pour travailler sur le code source de vos fichiers.

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

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

Ne vous attendez pas à un résultat extraordinaire (en fait la page que nous allons créer ne va rien faire de spécial) mais ça va vous permettre de prendre nos marques. Vous allez en particulier comprendre comment on sépare le code XHTML classique du code PHP.

Vous êtes prêts ? Allons-y ! 

Les balises PHP

Vous savez donc que le code source d'une page XHTML est constitué de *balises* (aussi appelées *tags*). Par exemple est une balise.

Le code PHP vient s'insérer au milieu du code XHTML. On va progressivement placer dans nos pages web des morceaux de code PHP à l'intérieur du XHTML. Ces bouts de code PHP seront les parties *dynamiques* de la page, c'est-à-dire les parties qui peuvent changer toutes seules (c'est pour cela qu'on dit qu'elles sont dynamiques).

Voici un schéma qui illustre cela :

```
<!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>Ma page web</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>
        <h1>Ma page web</h1>

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

Comme vous pouvez le voir, on retrouve le code XHTML que l'on connaît bien... et on insère en plus au milieu des données dynamiques. Ici, par exemple, c'est le pseudonyme : il change en fonction du visiteur. La partie surlignée en vert peut donc changer selon les visiteurs.



Le Site du Zéro fait la même chose pour ses membres inscrits. Votre pseudonyme est affiché en haut des pages lorsque vous êtes connecté au Site du Zéro.

La forme d'une balise PHP

Si je vous parle de cela, ce n'est pas par hasard. Pour utiliser du PHP, on va devoir introduire une nouvelle balise... et 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.

Voici une balise PHP vide :

Code : PHP

```
<?php ?>
```

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

Code : PHP

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

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

Code : PHP

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

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

Insérer une balise PHP au milieu du code XHTML

La balise PHP que nous venons de découvrir s'insère au milieu du code XHTML comme je vous l'ai dit plus tôt. Pour reprendre l'exemple qu'on a vu au chapitre précédent :

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" />  
    </head>  
    <body>  
        <h2>Page de test</h2>  
  
        <p>  
            Cette page contient du code (x)HTML avec des balises  
PHP.<br />  
            <?php /* Insérer du code PHP ici */ ?>  
            Voici quelques petits tests :  
        </p>  
  
        <ul>  
            <li style="color: blue;">Texte en bleu</li>  
            <li style="color: red;">Texte en rouge</li>  
            <li style="color: green;">Texte en vert</li>  
        </ul>  
  
        <?php  
        /* Encore du PHP  
Toujours du PHP */  
        ?>  
    </body>  
</html>
```

Bien entendu cette page ne fonctionne pas vu que nous n'avons pas encore écrit de vrai code PHP (ce sont juste des balises d'exemple). 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 ?>`



On peut placer une balise PHP n'importe où dans le code ?

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

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 <?php /* Code PHP */ ?></title>
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
  </head>
```

Plus fort encore, vous pouvez même insérer une balise PHP au milieu d'une balise XHTML (bon ce n'est pas très joli je vous l'accorde) :

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</title>
    <meta http-equiv="Content-Type" <?php /* Code PHP */ ?>
      content="text/html; charset=iso-8859-1" />
  </head>
```



Comment ça fonctionne ? A quoi ça peut servir ?

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

Afficher du texte

Bon tout ça c'est bien beau, mais il est temps de commencer à écrire 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. 🤪

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

L'instruction echo

Le PHP est un langage de programmation, ce qui n'était pas le cas du XHTML (on parlait plutôt de langage de description, car il permet de décrire une page web). Si vous avez déjà programmé dans d'autres langages comme le C ou le Java, cela ne devrait pas vous surprendre. Néanmoins, dans ce cours, nous partons de Zéro donc je vais supposer que vous n'avez jamais fait de programmation auparavant. 😊

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

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

Voici un exemple d'utilisation de cette instruction :

Code : PHP

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

Comme vous le voyez, à l'intérieur de la balise PHP on écrit l'instruction `echo` suivie du texte à afficher entre guillemets. Les guillemets permettent de délimiter le début et la fin du texte, cela aide l'ordinateur à se repérer. Enfin, l'instruction se termine par un point-virgule comme je vous l'avais annoncé, ce qui signifie *Fin de l'instruction*.



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

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

Code : PHP

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

Le mot "texte" 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. Vous risquez au mieux de faire planter votre beau code et d'avoir une terrible "Parse error". 😬

La solution consiste à faire précédé le guillemet d'un backslash \ :

Code : PHP

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

Vous savez que le code PHP s'insère au milieu du code XHTML. Alors allons-y, prenons une page basique en XHTML et plaçons-y du code 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" 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>
```

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

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

Enregistrer une page PHP

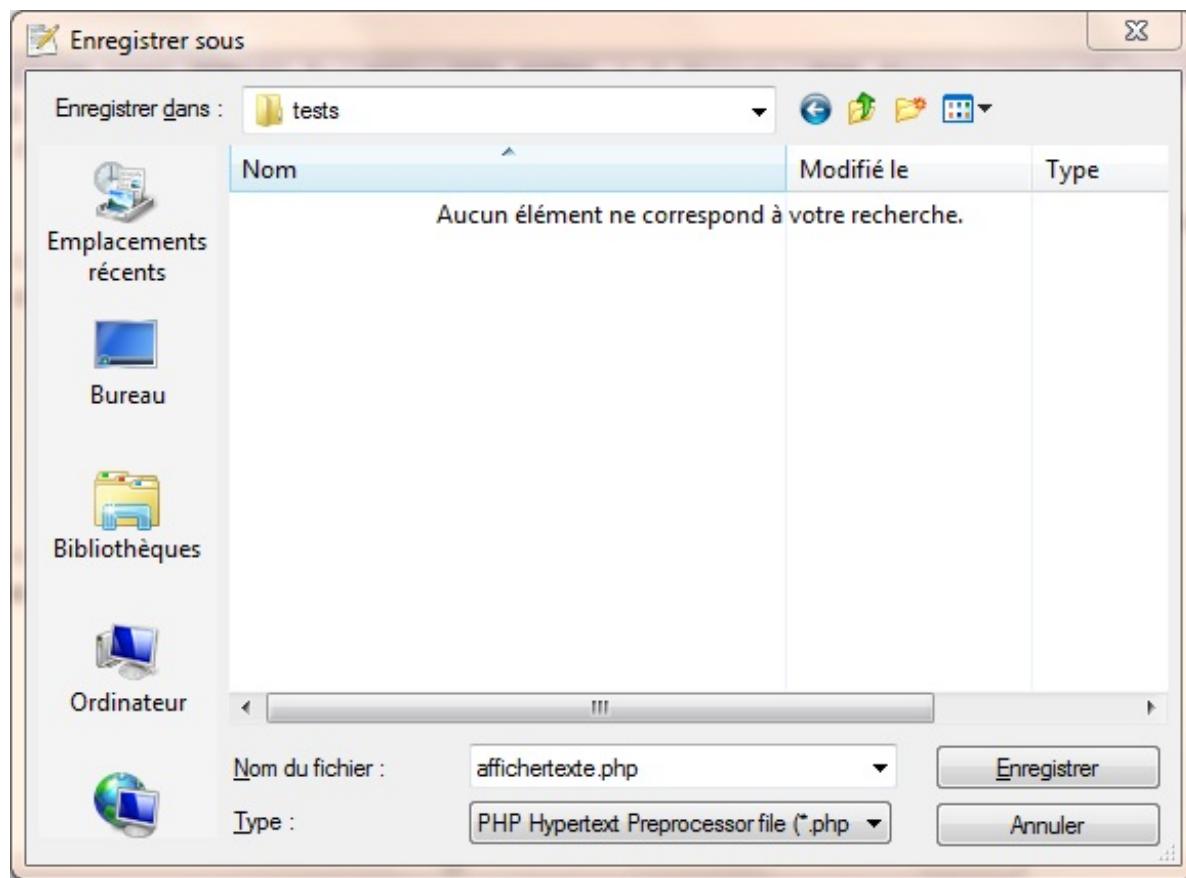
Je vous ai expliqué comment faire dans le chapitre précédent mais un petit rappel ne peut pas faire de mal.

Enregistrez la page avec l'extension .php, par exemple affichertexte.php, dans le dossier tests que je vous ai fait créer. Il doit se trouver dans C:\wamp\www\tests sous Windows.



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

Si vous utilisez Notepad++, sélectionnez PHP Hypertext Preprocessor file (*.php) dans la fenêtre pour enregistrer :



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

Tester la page PHP

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

Sous Windows, démarrez WAMP si ce n'est déjà fait. Allez dans le menu `localhost`, la page d'accueil s'ouvre. Là, si vous avez bien créé le dossier `tests` dans le répertoire `www` comme indiqué au chapitre précédent, vous devriez voir un lien vers le dossier `tests`. Cliquez dessus (nous avons déjà fait cela dans le chapitre précédent).

Une page web s'ouvre indiquant tous les fichiers qui se trouvent dans le dossier `tests`. 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. 😊

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

Je vous propose d'essayer aussi le résultat directement sur le Site du Zéro si vous le souhaitez pour comparer (mais je vous conseille fortement de savoir afficher la page chez vous directement). Cliquez sur le lien "Essayer !" ci-dessous :

[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 ?



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

Comment PHP génère du code XHTML

L'instruction `echo` demande à PHP d'insérer à cet endroit le texte que vous demandez. Si on traduit l'instruction en français, ça donnerait : *Insérer le texte : "Celle-ci a été écrite entièrement en PHP."*.

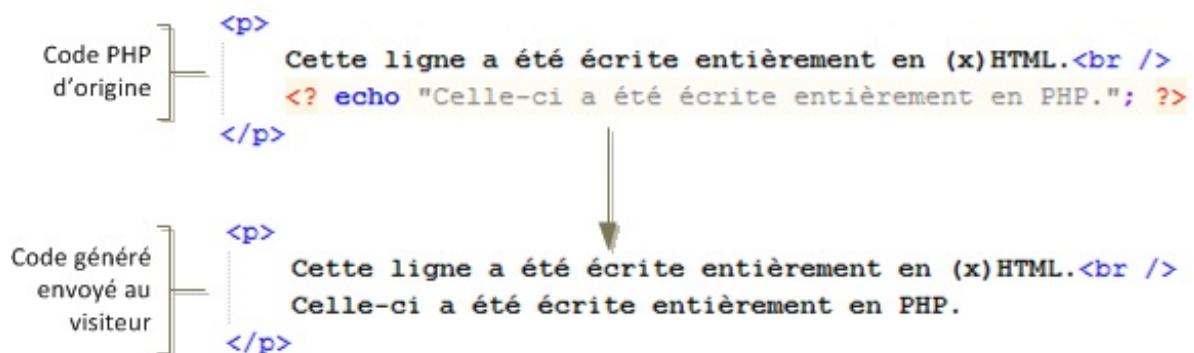
 Il ne faut jamais oublier le point-virgule à la fin d'une instruction. Si jamais ça arrive, vous aurez le message d'erreur : "Parse Error"

Notez que ça plante uniquement si votre code PHP fait plus d'une ligne (ça sera tout le temps le cas). Donc prenez l'habitude de toujours mettre un ";" à la fin des instructions.

Je vous ai expliqué dans le tout premier chapitre que le PHP générait du code XHTML et renvoyait au visiteur uniquement du code XHTML (accompagné de sa feuille de style CSS éventuellement) :



Ici, concrètement, voici ce qu'il se passe avec notre code source :



Le code PHP est exécuté en premier et l'ordinateur fait ce qu'on lui demande. Ici on lui a dit "Affiche ce texte ici".

Une fois toutes les instructions PHP exécutées (ici c'était simple, il n'y en avait qu'une), la page qui sort est une page qui ne contient que du XHTML ! C'est cette page de "résultat" qui est envoyée au visiteur, car celui-ci ne sait lire que le XHTML.



Rappelez-vous, seul le serveur peut exécuter du PHP. Le PHP n'est **jamais** envoyé au visiteur. Pour que nous puissions exécuter du PHP sur notre ordinateur (afin de faire nos tests), nous avons dû le transformer en mini-serveur en installant un programme tel que WAMP.

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 n'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.

Il existe 2 types de commentaires :

- Les commentaires monolignes
- Les commentaires multilignes

Tout dépend si votre commentaire est court ou long. Je vais vous présenter les deux.

Les commentaires monolignes

Pour indiquer que vous écrivez un commentaire sur une seule ligne, vous devez taper 2 slash : `//`. Tapez ensuite votre commentaire.

Un exemple ?

Code : PHP

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

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

Je vous ai mis deux commentaires à des endroits différents :

- 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.

Les commentaires multilignes

Ce sont les plus pratiques si vous pensez écrire un commentaire sur plusieurs lignes (mais on peut aussi s'en servir pour écrire

des commentaires d'une seule ligne). Il faut commencer par écrire `/*` puis refermer par `*/` :

Code : PHP

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

Ici les commentaires n'ont pas grande utilité, mais vous verrez comment je les utilise dans les prochains chapitres pour vous décrire le code PHP. Vous aussi vous allez vite apprendre 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 l'instruction `echo`, et vous allez même vous rendre compte à quel point cela permet de simplifier votre travail !

Inclure des portions de page

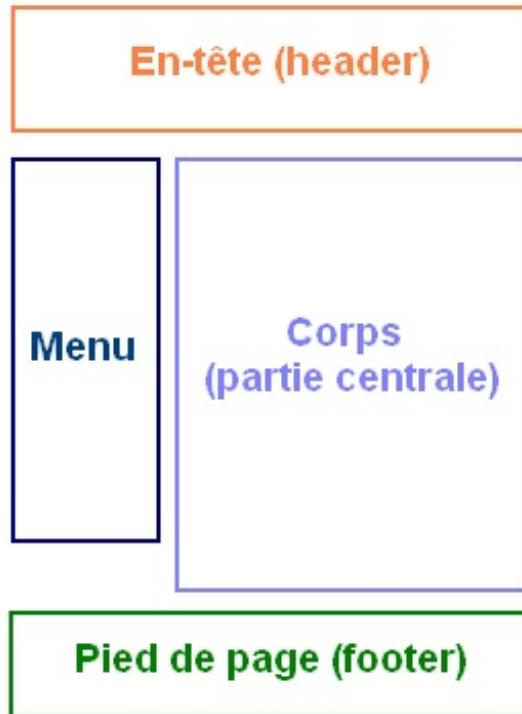
Vous est-il déjà arrivé de vouloir modifier le menu de votre site et de devoir pour cela corriger le code XHTML de chacune de vos pages web ? Le menu d'une page web apparaît en effet sur chacune des pages et vous avez très certainement dû le recopier sur chacune de vos pages. Ca marche, mais ce n'est pas très pratique...

Une des fonctionnalités les plus simples et les plus utiles de PHP est l'*inclusion de pages*. On peut très facilement inclure toute une page ou un bout de page à l'intérieur d'une autre page. Cela va grandement vous faciliter la tâche en vous évitant d'avoir à copier le même code XHTML plusieurs fois.

Avec ce chapitre, vous allez découvrir un des multiples avantages que vous donne le PHP lors de la création de votre site. C'est d'ailleurs ce qui m'a fait instantanément aimer ce langage lorsque je l'ai découvert, alors que je venais comme vous seulement d'apprendre le XHTML et le CSS. 😊

Le principe

La plupart des sites web sont généralement découpés selon le schéma suivant (que je reprends honteusement de mon propre cours sur le XHTML ):



Le Site du Zéro ne fait lui-même pas exception à la règle d'ailleurs :

The screenshot shows the homepage of www.siteduzero.com. The layout is divided into sections:

- En-tête (Header):** At the top, it features the site's logo (a graduation cap icon), the text "le Site du Zéro", and a user profile for "Habro21".
- Menu (Menu):** On the left side, there is a vertical sidebar menu titled "Menu" which includes links for "Site Web", "XHTML / CSS", "PHP / MySQL", "Langage C", "Langage C++", "Langage Java", "Langage Java (API)", "Système alternatif", "Linux", "Modélisation 3D", "Blender", "Maya", "Favoris", "Microlua", "Tunetrix du Jour", "Utilisez MacPorts 1", and "Plus >".
- Corps (Central Content):** The main content area is titled "En-tête". It contains several sections:
 - "Bienvenue sur le Site du Zéro": A welcome message.
 - "Que propose le Site du Zéro ?": A section listing "Apprenez à maîtriser votre ordinateur", "Echangez avec la communauté", and "Informez-vous de l'actualité".
 - "Derniers tutoriels": A list of recent tutorials including "Un site dynamique avec PHP !", "Gestion de verrou MySQL", and "Apprendre à utiliser QScintilla".
 - "Dernières news": A list of recent news items including "Microsoft contraint de laisser le choix du navigateur par défaut dans Windows", "IE et Windows", and "Google Chrome dans le métro parisien !".
- Pied de page (Footer):** At the bottom, there is a footer section titled "Corps" which contains the text "Dernières news".

Le problème

Jusqu'ici, vous étiez condamnés à copier sur chaque page à l'identique :

- L'en-tête
- Le menu
- Le pied de page

Cela donnait du code lourd et répétitif sur toutes les pages !

Regardez le code d'exemple ci-dessous qui représente une page web (appelons-là `index.php`) avec en-tête, menus et pied de page :

Code : HTML

```
<!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>Mon super site</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>

    <body>

        <!-- L'en-tête -->

        <div id="en_tete">

        </div>

        <!-- Les menus -->

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

        <!-- Le corps -->

        <div id="corps">
            <h1>Mon super site</h1>

            <p>
                Bienvenue sur mon super site !<br />
                Vous allez adorer ici, c'est un site génial qui va
                parler de... heu... Je cherche encore un peu le thème de mon site :-)
            </p>
        </div>

        <!-- Le pied de page -->

        <div id="pied_de_page">
```

```
<p>Copyright moi, tous droits réservés</p>
</div>

</body>
</html>
```

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

La solution

En PHP, nous pouvons facilement insérer d'autres pages (ou morceaux de pages) à l'intérieur d'une page.

Le principe de fonctionnement des *inclusions* en PHP est plutôt simple à comprendre. Vous avez un site web composé de disons 20 pages. Sur chaque page, il y a un menu, toujours le même. Pourquoi ne pas écrire ce menu (et seulement lui) une seule fois dans une page menu.php ?

En PHP, vous allez pouvoir inclure votre menu sur toutes vos pages. Lorsque vous voudrez modifier votre menu vous n'aurez qu'à modifier menu.php et l'ensemble des pages de votre site web sera automatiquement mis à jour !

La pratique

Comme je vous le disais, je vous propose de créer un nouveau fichier PHP et d'y insérer uniquement le code XHTML correspondant à votre menu, comme ceci :

Code : PHP

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

Faites de même pour une page `entete.php` et une page `pied_de_page.php` au besoin pour votre site.



Mais... la page `menu.php` ne contiendra pas le moindre code PHP... c'est normal ?

Une page dont l'extension est `.php` peut très bien ne contenir aucune balise PHP (même si c'est plutôt rare). Dans ce cas, cela redevient une page XHTML classique qui n'est pas modifiée avant l'envoi.

En théorie, vous pourriez très bien enregistrer votre page avec l'extension `.html` : `menu.html`. Néanmoins, afin d'éviter de mélanger des pages `.php` et `.html` sur votre site, je vous recommande de travailler uniquement avec l'extension `.php` à partir d'aujourd'hui.

Maintenant que vos "morceaux de pages" sont prêts, reprenez les pages de votre site, par exemple la page d'accueil nommée `index.php`. Remplacez le menu par le code PHP suivant :

Code : PHP

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

Cette instruction demande à l'ordinateur : "Insère ici le contenu de la page `menu.php`".

Vous noterez que, contrairement à `echo`, j'ai ici placé des parenthèses autour des guillemets. Il faut dire que `echo` était un peu une exception. Dorénavant vous verrez souvent des parenthèses.

`include` est en réalité une structure de langage particulière, comme `echo`, et peut donc s'utiliser avec ou sans parenthèses.

Pour le moment nous débutons, donc nous nous contenterons de faire comme cela sans trop rentrer dans les détails pour ne pas nous brûler les ailes. ☺

Si nous reprenons le code que nous avons vu tout à l'heure et que nous remplaçons chaque code répétitif par un `include`, cela donne le code source suivant :

Code : PHP

```
<!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>Mon super site</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
    </head>

    <body>

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

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

        <div id="corps">
            <h1>Mon super site</h1>

            <p>
                Bienvenue sur mon super site !<br />
                Vous allez adorer ici, c'est un site génial qui va
                parler de... heu... Je cherche encore un peu le thème de mon site :-)
            </p>
        </div>

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

    </body>
</html>
```

 Ce code suppose que votre page `index.php` et celles qui sont incluses (comme `menu.php`) sont dans le même dossier. Si le menu était dans un sous-dossier appelé `includes`, il aurait fallu écrire :

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

C'est le même principe que pour les liens relatifs, que vous connaissez déjà dans le langage XHTML.

Nous avons vu que la page PHP était *générée*, donc la question que vous devez vous poser est : que reçoit le visiteur ? Eh bien il reçoit exactement le même code qu'avant ! 

Ce schéma vous aidera à comprendre comment les pages sont incluses :



La page finale que reçoit le visiteur est identique à celle que je vous ai montrée au début du chapitre... mais vous, vous avez gagné énormément en flexibilité puisque votre code n'est plus recopié 150 fois inutilement. 😊

Le nombre d'include par page n'est pas limité, par conséquent vous pouvez découper votre code en sous-parties autant que vous le souhaitez !

Comme vous avez pu le constater, tout ça n'est absolument pas sorcier, et pourtant grâce aux inclusions on peut déjà rendre son site bien plus agréable !

Notez qu'il existe une autre façon d'inclure des pages en faisant le raisonnement inverse : on crée une page qui contient tout (en-tête, menus, pied de page) sauf le corps, et on inclut le corps de la page.

Pour réaliser cela cependant, il faut savoir manipuler des notions que nous ne connaissons pas pour le moment... et il y a des risques de provoquer des erreurs dangereuses pour votre site (ça fait peur hein 😱). Nous n'en parlerons donc pas pour le moment.

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 quelque chose qui va nous simplifier la vie. Sans les variables, vous n'irez pas bien loin. 🎂

Les variables nous permettent de retenir temporairement des informations en mémoire. Avec elles, nous allons pouvoir par exemple retenir le pseudonyme du visiteur, effectuer des calculs et bien d'autres choses !

Qu'est-ce qu'une variable ?

Rien qu'avec leur nom, vous devez vous dire que c'est quelque chose qui change tout le temps. En effet, le propre d'une variable c'est de pouvoir *varier* (lapalissade 😊). Mais qu'est-ce que c'est concrètement ?

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

C'est à vous de créer des variables. Vous en créez quand vous en avez besoin pour retenir des informations.

Un nom et une valeur

Une variable est toujours constituée de deux éléments :

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

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

On peut modifier quand on veut la valeur de cette variable, faire des opérations dessus, 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*
- *Merci !*

Vous allez voir que ces petites bêtises, 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 !".

Les différents types de variables

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

- **Les chaînes de caractères (string)** : les *chaînes de caractères* sont le nom informatique qu'on donne au texte. Tout texte est appelé chaîne de caractères. En PHP, ce type de données a un nom : string. On peut stocker des textes courts comme très longs au besoin.
Exemple : "Je suis un texte". Une chaîne de caractères est habituellement écrite entre guillemets ou entre apostrophes (on parle de guillemets simples) : 'Je suis un texte'. Les deux fonctionnent mais il y a une petite différence que l'on va découvrir plus loin.
- **Les nombres entiers (int)** : ce sont les nombres du type 1, 2, 3, 4, etc. On compte aussi parmi eux les nombres relatifs : -1, -2, -3...
Exemple : 42
- **Les nombres décimaux (float)** : ce sont les nombres à virgule, comme 14,738. On peut stocker de nombreux chiffres après la virgule, ce qui devrait convenir pour la plupart des usages que vous en ferez. Attention, les nombres doivent être écrits avec un point au lieu de la virgule (c'est la notation anglaise).
Exemple : 14.738
- **Les booléens (bool)** : c'est un type très important qui permet de stocker soit vrai soit faux. Cela permet de retenir si une information est vraie ou fausse. On les utilise très fréquemment. On écrit true pour vrai, et false pour faux.
Exemple : true

- **Rien (NULL)** : aussi bizarre que cela puisse paraître, on a parfois besoin de dire qu'une variable ne contient rien. Rien du tout. On indique donc qu'elle vaut NULL. Ce n'est pas vraiment un type de données, mais plutôt *l'absence* de type.

En résumé, voici ce qu'il faut retenir des différents types d'informations qu'est capable de stocker PHP dans les variables :

Type de données	Exemple de valeur
string	"Du texte "
int	42
float	14.738
bool	true  false 
NULL	

Cela devrait vous donner une idée de tout ce qu'est capable de stocker PHP en mémoire. Ces types suffiront pour la création de notre site !

Maintenant, rentrons dans le concret. Comment créer une variable et comment afficher ce qu'elle contient ?

Affecter une valeur à une variable

Premières manipulations de variables

Je vous propose de commencer par regarder ce code d'exemple :

Code : PHP

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

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

- Son nom est `age_du_visiteur`
- Sa valeur est 17



Notez qu'on ne peut pas mettre d'espaces dans un nom de variable. A la place, utilisez un underscore _ (c'est le symbole sous le chiffre 8 sur un clavier AZERTY).

Évitez aussi les accents, les cédilles et tout autre symbole pour le nom. PHP ne les apprécie pas trop...

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

- D'abord, on écrit le symbole Dollar (`$`) : il précède toujours le nom d'une variable. C'est comme un signe de reconnaissance si vous préférez : ça permet de dire à PHP "J'utilise une variable". Vous reconnaîtrez toujours qu'il y a une variable par la présence du symbole Dollar (`$`).
- Ensuite, il y a le signe Egal (`=`) : celui-là c'est logique, c'est pour dire que `$age_du_visiteur` est égal à...
- A la suite, il y a la valeur de la variable, ici 17.
- Enfin, il y a l'incontournable 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.

Supposons maintenant que l'on écrive ceci :

Code : PHP

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

Que se passera-t-il ? La variable `$age_du_visiteur` va être créée et prendre pour valeur, dans l'ordre : 17, 23, puis 55. Tout cela va très vite, l'ordinateur étant très rapide vous n'aurez pas le temps de dire "ouf" que tout ce code PHP aura été exécuté.

Comme tout à l'heure, rien ne s'affiche. Seulement, quelque part dans la mémoire de l'ordinateur, une petite zone nommée `age_du_visiteur` vient de prendre la valeur 17, puis 23, puis 55.

Utiliser les types de données

Vous vous souvenez des types de données dont je vous ai parlé il y a quelques minutes ? Les `string`, `int`, `float`... Voici un exemple de variable pour chacun de ces types.

Le type `string` (chaîne de caractères)

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

Voici 2 exemples, l'un avec des guillemets simples et l'autre avec des guillemets doubles :

Code : PHP

```
<?php  
$nom_du_visiteur = "Mateo21";  
$nom_du_visiteur = 'Mateo21';  
?>
```

Attention petit piège : si vous voulez insérer un guillemet simple alors que le texte est entouré de guillemets simples, il faut l'échapper comme on l'a vu précédemment en écrivant un antislash devant. De même pour les guillemets doubles. Voici un exemple pour bien comprendre :

Code : PHP

```
<?php  
$variable = "Mon \"nom\" est Mateo21";  
$variable = 'Je m\'appelle Mateo21';  
?>
```

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

Vous pouvez en revanche insérer sans problème des guillemets simples au milieu de guillemets doubles et inversement :

Code : PHP

```
<?php  
$variable = 'Mon "nom" est Mateo21';  
$variable = "Je m'appelle Mateo21";  
?>
```

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

Le type `int` (nombre entier)

On vient de l'utiliser pour nos exemples précédents. Il suffit tout simplement d'écrire le nombre que vous voulez stocker, sans guillemets.

Code : PHP

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

Le type float (nombre décimal)

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

Code : PHP

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

Le type bool (booléen)

Pour dire si une variable vaut vrai ou faux, vous devez écrire le mot `true` ou `false` sans guillemets autour (ce n'est pas une chaîne de caractères !). Je vous conseille de bien choisir le nom de votre variable pour que l'on comprenne ce que ça signifie. Voyez vous-mêmes :

Code : PHP

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

Une variable vide avec NULL

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

Code : PHP

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

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

Afficher et concaténer des variables

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

Afficher le contenu d'une variable

Vous vous souvenez que l'on peut afficher du texte avec `echo`? On peut aussi s'en servir pour afficher la valeur d'une variable!

Code : PHP

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

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

 Au fait, on ne doit pas mettre de guillemets après le `echo` comme tu nous as appris ? 

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

Créez un fichier PHP avec ce code source pour le tester. Inutile de mettre tout le code XHTML autour, ce n'est pas grave ce ne sera pas une "vraie" page XHTML valide mais c'est bien suffisant pour nos tests. Vous devriez voir le résultat s'afficher sur un fond blanc dans votre navigateur :

17

Le nombre contenu à l'intérieur de la variable s'affiche dans la page (ici 17).

La concaténation

Non, ce n'est pas une insulte. 

Cela signifie *assemblage*.

En fait, écrire 17 tout seul comme on l'a fait n'est pas très parlant. On aimerait écrire du texte autour pour dire : "Le visiteur a 17 ans". La concaténation est justement un moyen d'assembler du texte et des variables.

Comment faire cela ? Les petits malins auront l'idée d'écrire 3 instructions `echo`:

Code : PHP

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

Vous pouvez tester, ça fonctionne :

Le visiteur a 17 ans

Mais il y a plus malin. On peut tout faire sur une ligne. Pour cela, il y a 2 méthodes et c'est justement maintenant que le fait d'utiliser des guillemets simples ou doubles va faire la différence.

Concaténer avec des guillemets doubles

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

Concrètement, essayez ce code :

Code : PHP

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

Ca affiche : Le visiteur a 17 ans. Ca fonctionne bien. 😊

En effet, lorsque vous utilisez des guillemets doubles, les variables qui se trouvent à l'intérieur sont analysées et remplacées par leur vraie valeur. Ca a le mérite d'être une solution facile à utiliser mais je vous recommande plutôt la solution qu'on va voir avec des guillemets simples.

Concaténer avec des guillemets simples

Si vous écrivez le code précédent entre guillemets simples, vous allez avoir une drôle de surprise :

Code : PHP

```
<?php  
$age_du_visiteur = 17;  
echo 'Le visiteur a $age_du_visiteur ans'; // Ne marche pas  
?>
```

Ca affiche : Le visiteur a \$age_du_visiteur ans.



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

Eh bien si ! Mais cette fois, il va falloir écrire la variable en dehors des guillemets et séparer les éléments entre eux à l'aide d'un point. Regardez :

Code : PHP

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

```
echo 'Le visiteur a ' . $age_du_visiteur . ' ans';
?>
```

Cette fois, ça affiche bien comme on voulait : Le visiteur a 17 ans

Ca a l'air bien plus compliqué, mais en fait c'est cette méthode qu'utilisent la plupart des programmeurs expérimentés en PHP. En effet, le code est plus lisible, on repère bien la variable alors que tout à l'heure elle était comme "noyée" dans le texte. D'autre part, votre éditeur de texte devrait vous colorier la variable ce qu'il ne faisait pas pour le code précédent.



Il faut noter aussi que cette méthode d'écriture est un chouilla plus rapide car PHP voit de suite où se trouve la variable et n'a pas besoin de la chercher au milieu du texte.

Dorénavant, j'écrirai toutes mes chaînes de caractères entre guillemets simples (à de rares exceptions près) et j'utiliserai la seconde méthode de concaténation qu'on vient de voir. Prenez le temps de vous habituer à l'utiliser et cela finira par devenir complètement naturel pour vous. 😊

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 ? 😊

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

Les opérations de base : addition, soustraction...

Voici les signes à connaître pour faire les 4 opérations de base (vous les trouverez sur votre pavé numérique, à droite du clavier) :

Symbol	Signification
+	Addition
-	Soustraction
*	Multiplication
/	Division

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

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

C'est de la pure logique, je ne peux rien vous dire de plus.

Si vous avez compris ces bouts de code, vous avez tout compris. 😊

Le modulo

Il est possible de faire un autre type d'opération un peu moins connu : le **modulo**. Cela représente le reste de la division entière.

Par exemple, $6 / 3 = 2$ et il n'y a pas de reste. En revanche, $7 / 3 = 2$ (car le nombre 3 "rentre" 2 fois dans le nombre 7) et il reste 1. Vous avez fait ce type de calcul à l'école primaire, souvenez-vous !

Le modulo permet justement de récupérer ce "reste". Pour faire un calcul avec un modulo, on utilise le symbole %.

Code : PHP

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

Et les autres opérations ?

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

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 (ce sera facile 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 conditions

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

A quoi servent les conditions ? 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 sont vraiment la base pour rendre votre site dynamique, c'est à dire d'afficher des choses **differentes** en fonction du visiteur, de l'heure de la journée, de la date, etc.

Voilà pourquoi ce chapitre est si important !

Allez, on y va ! 

La structure de base : If... Else

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

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

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 !

Symbol	Signification
<code>==</code>	Est égal à
<code>></code>	Est supérieur à
<code><</code>	Est inférieur à
<code>>=</code>	Est supérieur ou égal à
<code><=</code>	Est inférieur ou égal à
<code>!=</code>	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, 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  
$age = 8;  
  
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  
$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  
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 à 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 nouveauté 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) ...**

Le cas des booléens

Si vous regardez bien le dernier code source (avec \$autorisation_entrer), vous ne 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  
if ($autorisation_entrer == true)  
{  
    echo "Bienvenue petit Zéro :o)";  
}  
elseif ($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
if ($autorisation_entrer)
{
    echo "Bienvenue petit Zéro :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 :

Code : PHP

```
<?php
if (! $autorisation_entrer)
{
}
?>
```

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

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	Symbolé équivalent
---------	---------------	--------------------

AND	Et	&&
OR	Ou	



Le symbole équivalent pour OR est constitué 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).

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
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 superhéros préféré. Sinon, si c'est une fille dont l'âge est inférieur ou égal à 12 ans, on l'envoie gentiment balader (hum hum, m'accusez pas de sexismehin, 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
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 ?
";
}
?>
```

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

Code : 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 (" ou ').

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 structures à base de `if... elseif... else` que je viens de vous montrer suffisent pour traiter n'importe quelle condition.



Mais alors pourquoi se compliquer la vie avec une autre structure ? 😐

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

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

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

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

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

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

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

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

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

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

Code : 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.). 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 si la valeur ne correspond à aucun `case`.

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 précédent, 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. 

Les ternaires : des conditions condensées

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

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

- On teste la valeur d'une variable dans une condition.
- On affecte une valeur à une variable selon que la condition est vraie ou non.

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

Code : PHP

```
<?php  
$age = 24;  
  
if ($age >= 18)  
{  
    $majeur = true;  
}  
else  
{  
    $majeur = false;  
}  
?>
```

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

Code : PHP

```
<?php  
$age = 24;  
  
$majeur = ($age >= 18) ? true : false;  
?>
```

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

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

C'est un peu tordu mais ça marche. 😊

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

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, car ces connaissances de base vont vous être indispensables par la suite !

Les boucles

Dans la série des éléments de base de PHP à connaître absolument, voici les boucles ! Demander à l'ordinateur de répéter des instructions, ça il sait faire (et en plus il ne bronche jamais) !

Imaginez par exemple que vous êtes en train de créer le forum de votre site. Sur une page, on affiche par exemple une trentaine de messages. Il serait bien trop long et répétitif de dire "Affiche le message 1 et le nom de son auteur", "Affiche le message 2 et le nom de son auteur", "Affiche le message 3 et le nom de son auteur", etc. Pour éviter d'avoir à faire cela, on peut utiliser un système de boucle qui nous permettra de dire une seule fois : "Affiche 30 messages et le nom de leur auteur respectif à chaque fois".

Bien entendu, nous n'allons pas pouvoir apprendre à créer le forum de votre site dans ce chapitre (il est encore trop tôt). Néanmoins, prenez bien le temps de comprendre le fonctionnement des boucles car nous en aurons besoin tout le long de ce cours. Ce n'est pas bien compliqué vous allez voir !

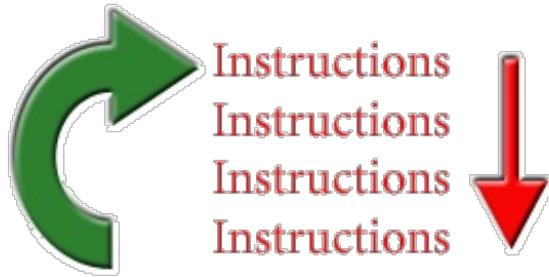
Une boucle simple : While

Qu'est-ce qu'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 beaucoup 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 si vous voulez présenter le principe dans un schéma :



Voilà ce qui se passe dans une boucle :

1. Comme d'habitude, les instructions sont d'abord exécutées dans l'ordre, de haut en bas (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 == true)
{
    // instructions à exécuter dans la boucle
}
?>
```

`While` peut se traduire par "Tant que". Ici, on demande à PHP : *TANT QUE \$continuer_boucle est vrai, 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. 😊

Ce n'est pas beaucoup plus compliqué que ça, il n'y a guère plus de choses à savoir. Cependant, 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
$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 ou égal à 100*

Dans cette boucle, il y a 2 instructions :

- Le `echo`, qui permet d'afficher du texte en PHP. A noter qu'il y a une balise HTML `
` à la fin : cela permet d'aller à la ligne. Vu que vous connaissez le HTML, ça n'a rien de surprenant : chaque phrase sera écrite sur une seule ligne.
- 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 a atteint 100 puis qu'elle passe à 101, la boucle s'arrête sans afficher l'echo. 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 fausse au moins une fois. Si elle ne l'est jamais, alors la boucle s'exécutera à l'infini !

PHP refuse normalement de travailler plus d'une quinzaine de secondes. Il s'arrêtera tout seul s'il voit que son travail dure trop longtemps et affichera un message d'erreur.

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

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

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

Je vous demande 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
$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 information, l'astuce que je vous avais donnée dans le chapitre sur les conditions marche aussi ici : vous pouvez fermer la balise PHP ?>, écrire du texte en HTML, puis réouvrir la balise PHP <?php. Cela vous évite d'utiliser une ou plusieurs instructions echo au milieu. On aura l'occasion d'utiliser cette astuce de nombreuses fois dans la suite du cours.



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 ici "compliqué".

`for` est un autre type de boucle, dans une forme un peu plus condensée et plus pratique à écrire. Cela fait que `for` est assez fréquemment utilisé dans la pratique.

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

Alors, comment ça marche un `for`? Ça 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
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 à chaque tour de boucle.

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



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

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

Croyez-moi, les boucles vont vraiment nous faire gagner un temps fou !

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 l'étude de la base de données un peu

plus loin dans ce cours.

Les fonctions

En PHP, on n'aime pas avoir à répéter le même code plusieurs fois. Pour répondre à ce problème, nous avons découvert les boucles qui permettent d'exécuter des instructions plusieurs fois. Nous allons ici découvrir un autre type de structure très important à connaître : les fonctions.

Comme les boucles, les fonctions permettent d'éviter d'avoir à répéter du code PHP que l'on utilise souvent. Mais alors que les boucles sont de bêtes machines tout juste capables de répéter 200 fois la même chose, les fonctions sont des robots "intelligents" qui s'adaptent en fonction de ce que vous voulez faire et qui automatisent grandement la plupart des tâches courantes.

Qu'est-ce qu'une fonction ?

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

Imaginez que les fonctions sont des robots comme celui-ci :



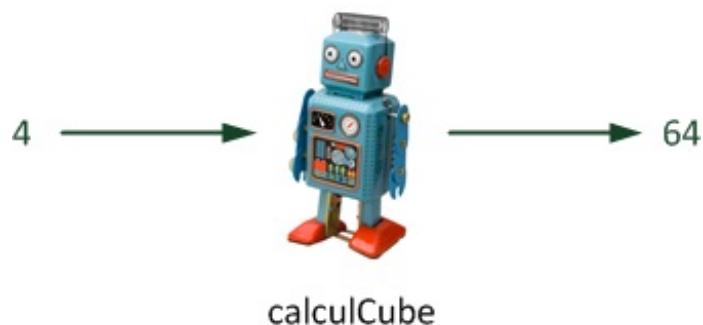
Vous ne savez pas ce qui se passe à l'intérieur de ce robot, mais vous pouvez appuyer sur un bouton pour lui demander de faire quelque chose de précis. Avec les fonctions, c'est le même principe !

Dialogue avec une fonction

Voici le genre de dialogue qu'on peut avoir avec 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 cm³.

On donne en *entrée* à la fonction un **paramètre** sur lequel elle va faire des calculs (ici la longueur de l'arête : 4) et la fonction nous retourne en *sortie* le résultat : 64.



Grâce à la fonction, vous n'avez pas eu besoin de vous souvenir comment on calcule le volume d'un cube. Bon ici c'était assez simple (il suffisait de faire $4 \times 4 \times 4$), mais vous serez souvent amené à faire des opérations de plus en plus complexes et les fonctions vous permettront de ne pas avoir à vous soucier des détails des calculs.

Si vous aviez eu à faire le calcul du volume du cube une seule fois, vous auriez pu chercher dans un livre comment on le calcule (si vous ne vous en souveniez pas 😊) et écrire à la main le calcul. Mais si vous aviez à le faire 5 fois ? 10 fois ? 100 fois ?



En quoi est-ce que c'est différent des boucles ? Avec les boucles on peut faire répéter le même code plusieurs fois aussi !

Oui, mais les fonctions sont capables de s'adapter en fonction des informations que vous leur envoyez. Par exemple dans notre

cas, il suffit d'envoyer la longueur de l'arête du cube à notre fonction pour qu'elle nous retourne le résultat. Ces informations que l'on donne en *entrée* à la fonction sont appelées **paramètres** (un mot à connaître !).



Les fonctions ne servent qu'à faire des calculs mathématiques ? Je veux juste créer un site web, pas faire des maths !

J'ai choisi un exemple mathématique ici parce que je le trouve simple et parlant, mais dans la pratique on ne passe pas son temps à calculer des logarithmes et des exponentielles quand on crée un site web, je suis d'accord. 😊

Concrètement, les fonctions peuvent permettre de récupérer des informations comme la date et l'heure actuelles, de crypter des données, d'envoyer des emails, de faire des recherches dans du texte, et bien d'autres choses encore !

Les fonctions en PHP

Nous avons jusqu'ici imaginé le dialogue avec une fonction représentée par un robot, ce n'est pas très sérieux. 😊 Retournons aux choses sérieuses et au concret.

Appeler une fonction

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

Code : PHP

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



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

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

Certaines fonctions peuvent fonctionner sans paramètres, mais elles sont assez rares. Dans le cas de `calculCube`, ça n'a pas de sens de l'appeler sans lui donner la longueur de l'arête du cube pour faire le calcul !

Si on veut lui envoyer un paramètre (que ce soit un nombre, une chaîne de caractère, un booléen), il faut l'écrire entre les parenthèses :

Code : PHP

```
<?php  
calculCube (4);  
?>
```

Ainsi, `calculCube` saura qu'elle doit travailler avec le nombre 4.

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

Code : PHP

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

Cette fonction recevra 4 paramètres : 17, le texte "Vert", le booléen vrai et le nombre 41,7.

Récupérer la valeur de retour de la fonction

Maintenant que nous savons appeler une fonction et même lui envoyer plusieurs paramètres, il faut récupérer ce qu'elle nous retourne si toutefois elle retourne quelque chose. Il y a en effet 2 types de fonctions :

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

Si la fonction ne retourne aucune valeur, il n'y a rien de plus à faire que dans les codes précédents. La fonction est appelée, fait son travail et on ne lui demande rien de plus.

En revanche, si la fonction retourne une valeur, comme ça devrait être le cas pour `calculCube`, on la récupère dans une variable comme ceci :

Code : PHP

```
<?php  
$volume = calculCube(4);  
?>
```

Sur une ligne comme celle-ci, il se passe en fait 2 choses, dans l'ordre de droite à gauche :

1. La fonction `calculCube` est appelée avec le paramètre 4.
2. Le résultat renvoyé par la fonction (lorsqu'elle a terminé) est stocké dans la variable `$volume`.

La variable `$volume` aura donc pour valeur 64 après l'exécution de cette ligne de code !



Bon à savoir : on peut envoyer en entrée plusieurs paramètres à une fonction comme on l'a vu, mais en revanche la fonction ne peut renvoyer en retour qu'une seule valeur. Il existe un moyen de contourner cette limitation en combinant des variables au sein d'un tableau de variables (appelé *array*) dont on parlera dans un prochain chapitre.

Les fonctions prêtes à l'emploi de PHP

PHP propose des centaines et des centaines de fonctions prêtes à l'emploi. La documentation de PHP, sur le site web officiel, répertorie toutes les fonctions classées par catégories.

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 car elles couvrent la quasi-totalité de nos besoins. J'ai en fait remarqué que, pratiquement à chaque fois que je m'apprétais à écrire une fonction, celle-ci existait déjà.

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.

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

Nous allons ici découvrir rapidement quelques fonctions pour vous habituer à les utiliser. Nous ne pourrons jamais toutes les passer en revue (j'ai dit qu'il y en avait des centaines et des centaines !) mais avec l'expérience de ces premières fonctions et la documentation de PHP, vous n'aurez aucun mal à aller plus loin tous seuls. 😊

Nous allons voir quelques fonctions qui effectuent des modifications sur des chaînes de caractères ainsi qu'une fonction qui permet de récupérer la date. Ce sont seulement des exemples destinés à vous habituer à utiliser des fonctions.

Traitement des chaînes de caractères

De nombreuses fonctions permettent de manipuler le texte. En voici quelques-unes qui vont vous montrer l'intérêt des fonctions.

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

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

Essayez !



Méfiez-vous, il se peut que le nombre de caractères soit parfois inexact dû à un bug de PHP dans la gestion des encodages de caractères. Cela sera corrigé dans les prochaines versions de PHP.

str_replace

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

Code : PHP

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

Essayez !

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

Essayez !

strtolower

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

Code : PHP

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

[Essayez !](#)

A noter qu'il existe `strtoupper` qui fait la même chose en sens inverse : minuscules => majuscules.

Récupérer la date

Nous allons découvrir la fonction qui renvoie l'heure et la date. Il s'agit de `date` (un nom facile à retenir, avouez !). Cette fonction peut donner beaucoup d'informations. Voici les principaux paramètres à connaître :



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

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

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

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;  
?>
```

[Essayez !](#)

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

Normalement, quand vous avez cliqué sur "Essayez !", 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 😊

Créer ses propres fonctions

Bien que PHP propose des centaines et des centaines de fonctions (j'insiste dessus, mais il faut dire qu'il y en a tellement !), parfois il n'y aura pas ce que vous cherchez et il faudra écrire vous-même la fonction. C'est une façon pratique d'étendre les possibilités offertes par PHP.

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

Nous allons découvrir la création de fonctions à travers 2 exemples :

- Afficher un message de bienvenue en fonction du nom
- Calculer le volume d'un cône

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
$nom = 'Sandra';
echo 'Bonjour, ' . $nom . ' !<br />';

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

$nom = 'Claude';
echo 'Bonjour, ' . $nom . ' !<br />';
?>
```

[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
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...). Elles ne font rien de particulier, mais elles disent à PHP : "Une fonction DireBonjour existe maintenant".

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. C'est le **paramètre** dont a besoin la fonction pour travailler, afin qu'elle sache à qui elle doit dire bonjour dans notre cas. Notre fonction doit forcément être appelée avec un paramètre (le nom) sans quoi elle ne pourra pas travailler.

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

Ensuite, vous repérez des accolades. 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 qu'en pratique une fonction contient plus d'instructions que cela.

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 :

Code : PHP

```
<?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)

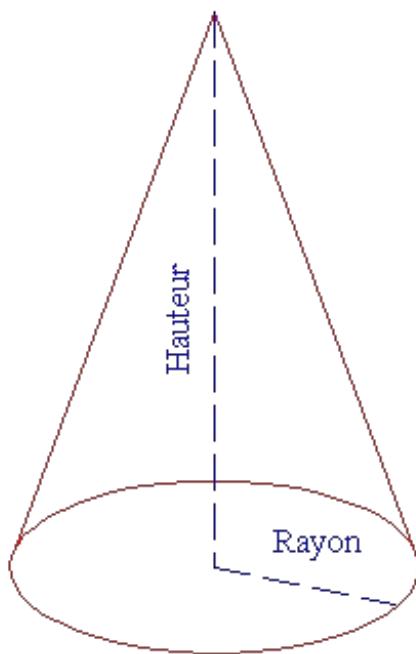
 Un conseil pour que vous vous entraîniez sur les fonctions : basez-vous sur mes exemples et essayez de les retoucher petit à petit vous-mêmes pour voir ce que ça donne. Il peut y avoir des fonctions très simples comme des fonctions très compliquées, alors allez-y prudemment.

2ème exemple : calculer le volume d'un cône

Allez on passe à la vitesse supérieure. La fonction `DireBonjour` que l'on a créée ne renvoyait aucune valeur, elle se contentait d'afficher des actions (afficher un texte dans le cas présent). Maintenant, nous allons créer une fonction qui renvoie une valeur.

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.

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 ne 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 !

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
// 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 !).

La fonction renvoie une valeur, donc on doit récupérer cette valeur dans une variable :

Code : PHP

```
<?php  
$volume = VolumeCone(3, 1);  
?>
```

Ensuite, on peut afficher ce que contient la variable à l'aide d'une instruction `echo`.

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.

Vous en savez suffisamment sur les fonctions ! Il y aurait d'autres choses à apprendre mais vous connaissez les bases.

D'ailleurs en parlant de bases, vous êtes de moins en moins un débutant total en PHP, nous avons bientôt fini de couvrir les bases !



Les tableaux

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. Cependant, ils vont très rapidement nous devenir indispensables et vous *devez* bien comprendre leur fonctionnement. Si vous y parvenez, nous aurons fait le tour des bases du PHP et vous serez fin prêts pour la suite, qui s'annonce concrète et passionnante.

Mais trève de bavardages, à l'abordaaaage ! 

Les deux types de tableaux

Un tableau (*array*) est une variable. Mais une variable un peu spéciale.

Reprenez. 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 de nombreuses informations dans une seule variable (bien plus que "Nicole") grâce aux tableaux. On distingue deux types de tableaux :

- Les tableaux numérotés
- Les tableaux associatifs

Les tableaux numérotés

Ces tableaux sont très simples à imaginer. Regardez par exemple ce tableau, contenu de la variable \$prenoms :

Clé	Valeur
0	François
1	Michel
2	Nicole
3	Véronique
4	Benoît
...	...

\$prenoms est un **array** : c'est ce qu'on 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, c'est-à-dire que chaque case est identifiée par un numéro. Ce numéro est appelé **clé**.



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...

Construire un tableau numéroté

Pour créer un tableau numéroté en PHP, on utilise généralement la fonction `array`.

Cet exemple vous montre comment créer l'array `$prenoms` :

Code : 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.

Vous pouvez aussi créer manuellement le tableau case par case :

Code : PHP

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

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

Code : PHP

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

Afficher un tableau numéroté

Pour afficher un élément, il faut donner sa position entre crochets après `$prenoms`. Cela revient à dire à PHP :

Affiche-moi le contenu de \$prenoms dans la case n°1

Pour faire cela en PHP, il faut écrire le nom de la variable, suivi du numéro entre crochets. Pour afficher "Michel", on doit donc écrire :

Code : PHP

```
<?php
echo $prenoms[1];
?>
```

C'est tout bête, du temps que vous n'oubliez pas que Michel est en seconde position et donc qu'il a le numéro 1 (étant donné

qu'on commence à compter à partir de 0). 😊



Si vous oubliez de mettre les crochets, ça ne marchera pas (ça affichera juste "Array"...). 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, sinon PHP ne sait pas quoi récupérer.

Les tableaux associatifs

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

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 est le nom, le n°1 le prénom, le n°2 l'adresse ?... C'est là que deviennent utiles les tableaux associatifs.

Construire un tableau associatif

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



Note importante : il n'y a ici 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 écrit une flèche (`=>`) pour dire "associé à". Par exemple, on dit "ville associé à Marseille".

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

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

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

Code : PHP

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

Afficher un tableau associatif

Pour afficher un élément, il suffit d'indiquer le nom de cet élément entre crochets, ainsi qu'entre guillemets ou apostrophes puisque l'étiquette du tableau associatif est un texte.

Par exemple, pour extraire la ville, on devra taper :

Code : PHP

```
<?php  
echo $coordonnees['ville'];  
?>
```

Essayer !



Quand utiliser un array numéroté et quand utiliser un array associatif ?

Comme vous l'avez vu dans mes exemples, ils ne servent pas à stocker la même chose :

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

Parcourir un tableau

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

- La boucle `for`
- La boucle `foreach`
- La fonction `print_r` (utilisée principalement pour le déboggage)

La boucle `for`

Il est très simple de parcourir un tableau numéroté avec une boucle `for`. En effet, puisqu'il est numéroté à partir de 0, on peut faire une boucle `for` qui incrémentera un compteur à partir de 0 :

Code : 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] . '<br />'; // affichera $prenoms[0],
$prenoms[1] etc...
}
?>
```

Essayer !

Quand on écrit `$prenoms[$numero]`, la variable `$numero` est d'abord remplacée par sa valeur. Par exemple, si `$numero` vaut 2, alors cela signifie qu'on cherche à obtenir ce que contient `$prenoms[2]`, c'est-à-dire... Nicole, bravo vous avez compris. 😊

La boucle `foreach`

La boucle `for` a beau fonctionner, on peut utiliser un type de boucle plus adapté aux tableaux qu'on n'a pas vu jusqu'ici : `foreach`. C'est une sorte de boucle `for` spécialisée dans les tableaux.

`foreach` va passer en revue chaque ligne du tableau, et lors de chaque passage, elle va mettre la valeur de cette ligne dans une variable temporaire (par exemple `$element`).

Je parle chinois ? Ok, alors regardez :

Code : PHP

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

foreach($prenoms as $element)
```

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

Essayer !

C'est le même code que tout à l'heure mais basé ici sur une boucle `foreach`. A chaque tour de boucle, la valeur de l'élément suivant est mise dans la variable `$element`. On peut donc utiliser `$element` uniquement à l'intérieur de la boucle afin d'afficher l'élément en cours.

L'avantage de `foreach` est qu'il permet aussi de parcourir les tableaux associatifs.

Code : PHP

```
<?php  
$coordonnees = array (  
    'prenom' => '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'utilise 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. Or, on peut aussi récupérer la clé de l'élément. On doit dans ce cas écrire `foreach` comme ceci :

Code : PHP

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

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

- \$cle : elle contiendra la clé de l'élément en cours d'analyse ("prenom", "nom", etc.).
- \$element : il contiendra la valeur de l'élément en cours ("François", "Dupont", etc.).

Testons le fonctionnement avec un exemple :

Code : PHP

```
<?php
$coordonnees = array (
    'prenom' => '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 maintenant dans la boucle la clé ET la valeur.

Et `foreach`, croyez-moi, c'est un truc vraiment pratique ! On aura souvent l'occasion de s'en servir dans nos prochains scripts PHP !

Afficher rapidement un array avec print_r

Parfois, en codant votre site en PHP, vous aurez sous les bras un array et vous voudrez savoir ce qu'il contient, juste pour votre information. Vous pourriez utiliser une boucle `for` ou, mieux, une boucle `foreach`. Mais si vous n'avez pas besoin d'une mise en forme spéciale et que vous voulez juste savoir ce que contient l'array, vous pouvez faire appel à la fonction `print_r`. C'est une sorte de `echo` spécialisé dans les array.

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

Code : PHP

```
<?php
$coordonnees = array (
    'prenom' => '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>` pour avoir un affichage correct.

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

Rechercher dans un tableau

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, basées sur des fonctions PHP :

- `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` qui va parcourir le tableau pour nous et nous dire si le tableau contient cette clé.

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 :

Code : PHP

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

La fonction renvoie 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
$coordonnees = array (
    'prenom' => 'Fran ois',
    'nom' => 'Dupont',
    'adresse' => '3 Rue du Paradis',
    'ville' => 'Marseille');

if (array_key_exists('nom', $coordonnees))
{
    echo 'La cl  "nom" se trouve dans les coordonn es !';
}

if (array_key_exists('pays', $coordonnees))
{
    echo 'La cl  "pays" se trouve dans les coordonn es !';
}

?>
```

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`

Le principe est le même que `array_key_exists`... mais cette fois on recherche dans les valeurs. `in_array` renvoie `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
$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".

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 renvoie la fonction :

- Si elle a trouvé la valeur, `array_search` renvoie 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` renvoie `false`.

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

Code : 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. Il y en a d'autres mais vous connaissez maintenant les principales.

Et voilà ! Bravo vous avez terminé la partie I, vous connaissez les bases de PHP !

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 : Transmettre des données de page en page

Maintenant que vous avez acquis les bases de PHP, nous pouvons nous intéresser à du concret.

Le langage PHP a été conçu pour que vous puissiez transmettre des informations de page en page, au fil de la navigation d'un visiteur sur votre site. C'est notamment ce qui vous permet de retenir son pseudonyme tout au long de sa visite, mais aussi de récupérer et traiter les informations qu'il rentre sur votre site, notamment dans des formulaires.

Grâce à cette partie, vous allez pouvoir commencer à communiquer vraiment avec vos visiteurs !

Transmettre des données avec l'URL

Savez-vous ce qu'est une URL ? Cela signifie *Uniform Resource Locator*, cela sert à représenter une adresse sur le web. Toutes les adresses que vous voyez en haut de votre navigateur, comme <http://www.siteduzero.com>, sont des URL.

Je me doute bien que vous ne passez pas votre temps à regarder ces URL (il y a bien mieux à faire !), mais je suis sûr que vous avez déjà été surpris de voir certaines URL assez longues avec des caractères un peu curieux. Par exemple, après avoir fait une recherche sur Google, la barre d'adresse contient une URL longue qui ressemble à ceci :

<http://www.google.fr/search?rlz=1C1GFR343&q=siteduzero>. Les informations après le point d'interrogation dans cette URL sont des données que l'on fait transiter d'une page à une autre. Nous allons découvrir dans ce chapitre comment cela fonctionne.

Envoyer des paramètres dans l'URL

En introduction, je vous disais que l'URL permettait de transmettre des informations. Comment est-ce que ça fonctionne exactement ?

Former une URL pour envoyer des paramètres

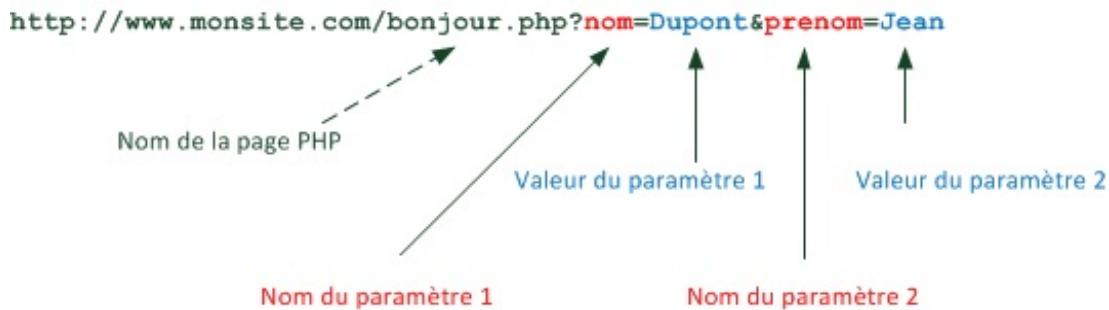
Imaginons que votre site s'appelle monsite.com et que vous avez une page PHP intitulée bonjour.php. Pour accéder à cette page, vous devez aller à l'URL suivante :

<http://www.monsite.com/bonjour.php>

Jusque-là, rien de bien nouveau. Ce que je vous propose d'apprendre à faire, c'est d'*envoyer* des informations à la page bonjour.php. Pour cela, on va ajouter à la fin de l'URL des informations, comme ceci :

<http://www.monsite.com/bonjour.php?nom=Dupont&prenom=Jean>

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



Le point d'interrogation sépare le nom de la page PHP des paramètres. Ensuite, les paramètres s'enchaînent selon la forme `nom=valeur` et sont séparés entre eux par le symbole `&`.



En théorie oui. Il suffit de les séparer par des `&` comme je l'ai fait. On peut donc voir une URL de la forme :

`page.php?param1=valeur1¶m2=valeur2¶m3=valeur3¶m4=valeur4 [...]`

La seule limite est la longueur de l'URL. En général il n'est pas conseillé de dépasser les 256 caractères, mais parfois les navigateurs arrivent à gérer des URL plus longues. Quoiqu'il en soit vous l'aurez compris, on ne peut pas mettre un roman dans l'URL non plus. 😊

Créer un lien avec des paramètres

Maintenant que nous savons cela, nous pouvons créer des liens en HTML d'une page vers une autre qui transmettent des paramètres.

Imaginez que vous ayez 2 fichiers sur votre site :

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

Nous voulons faire un lien de index.php qui amène vers bonjour.php et qui lui transmet des informations dans l'URL.



Pour cela, ouvrez **index.php** (puisque c'est lui qui contiendra le lien) et insérez-y par exemple le code suivant :

Code : PHP

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



Comme vous le voyez, le & dans le code a été remplacé par & dans le lien. Ca n'a rien à voir avec le PHP, simplement en HTML on demande à ce que les & soient écrits & dans le code source. Si vous ne le faites pas, le code ne passera pas la validation W3C.

Ce lien appelle la page bonjour.php et lui envoie 2 paramètres :

- nom : Dupont
- prenom : Jean

Vous avez sûrement deviné ce qu'on essaie de faire ici. On appelle une page bonjour.php qui va dire "Bonjour" à la personne dont le nom et le prénom ont été envoyés en paramètres.

Comment faire dans la page bonjour.php pour récupérer ces informations ? C'est ce que nous allons voir maintenant. 😊

Récupérer les paramètres en PHP

Nous savons maintenant comment former des liens pour envoyer des paramètres vers une autre page. Nous avons pour cela ajouté des paramètres à la fin de l'URL.

Intéressons-nous maintenant à la page qui réceptionne les paramètres. Dans notre exemple, il s'agit de la page `bonjour.php`. Cette page va automatiquement créer un array au nom un peu spécial : `$_GET`. Il s'agit d'un array associatif (vous vous souvenez, nous avons étudié cela il y a peu !) dont les clés correspondent aux noms des paramètres envoyés en URL.

Reprendons notre exemple pour mieux voir comment cela fonctionne. Nous avons fait un lien vers : `bonjour.php?nom=Dupont&prenom=Jean`, cela signifie que nous aurons accès aux variables suivantes :

Nom	Valeur
<code>\$_GET['nom']</code>	Dupont
<code>\$_GET['prenom']</code>	Jean

On peut donc récupérer ces informations, les traiter, les afficher, bref faire ce qu'on veut avec. Essayons pour commencer de les afficher, tout simplement.

Créez un nouveau fichier PHP que vous appellerez `bonjour.php` et placez-y le code suivant :

Code : PHP

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

 Bien sûr, pour une vraie page web il faudrait écrire toutes les informations d'en-tête nécessaires en HTML : le doctype, la balise `<head>`, etc. Ici, je ne mets pas tout ce code pour faire simple. La page ne sera pas très belle (ni valide W3C d'ailleurs) mais ce n'est pas notre problème ici. Nous faisons juste des tests.

Ici, nous affichons le prénom qui a été passé dans l'URL. Bien entendu, nous avons aussi accès au nom. Nous pouvons afficher le nom et le prénom sans problème :

Code : PHP

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

Comme vous le voyez j'en ai profité pour faire une petite concaténation comme nous avons appris à le faire. Ce code que vous voyez là n'est pas bien complexe : nous affichons le contenu de 2 cases de l'array `$_GET`. C'est vraiment très simple et il n'y a rien de nouveau. Tout ce qu'il fallait savoir, c'est qu'on peut retrouver les paramètres envoyés dans l'URL grâce à un array nommé `$_GET`.

Si vous voulez tester le résultat, je vous propose d'essayer en ligne ce que ça donne directement sur le Site du Zéro. Ce lien ouvre la page `index.php` :

[Essayer !](#)

Vous devriez faire le test aussi chez vous pour vous assurer que vous comprenez bien le fonctionnement ! Si besoin est, vous pouvez télécharger mes fichiers `index.php` et `bonjour.php` d'exemple.

[Télécharger les fichiers d'exemple](#)

Ne faites jamais confiance aux données reçues !

Il est impossible de terminer ce chapitre sans que je vous mette en garde contre les **dangers** qui guettent l'apprenti webmaster que vous êtes. Nous allons en effet découvrir qu'il ne faut JAMAIS faire confiance aux variables qui transitent de page en page, comme `$_GET` que nous étudions ici.

Le but de cette section est, je l'avoue, de vous faire peur. Car trop peu de développeurs PHP ont conscience des dangers et des failles de sécurité qu'ils peuvent rencontrer, ce qui risque pourtant de mettre en péril leur site web ! Oui je suis alarmiste, oui je veux que vous ayez (un peu) peur ! Mais rassurez-vous : je vais vous expliquer tout ce qu'il faut savoir pour que ça se passe bien et que vous ne risquiez rien.

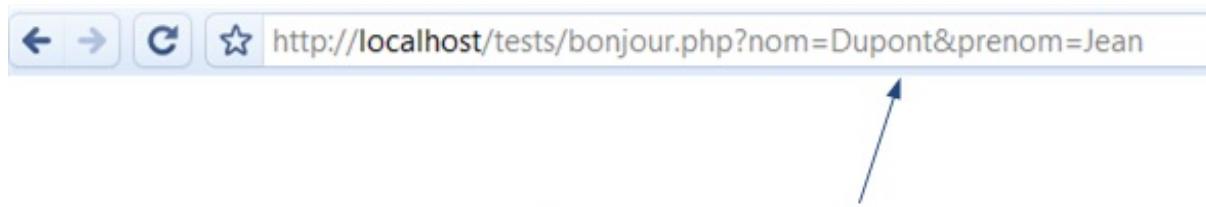
Ce qui compte, c'est que vous ayez conscience très tôt, dès maintenant, des problèmes que vous pourrez rencontrer lorsque vous recevrez des paramètres de l'utilisateur.

Tous les visiteurs peuvent trafiquer les URL

Si vous faites les tests des codes précédents chez vous, vous devriez tomber sur une URL de la forme :

`http://localhost/tests/bonjour.php?nom=Dupont&prenom=Jean`

On vous dit bien "Bonjour Jean Dupont !". Mais si vous êtes un peu bricoleur, vous pouvez vous amuser à changer les paramètres directement dans la barre d'adresse.



N'importe qui peut modifier les paramètres dans la barre d'adresse de son navigateur !

Essayez par exemple de modifier l'adresse pour :

`http://localhost/tests/bonjour.php?nom=Dupont&prenom=Marc`

Comme vous le voyez, ça marche ! N'importe qui peut facilement modifier les URL et y mettre ce qu'il veut : il suffit simplement de changer l'URL dans la barre d'adresse de votre navigateur.



Et alors, quel est le problème ? C'est pas plus mal non, si le visiteur veut adapter l'URL pour qu'on lui dise bonjour avec son vrai nom et son vrai prénom ?

Peut-être, mais cela montre une chose : **on ne peut pas avoir confiance dans les données qu'on reçoit**. N'importe quel visiteur peut les changer. Dans la page `index.php` j'ai fait un lien qui dit bonjour à Jean Dupont, mais la page `bonjour.php` ne va pas forcément afficher "Bonjour Jean Dupont !" puisque n'importe quel visiteur peut s'amuser à modifier l'URL.

Je vous propose de faire des modifications encore plus vicieuses et de voir ce qu'on peut faire pour éviter les problèmes qui en découlent.

Tester la présence d'un paramètre

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

<http://localhost/tests/bonjour.php>

Que va afficher la page `bonjour.php`? Faites le test ! Elle va afficher quelque chose comme :

Code : Console

```
Bonjour
Notice: Undefined index: prenom in C:\wamp\www\tests\bonjour.php on line 9
Notice: Undefined index: nom in C:\wamp\www\tests\bonjour.php on line 9
!
```

Que s'est-il passé ? On a essayé d'afficher la valeur de `$_GET['prenom']` et de `$_GET['nom']` ... Mais comme on vient de les supprimer de l'URL, ces variables n'ont pas été créées et donc elles n'existent pas ! PHP nous avertit qu'on essaie d'utiliser des éléments du tableau qui n'existent pas, d'où les "Undefined index".

Pour résoudre ce problème, on peut faire appel à une fonction un peu spéciale : `isset()`. Cette fonction teste si une variable existe. Nous allons nous en servir pour afficher un message spécifique si le nom ou le prénom sont absents.

Code : PHP

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

Que fait ce code ? Il teste si les variables `$_GET['prenom']` et `$_GET['nom']` existent. Si elles existent, on dit bonjour au visiteur. S'il nous manque une des variables (ou les deux) on affiche un message d'erreur : "Il faut renseigner un nom et un prénom !".

Essayez maintenant d'accéder à la page `bonjour.php` sans les paramètres, vous allez voir qu'on gère bien le cas où le visiteur aurait retiré les paramètres de l'URL. ☺



Est-ce que c'est vraiment la peine de gérer ce cas ? C'est vrai quoi, moi je ne m'amuse jamais à modifier mes URL et mes visiteurs non plus je pense !

Oui, oui, trois fois oui : il faut que vous pensiez à gérer le cas où des paramètres que vous attendiez viendraient à manquer.

Vous vous souvenez de ce que je vous ai dit ? Il ne faut jamais faire confiance à l'utilisateur. Tôt ou tard vous tomberez sur un utilisateur mal intentionné qui essaiera de trafiquer l'URL pour mettre n'importe quoi dans les paramètres. Il faut que votre site soit prêt à gérer le cas.

Dans notre exemple, si on ne gérait pas le cas ça ne faisait rien de bien grave (ça affichait juste des messages d'erreur). Mais lorsque votre site web deviendra plus complexe, cela pourrait avoir de plus graves conséquences.



Un exemple ? Sur le Site du Zéro lui-même, nous avions une fois oublié de gérer le cas où un paramètre viendrait à manquer. Un utilisateur avait essayé de l'enlever "pour voir" et notre page PHP était faite de telle sorte que si le paramètre manquait, ça supprimait toutes les préférences de tous les membres inscrits du site !



Vous n'en êtes pas encore là, mais je tiens à vous sensibiliser aux problèmes potentiels que cela peut provoquer. Soyez donc vigilants dès maintenant et ne supposez jamais que vous allez recevoir tous les paramètres que vous attendiez.

Contrôler la valeur des paramètres

Allons plus loin dans notre tripatouillage vicieux de l'URL. On va modifier notre code source pour gérer un nouveau paramètre : le nombre de fois que le message doit être répété. Les paramètres vont donc être :

bonjour.php?nom=Dupont&prenom=Jean&repeter=8

Le paramètre `repeter` définit le nombre de fois qu'on dit bonjour sur la page. Sauriez-vous adapter notre code pour qu'il dise bonjour le nombre de fois indiqué ? Voici la solution que je vous propose :

Code : PHP

```
<?php
if (isset($_GET['prenom']) AND isset($_GET['nom']) AND
    isset($_GET['repeter']))
{
    for ($i = 0 ; $i < $_GET['repeter'] ; $i++)
    {
        echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !<br
    />';
    }
}
else
{
    echo 'Il faut renseigner un nom, un prénom et un nombre de
répétitions !';
}
?>
```

On teste si le paramètre `repeter` existe lui aussi (avec `isset($_GET['repeter'])`). Si tous les paramètres sont bien là, on fait une boucle (j'ai choisi de faire un `for`, mais on aurait aussi pu faire un `while`). La boucle incrémente une petite variable `$i` pour répéter le message de bienvenue le nombre de fois indiqué.

Le résultat ? Si on va sur :

<http://localhost/tests/bonjour.php?nom=Dupont&prenom=Jean&repeter=8>

Alors le message de bienvenue s'affichera 8 fois :

Code : Console

```
Bonjour Jean Dupont !
```

Nous avons amélioré notre page pour qu'elle puisse dire bonjour plusieurs fois, et ce nombre de fois est personnalisable dans

l'URL. Très bien.

Maintenant, mettez-vous dans la peau du *méchant*. Soyez méchant. Soyez vicieux. Qu'est-ce qu'on pourrait faire que nous n'avions pas prévu juste en modifiant l'URL ?

J'ai une idée ! On peut mettre un nombre de répétitions très grand, par exemple 1984986545665156. La page PHP va boucler un très grand nombre de fois et sûrement que votre PC ne pourra jamais supporter une telle charge de travail. 

`http://localhost/tests/bonjour.php?nom=Dupont&prenom=Jean&repeter=1984986545665156`

Si vous avez peur d'essayer, je vous rassure : votre PC ne va pas prendre feu en faisant ça. Par contre, il va se mettre à consommer énormément de mémoire pour stocker tous les messages bonjour et n'arrivera sûrement pas jusqu'au bout (PHP s'arrête au bout d'un certain temps d'exécution). Ou alors s'il y arrive, votre page va être extrêmement surchargée de messages bonjour.



Mon dieu c'est horrible ! Comment peut-on empêcher ça ?

En étant plus malin et surtout plus prévoyant. Voici ce qu'il faut anticiper :

- Le cas où le nombre qu'on vous envoie **n'est pas une valeur raisonnable**.
 - Par exemple on peut dire que si on dépasse 100 fois, c'est trop, et il faut refuser d'exécuter la page.
 - De même, que se passe-t-il si je demande de répéter "-4 fois" ? Ca ne devrait pas être autorisé. Il faut que le nombre soit au moins égal à 1.
- Le cas où **la valeur n'est pas logique**, où elle n'est pas du tout ce qu'on attendait. Rien n'empêche le visiteur de remplacer la valeur du paramètre `repeter` par du texte !
 - Que se passe-t-il si on essaie d'accéder à `bonjour.php?nom=Dupont&prenom=Jean&repeter=grenouille` ? Cela ne devrait pas être autorisé. Le mot "grenouille" n'a pas de sens, on veut un nombre entier positif.
 - Que se passe-t-il si on essaie d'accéder à `bonjour.php?nom=Dupont&prenom=Jean&repeter=true` ? Cela ne devrait pas être autorisé non plus, on attendait un nombre entier positif, pas un booléen.

Pour résoudre ces problèmes, on doit :

1. Vérifier que `repeter` contient bien un nombre.
2. Vérifier ensuite que ce nombre est compris dans un intervalle raisonnable (entre 1 et 100 par exemple).

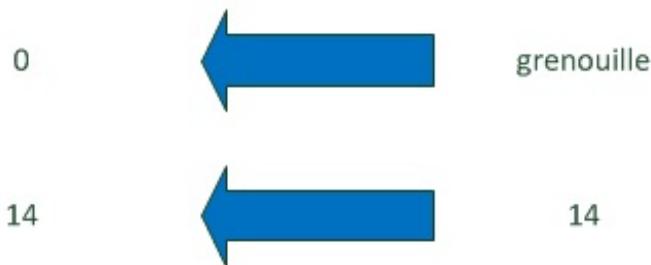
Pour vérifier que `repeter` contient bien un nombre, une bonne solution pourrait être de forcer la conversion de la variable en type entier (`int`). On peut utiliser pour cela ce qu'on appelle le *transtypage*, une notion qu'on n'a pas vue jusqu'ici mais qui est très simple à comprendre. Regardez ce code :

Code : PHP

```
<?php  
$_GET['repeter'] = (int) $_GET['repeter'];  
?>
```

Il faut lire cette instruction de droite à gauche. Le `(int)` entre parenthèses est comme un tuyau de conversion. Tout ce qui transite à travers ressort forcément en une valeur de type `int` (entier).

```
$_GET['repeter'] = (int) $_GET['repeter'];
```



Si la valeur est bien un entier (par exemple 14) alors elle n'est pas modifiée. En revanche, si la valeur contient autre chose qu'un entier (par exemple "grenouille") elle est transformée en entier. Ici, comme PHP ne peut pas associer de valeur à grenouille, il lui donne 0.

Après avoir exécuté cette instruction, la variable `$_GET['repeter']` contient maintenant forcément un nombre entier. Il ne reste plus qu'à vérifier que ce nombre est bien compris entre 1 et 100 à l'aide d'une condition, ce que vous savez faire normalement. 😊

Voici donc le code final sécurisé qui prévoit tous les cas pour éviter d'être pris au dépourvu par un utilisateur mal intentionné :

Code : PHP

```
<?php
if (isset($_GET['prenom']) AND isset($_GET['nom']) AND
    isset($_GET['repeter']))
{
    // 1 : On force la conversion en nombre entier
    $_GET['repeter'] = (int) $_GET['repeter'];

    // 2 : Le nombre doit être compris entre 1 et 100
    if ($_GET['repeter'] >= 1 AND $_GET['repeter'] <= 100)
    {
        for ($i = 0 ; $i < $_GET['repeter'] ; $i++)
        {
            echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !<br
        />';
        }
    }
    else
    {
        echo 'Il faut renseigner un nom, un prénom et un nombre de
répétitions !';
    }
?>
```

Cela fait beaucoup de conditions pour quelque chose qui était à la base assez simple. Mais c'était nécessaire. Vous devrez toujours faire très attention comme on l'a fait à prévoir tous les cas les plus tordus :

- Vérifier que tous les paramètres que vous attendiez sont bien là.
- Vérifier qu'ils contiennent bien des valeurs correctes comprises dans des intervalles raisonnables.

Si vous gardez cela en tête, vous n'aurez pas de problèmes. 😊



Nous n'avons pas encore vu tous les risques liés aux données envoyées par l'utilisateur. Cette première approche



devrait déjà vous avoir sensibilisé au problème, mais nous irons plus loin dans le chapitre suivant pour finir de couvrir ce sujet. Tout ceci est un peu complexe je suis d'accord, mais c'est réellement important !

L'utilisation des paramètres envoyés dans l'URL est simple : il suffit de récupérer ces valeurs dans l'array `$_GET`. Vous ne devriez pas avoir eu de mal à comprendre cela dans la première moitié de ce chapitre.

En revanche, j'ai particulièrement insisté dans ce chapitre sur le fait qu'il faut être très **vigilant** quand on récupère des données qui sont modifiables par l'utilisateur. N'ayez jamais confiance. Les programmeurs disent "*Never trust user input*" (ne faites jamais confiance en ce que l'utilisateur a saisi). Cette phrase devrait dès aujourd'hui être affichée sur un mur de votre chambre. Trop de programmeurs débutants ont fait confiance à ce que l'utilisateur allait saisir ("allez quoi, il ne va jamais rentrer *grenouille* à la place d'un nombre !") et je peux vous dire qu'ils s'en mordent aujourd'hui les doigts.

Heureusement ça ne sera pas votre cas ! J'ai été alarmiste, voire même un peu lourd dans ce chapitre, mais c'est pour votre bien. Vous allez voir, vous allez très vite avoir le réflexe de tester les données que votre page PHP reçoit !

Never trust user input.
Never trust user input.
Never trust user input.
Never trust user input.



Transmettre des données avec les formulaires

Les formulaires sont le principal moyen pour vos visiteurs de rentrer des informations sur votre site. Les formulaires permettent de créer l'*interactivité*.

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 pour échanger des informations avec nos visiteurs.

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. Le HTML permet de créer le formulaire, tandis que le PHP permet de traiter les informations que le visiteur a entrées dans le formulaire.

Ce chapitre est particulièrement important, nous réutiliserons ce que nous avons appris ici dans toute la suite du cours. Soyez attentifs !

Créer la base du formulaire

En HTML, pour 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 insérera ici les éléments de notre formulaire.  
</p>  
  
</form>
```

 On écrira le contenu de notre formulaire entre les balises <form> et </form>. Si vous avez lu mon cours de HTML/CSS vous verrez qu'une bonne partie de ce chapitre ne sera que des rappels puisque je vais vous présenter comment on insère les champs du formulaire dans la page. Cependant, et ce sera nouveau, nous allons aussi découvrir comment traiter en PHP les données qui ont été envoyées par le visiteur.

Je souhaite attirer votre attention sur la toute première ligne de ce code. Il y a 2 attributs très importants à connaître pour la balise <form> : la méthode (*method*) et la cible (*action*). Il est impératif que vous compreniez à quoi ils servent.

La méthode

Il faut savoir qu'il y a plusieurs moyens d'envoyer les données du formulaire (plusieurs "méthodes"). Les 2 méthodes possibles sont :

- **get** : les données transiteront par l'URL comme on l'a appris précédemment. On pourra les récupérer grâce à l'array `$_GET`. Cette méthode est assez peu utilisée car on ne peut pas envoyer beaucoup d'informations dans l'URL (je vous disais dans le chapitre précédent qu'il était préférable de ne pas dépasser 256 caractères).
- **post** : les données ne transiteront pas par l'URL, l'utilisateur ne les verra donc pas passer dans la barre d'adresse. Cette méthode permet d'envoyer autant de données que l'on veut, ce qui fait qu'on la privilégie le plus souvent. Néanmoins, les données ne sont pas plus sécurisées qu'avec la méthode GET et il faudra toujours vérifier si tous les paramètres sont bien présents et valides comme on l'a fait dans le chapitre précédent. **On ne doit pas plus faire confiance aux formulaires qu'aux URL.**

C'est à vous de choisir par quelle méthode vous souhaitez que les données du formulaire soient envoyées. Si vous hésitez, sachez que dans 99% des cas la méthode que l'on utilise est `post`, vous écrirez donc `method="post"` comme je l'ai fait.

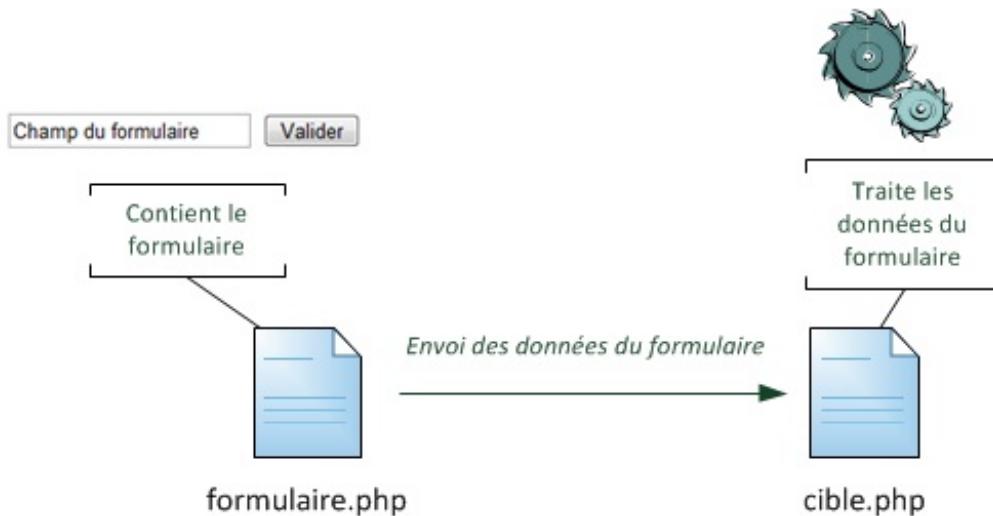
La cible

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

Imaginons le schéma suivant :



Dans cet exemple, le formulaire se trouve dans la page `formulaire.php`. Cette page ne fait aucun traitement particulier, mais une fois le formulaire envoyé (lorsqu'on a cliqué sur le bouton "Envoyer"), le visiteur est redirigé vers la page `cible.php` qui reçoit les données du formulaire.



Le nom de la page cible est défini grâce à l'attribut `action`.

La page cible ne doit pas forcément s'appeler `cible.php`. J'utilise ce nom dans mes exemples simplement pour que vous compreniez bien que c'est la page qui est appelée.

 En théorie, rien n'empêche d'ailleurs le formulaire de s'appeler lui-même. Il suffirait d'écrire `action="formulaire.php"`. Dans ce cas, la page du formulaire doit être capable aussi bien d'afficher le formulaire que de traiter les données. C'est tout à fait possible à faire mais afin de ne pas compliquer les choses trop vite, on va éviter de faire comme ça ici. 😊

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`).

Les éléments du formulaire

Dans un formulaire, vous le savez peut-être déjà, on peut insérer beaucoup d'éléments différents : zones de textes, boutons, cases à cocher, etc.

Je vais ici tous les lister et vous montrer comment vous servir de chacun d'eux dans la page `cible.php` qui fera le traitement. Vous allez voir, c'est vraiment très simple : au lieu de recevoir un array `$_GET` vous allez recevoir un array `$_POST` contenant les données du formulaire !

Les petites zones de texte

Une zone de texte ressemble à ceci : Votre pseudo :

En HTML, on l'insère tout simplement avec la balise :

Code : HTML

```
<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. A part ce détail, le fonctionnement reste le même.

Il y a 2 attributs à connaître que l'on peut ajouter à cette balise :

- `name` (obligatoire) : 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" />`
- `value` (facultatif) : c'est ce que contient la zone de texte au départ. Par défaut, la zone de texte est vide. Mais il peut être pratique de pré-remplir le champ. 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 mais n'ayez pas peur. Le fonctionnement est tout simple et a un air de déjà vu. Le texte que le visiteur aura rentré sera disponible dans `cible.php` sous la forme d'une variable appelée `$_POST['pseudo']`.

Pour l'exemple, je vous propose de créer un formulaire qui demande le prénom du visiteur et qui l'affiche ensuite fièrement sur la page `cible.php`. On va donc distinguer 2 codes sources : celui de la page du formulaire et celui de la page cible.

Voici le code de la page `formulaire.php` :

Code : PHP

```
<p>
    Cette page 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>
```



Rappel du HTML : le champ `<input type="submit" />` permet de créer le bouton de validation du formulaire qui commande l'envoi des données, et donc la redirection du visiteur vers la page cible.

Maintenant, je vous propose de créer la page `cible.php`. Cette page va recevoir le prénom dans une variable nommée `$_POST['prenom']`.

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="formulaire.php">clique  
ici</a> pour revenir à formulaire.php</p>
```

Ce lien ouvre la page `formulaire.php` pour que vous puissiez tester :

[Essayer !](#)

Dans `cible.php` on a affiché une variable `$_POST['prenom']` qui contient ce que l'utilisateur a rentré dans le formulaire.

Les grandes zones de texte

La grande zone de texte (qu'on appelle aussi "Zone de saisie multiligne" 😊), 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.
```

```
</textare>
```

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']`.

Vous remarquerez qu'il n'y a pas d'attribut `value`. En fait, le texte par défaut est ici écrit entre le `<textarea>` et le `</textarea>`. Si vous ne voulez rien mettre par défaut, alors n'écrivez rien entre `<textarea>` et `</textarea>`.



Les attributs `rows` et `cols` permettent de définir la taille de la zone de texte en hauteur et en largeur respectivement.

La liste déroulante

La liste déroulante, c'est ça :



On utilise le code HTML suivant pour construire une liste déroulante :

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 ensuite 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".

Vous pouvez aussi définir le choix par défaut de la liste. 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 :

Code : PHP

```
<option value="choix3" selected="selected">Choix 3</option>
```

Les cases à cocher

Une case à cocher ressemble à ceci :

- Frites
- Steak haché
- Epinards
- Huitres

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

Code : HTML

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

L'utilisation de la balise `<label>` n'est pas obligatoire mais je la recommande fortement. Elle permet d'associer le libellé à la case à cocher qui a le même `id` que son attribut `for`, ce qui permet de cliquer sur le libellé pour cocher la case. On y gagne donc en ergonomie d'utilisation. Pour en savoir plus, référez-vous au [chapitre sur les formulaires du cours sur HTML et CSS](#).

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

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

Si vous voulez que la case soit cochée par défaut, il faudra lui rajouter l'attribut `checked="checked"`. Par exemple :

Code : HTML

```
<input type="checkbox" name="case" checked="checked" />
```

Les boutons d'option

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

Aimez-vous les frites ? Oui Non

Le code correspondant à cet exemple est le suivant :

Code : HTML

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

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 doivent avoir 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, faites pareil que pour les cases à cocher : rajoutez un attribut `checked="checked"`. Ici, comme vous pouvez le voir, "Oui" est sélectionné par défaut.

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

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

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

Code : HTML

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

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

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

On croit par erreur que, parce que ces champs sont cachés, le visiteur ne peut pas les voir. C'est faux ! En effet, n'importe quel visiteur peut afficher le code source de la page et voir qu'il y a des champs cachés en lisant le code HTML. Mieux, il peut même modifier la valeur du champ caché s'il a les outils appropriés.

 Que faut-il retenir ? Ce n'est pas parce que le champ est caché que vous devez considérer qu'il est inviolable. N'importe quel visiteur (un peu malin) peut le lire, modifier sa valeur et même le supprimer. *Ne faites pas confiance aux données envoyées par le visiteur !* Vous vous souvenez de cette règle dont je vous ai parlé dans le chapitre précédent, elle est plus que jamais d'actualité.

Ne faites jamais confiance aux données reçues : la faille XSS

Vous vous souvenez des mises en garde que j'avais faites dans le chapitre précédent ? Elles ne concernaient pas que les paramètres qui transitent par l'URL : tout cela vaut aussi pour les formulaires !



Mais... autant je vois comment on peut modifier l'URL, autant je ne comprends pas comment peut faire un visiteur pour modifier le formulaire de mon site et trafiquer les données ?

On a tendance à croire que les visiteurs ne peuvent pas "bidouiller" le formulaire mais c'est faux. Je vais vous montrer dans un premier temps pourquoi les formulaires ne sont pas plus sûrs que les URL, puis je vous parlerai d'un autre danger important : la faille XSS. Avec ça, nous aurons vraiment fait le tour de ce qu'il faut savoir pour être tranquille par la suite !

Pourquoi les formulaires ne sont pas sûrs

Tout ce que nous avons appris dans le chapitre précédent sur les URL reste valable ici. Tout ce qui vient de l'utilisateur, à savoir les données de `$_GET` et de `$_POST`, doit être traité avec la plus grande méfiance.

Vous ne pouvez pas supposer que vous allez recevoir ce que vous attendiez.

Cette règle est très simple, mais je vous propose un exemple concret pour bien visualiser le problème. Imaginez que vous demandez à vos visiteurs dans un champ leur date de naissance "au format JJ/MM/AAAA". Combien vont respecter cette mise en forme ? Combien vont se tromper par erreur ? Je vous garantis que parmi tous vos visiteurs, alors que vous attendiez quelque chose comme "04/10/1987", vous allez tomber sur un type qui va écrire "Je suis né le 4 octobre 1987". C'est un exemple un peu extrême mais ça va vous arriver, soyez-en sûrs. Par conséquent, quand vous ferez le traitement de la date en PHP, il faudra bien vérifier qu'elle respecte le format que vous avez indiqué.



Pour vérifier si une chaîne de texte correspond à un certain format, comme JJ/MM/AAAA, on peut utiliser une validation par expression régulière. Les expressions régulières feront l'objet de 2 chapitres vers la fin de ce cours car il s'agit d'un sujet assez complexe.

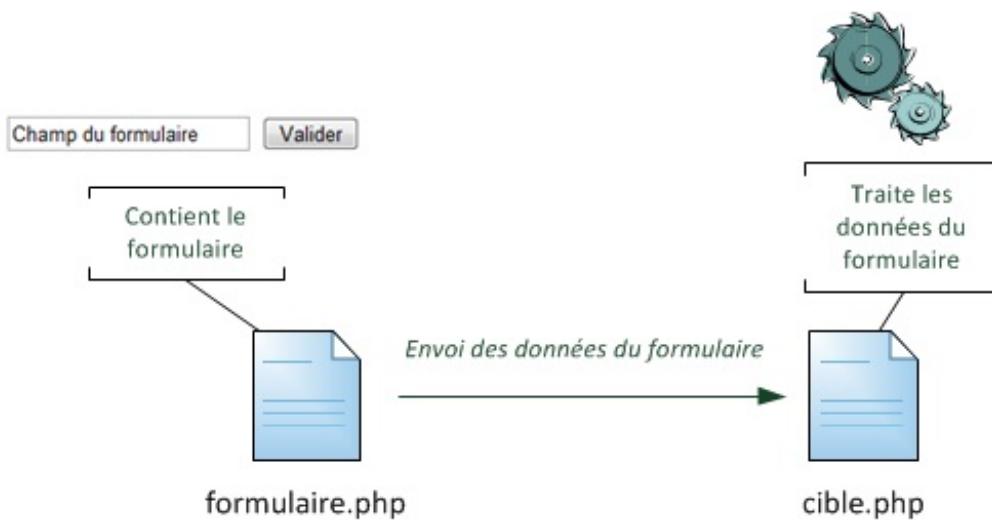
De la même manière, comme dans le chapitre précédent, même si vous demandez un nombre compris entre 1 et 100 il y aura bien quelqu'un pour écrire "48451254523". Soyez donc vigilants et n'ayez jamais confiance dans ce qui vient de l'utilisateur, à savoir les données issues des array `$_GET` et `$_POST`.

Avec les formulaires, vous ne pouvez pas non plus supposer qu'on va vous envoyer tous les champs que vous attendiez. Un visiteur peut très bien s'amuser à supprimer un champ de texte, et dans ce cas votre page `cible.php` ne recevra jamais le texte qu'elle attendait ! Il faudra impérativement qu'elle vérifie que toutes les données qu'elle attendait sont bien là avant d'effectuer des opérations.



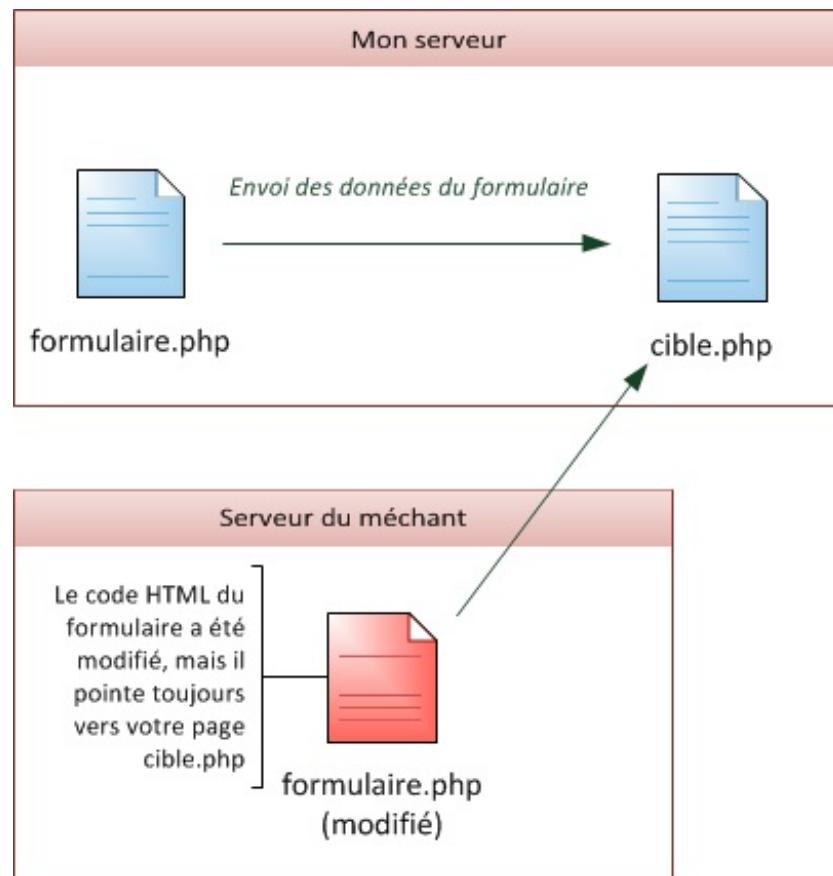
Puisque la page du formulaire se trouve sur mon site, comment peut faire un visiteur pour modifier ma page web ? Il peut voir les sources mais pas les modifier !

En effet, vos visiteurs ne peuvent pas modifier vos pages web sur le serveur... Mais ils peuvent les reprendre et les modifier ailleurs. Souvenez-vous de ce schéma :



La page `formulaire.php` contient le formulaire et `cible.php` traite les données qu'on lui a envoyées. Autant le code PHP n'est jamais visible par vos visiteurs, autant le code HTML du formulaire, lui, peut être vu par tout le monde.

A partir de là, qu'est-ce qui empêche quelqu'un de créer une copie légèrement modifiée de votre formulaire et de la stocker sur son serveur ?



Sur le schéma ci-dessus, le "méchant" (nous l'appellerons comme ça, parce que ça lui va bien 😈) a pris le code HTML de votre formulaire, l'a modifié et l'a enregistré sur son serveur (ou même sur son ordinateur). L'attribut action a été modifié pour indiquer l'adresse absolue (donc complète) de votre page cible :

Code : PHP

```
<form method="post" action="http://www.monsite.com/cible.php">
```

Le méchant peut maintenant modifier votre formulaire, ajouter des champs, en supprimer, bref faire ce qu'il veut avec ! Votre page `cible.php` n'y verra que du feu car il est *impossible* de savoir avec certitude de quel formulaire vient le visiteur.

Ces explications sont assez techniques. En fait, on les réserve normalement aux personnes plus expérimentées que les débutants. Cependant, je tenais volontairement à vous montrer comment c'est possible. Même si tout n'est pas totalement clair dans votre tête, vous avez au moins *l'idée* du mode de fonctionnement.

S'il y a une chose à retenir ici, c'est que **les formulaires sont modifiables par tous les visiteurs** contrairement à ce qu'on pourrait penser. Par conséquent, votre page `cible.php` devra être aussi vigilante que nous l'avons été dans le chapitre précédent et ne pas faire confiance aux données de l'utilisateur. Souvenez-vous de cette maxime de programmeur : *Never trust user input.* 😊



Il y a un moyen encore plus simple de modifier le formulaire de votre site sans avoir accès à votre serveur. Internet Explorer 8 et Google Chrome embarquent des "outils pour les développeurs" qui permettent de modifier le code HTML de la page que l'on visite en temps réel. Firefox peut aussi faire de même avec son célèbre plugin Firebug.

La faille XSS : attention au code HTML que vous recevez !

Il reste un dernier point important à voir ensemble et après j'arrête de vous faire peur, promis ! 😊

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

Reprenons la page qui affiche le prénom qu'on lui envoie. Elle contient notamment le code suivant :

Code : PHP

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

Si le visiteur décide d'écrire du code HTML à la place de son prénom, cela fonctionnera très bien ! Par exemple, imaginons qu'il écrive dans le champ "Prénom" le code : `Badaboum`. Le code source HTML qui sera généré par PHP sera le suivant :

Code : HTML

```
<p>Je sais comment tu t'appelles, hé hé. Tu t'appelles  
<strong>Badaboum</strong> !</p>
```



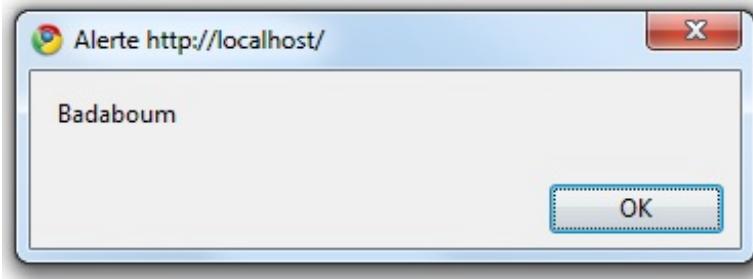
Et alors ? S'il veut mettre son prénom en gras c'est son problème non ?

Outre le fait qu'il peut insérer n'importe quel code HTML (et rendre votre page invalide), ce qui n'est pas le plus grave, il peut aussi ouvrir des balises de type `<script>` pour faire exécuter du code Javascript au visiteur qui visualisera la page !

Code : HTML

```
<p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <script type="text/javascript">alert('Badaboum')</script> !</p>
```

Tous les visiteurs qui arriveront sur cette page verront une boîte de dialogue Javascript s'afficher. Plutôt gênant.



Les choses deviennent vraiment critiques si le visiteur est assez malin pour récupérer vos cookies de cette façon-là. Les cookies stockent des informations sur votre session et parfois des informations plus confidentielles comme votre pseudonyme et votre mot de passe sur le site ! Il est possible de forcer le visiteur qui lira le code Javascript à vous envoyer tous les cookies qu'il a enregistrés pour votre site web, ce qui peut conduire au vol de son compte sur ce site. Je ne rentrerai pas dans le détail de cette méthode car on s'éloignerait un peu trop du sujet, mais sachez que c'est possible et qu'il faut donc à tout prix éviter qu'un visiteur puisse injecter du code Javascript dans vos pages !

Résoudre le problème est facile : il faut protéger le code HTML en l'échappant, c'est-à-dire en affichant les balises (ou en les retirant) plutôt que de les faire exécuter par le navigateur.

Mal : le code HTML a été inséré tel quel dans la page, n'importe quel visiteur peut placer du HTML



Je sais comment tu t'appelles, hé hé. Tu t'appelles **Badaboum** !

Je sais comment tu t'appelles, hé hé. Tu t'appelles **Badaboum** !



Bien : le code HTML a été « échappé » et il n'est pas interprété par le navigateur. On voit le HTML mais celui-ci est inoffensif, il ne s'exécute pas.

Pour échapper le code HTML, il suffit d'utiliser la fonction `htmlspecialchars` qui va transformer les chevrons des balises

HTML <> en < et > respectivement. Cela provoquera l'affichage de la balise plutôt que son exécution.

Code : PHP

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

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

Code : HTML

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

 Il faut penser à utiliser cette fonction sur tous les textes envoyés par l'utilisateur qui sont susceptibles d'être affichés sur une page web. Sur un forum par exemple, il faut penser à échapper les messages postés par vos membres, mais aussi leurs pseudos (ils peuvent s'amuser à mettre du HTML dedans !) ainsi que leurs signatures ! Bref, tout ce qui est affiché et qui vient à la base d'un visiteur, vous devez penser à le protéger avec htmlspecialchars.

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

L'envoi de fichiers

Vous saviez qu'on pouvait aussi envoyer des fichiers grâce aux formulaires ? Vous aurez besoin de lire cette section si vous voulez que vos visiteurs puissent envoyer (on dit aussi *uploader*) des images, des programmes ou tout autre type de fichier sur votre site.

Cette section est un peu plus complexe que le reste du chapitre. Sa lecture n'est d'ailleurs pas obligatoire pour la bonne compréhension de la suite du cours.

Par conséquent, n'hésitez pas à revenir la lire plus tard, lorsque vous en aurez besoin, si vous ne voulez pas vous attaquer de suite à une partie un peu "difficile".

Là encore, ça se passe en 2 temps :

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

 Attention : l'envoi du fichier peut être un peu long si celui-ci est gros. Il faudra dire au visiteur de ne pas s'impatienter pendant l'envoi.

On va commencer par créer le formulaire permettant d'envoyer un fichier (une simple page HTML). Nous verrons ensuite comment traiter l'envoi du fichier côté serveur avec PHP.

Le formulaire d'envoi de fichier

Dès l'instant où votre formulaire propose aux visiteurs d'envoyer un fichier, il faut ajouter l'attribut `enctype="multipart/form-data"` à la balise `<form>`.

Code : HTML

```
<form action="cible_envoi.php" method="post"
      enctype="multipart/form-data">
    <p>Formulaire d'envoi de fichier</p>
  </form>
```

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

Maintenant que c'est fait, nous pouvons ajouter à l'intérieur du formulaire une balise permettant d'envoyer un fichier. C'est une balise très simple de type `<input type="file" />`. Il faut penser comme toujours à donner un nom à ce champ de formulaire (grâce à l'attribut `name`) pour que PHP puisse reconnaître le champ après.

Code : HTML

```
<form action="cible_envoi.php" method="post"
      enctype="multipart/form-data">
  <p>
    Formulaire d'envoi de fichier :<br />
    <input type="file" name="monfichier" /><br />
    <input type="submit" value="Envoyer le fichier" />
  </p>
```

```
</form>
```

Voilà, c'est suffisant 😊

Vous pouvez ajouter d'autres champs plus classiques au formulaire (champ de texte, cases à cocher). Vous pouvez aussi proposer d'envoyer plusieurs fichiers en même temps.

Là on va se contenter d'un seul champ (envoi de fichier) pour rester simple.

Le traitement de l'envoi en PHP

Comme vous avez dû le remarquer, le formulaire pointe vers une page PHP qui s'appelle `cible_envoi.php`. Le visiteur sera donc redirigé sur cette page après l'envoi du formulaire.

C'est maintenant que ça devient important. Il faut que l'on écrive le code de la page `cible_envoi.php` pour traiter l'envoi du fichier.



"Traiter l'envoi du fichier" ? C'est-à-dire ?

Si le fichier a été envoyé sur le serveur c'est bon non ? Qu'est-ce que PHP aurait besoin de faire ?

En fait, au moment où la page PHP s'exécute, le fichier a été envoyé sur le serveur mais il est stocké dans un **dossier temporaire**.

C'est à vous de décider si vous acceptez définitivement le fichier ou non. Vous pouvez par exemple vérifier si le fichier a la bonne extension (si vous demandiez une image et qu'on vous envoie un ".txt", vous devrez refuser le fichier).

Si le fichier est bon, vous l'accepterez définitivement grâce à la fonction `move_uploaded_file`.



Mais comment je sais si "le fichier est bon" ?

Pour chaque fichier envoyé, une variable `$_FILES['nom_du_champ']` est créée. Dans notre cas, la variable s'appellera `$_FILES['monfichier']`.

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

Variable	Signification
<code>\$_FILES['monfichier']['name']</code>	Contient le nom du fichier envoyé par le visiteur.
<code>\$_FILES['monfichier']['type']</code>	Indique le type du fichier envoyé. Si c'est une image gif par exemple, le type sera <code>image/gif</code> .
<code>\$_FILES['monfichier']['size']</code>	Indique la taille du fichier envoyé. Attention : cette taille est en octets. Il faut environ 1 000 octets pour faire 1 Ko, et 1 000 000 octets pour faire 1 Mo. Attention, la taille de l'envoi est limitée par PHP. Par défaut, il est impossible d'uploader des fichiers de plus de 8 Mo.
<code>\$_FILES['monfichier']['tmp_name']</code>	Juste après l'envoi, le fichier est placé dans un répertoire temporaire sur le serveur en attendant que votre script PHP décide si oui ou non il accepte de le stocker définitivement. Cette variable contient l'emplacement temporaire du fichier (c'est PHP qui gère ça).
<code>\$_FILES['monfichier']['error']</code>	Contient un code d'erreur permettant de savoir si l'envoi s'est bien effectué ou s'il y a eu un problème, et si oui lequel.

La variable vaut 0 s'il n'y a pas eu d'erreur.

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

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

Je vous propose de faire les vérifications suivantes pour décider si on accepte le fichier ou non :

1. Vérifier tout d'abord si le visiteur a bien envoyé un fichier (en testant la variable `$_FILES['monfichier']` avec `isset()` et s'il n'y a pas eu d'erreur d'envoi (grâce à `$_FILES['monfichier']['error']`))
2. Vérifier si la taille du fichier ne dépasse pas 1 Mo par exemple (environ 1 000 000 d'octets) grâce à `$_FILES['monfichier']['size']`
3. Vérifier si l'extension du fichier est autorisée (il faut interdire à tout prix que les gens puissent envoyer des fichiers PHP, sinon ils pourraient exécuter des scripts sur votre serveur !). Dans notre cas, nous autoriserons seulement les images (fichiers .png, .jpg, .jpeg et .gif).
Nous analyserons pour cela la variable `$_FILES['monfichier']['name']`.

Nous allons donc faire une série de tests dans notre page **cible_envoi.php**.

1/ Tester si le fichier a bien été envoyé

On commence par vérifier qu'un fichier a été envoyé en testant si la variable `$_FILES['monfichier']` existe avec `isset()`.

On vérifie dans le même temps s'il n'y a pas d'erreur d'envoi.

Code : PHP

```
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas
d'erreur
if (isset($_FILES['monfichier']) AND $_FILES['monfichier']['error']
== 0)
{
}
?>
```

2/ Vérifier la taille du fichier

On veut interdire que le fichier dépasse 1 Mo, soit environ 1 000 000 d'octets (j'arrondis pour simplifier). On doit donc tester `$_FILES['monfichier']['size']` :

Code : PHP

```
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas
d'erreur
if (isset($_FILES['monfichier']) AND $_FILES['monfichier']['error']
== 0)
```

```

{
    // Testons si le fichier n'est pas trop gros
    if ($_FILES['monfichier']['size'] <= 1000000)
    {
        ...
    }
?>

```

3/ Vérifier l'extension du fichier

On peut récupérer l'extension du fichier dans une variable grâce à ce code :

Code : PHP

```

<?php
$infosfichier = pathinfo($_FILES['monfichier']['name']);
$extension_upload = $infosfichier['extension'];
?>

```

La fonction `pathinfo` renvoie un array contenant entre autres l'extension du fichier dans `$infosfichier['extension']`. On stocke ça dans une variable `$extension_upload`.

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

Ouf ! Ca donne ça au final :

Code : PHP

```

<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas
d'erreur
if (isset($_FILES['monfichier']) AND $_FILES['monfichier']['error']
== 0)
{
    // Testons si le fichier n'est pas trop gros
    if ($_FILES['monfichier']['size'] <= 1000000)
    {
        // Testons si l'extension est autorisée
        $infosfichier = pathinfo($_FILES['monfichier']['name']);
        $extension_upload = $infosfichier['extension'];
        $extensions_autorisees = array('jpg', 'jpeg', 'gif', 'png');
        if (in_array($extension_upload, $extensions_autorisees))
        {
            ...
        }
    }
?>

```

4/ Valider l'upload du fichier

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

Cette fonction prend 2 paramètres :

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

Je propose de placer le fichier dans un sous-dossier "uploads".

On gardera le même nom de fichier que celui d'origine. Comme `$_FILES['monfichier']['name']` contient le chemin entier vers le fichier d'origine (C:\dossier\fichier.png par exemple), il nous faudra extraire le nom du fichier. On peut utiliser pour cela la fonction `basename` qui renverra juste "fichier.png" 😊

Code : PHP

```
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas
d'erreur
if (isset($_FILES['monfichier'])) AND $_FILES['monfichier']['error']
== 0)
{
    // Testons si le fichier n'est pas trop gros
    if ($_FILES['monfichier']['size'] <= 1000000)
    {
        // Testons si l'extension est autorisée
        $infosfichier =
pathinfo($_FILES['monfichier']['name']);
        $extension_upload = $infosfichier['extension'];
        $extensions_authorized = array('jpg', 'jpeg', 'gif',
'png');
        if (in_array($extension_upload,
$extensions_authorized))
        {
            // On peut valider le fichier et le stocker
            définitivement
            move_uploaded_file($_FILES['monfichier']['tmp_name'],
'uploads/' . basename($_FILES['monfichier']['name']));
            echo "L'envoi a bien été effectué !";
        }
    }
}
?>
```

 Lorsque vous mettrez le script sur le net à l'aide d'un logiciel FTP, vérifiez que le dossier "uploads" sur le serveur existe et qu'il a les droits d'écriture. Pour ce faire, sous Filezilla par exemple, faites un clic droit sur le dossier et choisissez "Attributs du fichier".

Cela vous permettra d'édition les droits du dossier (on parle de CHMOD). Mettez les droits à 733, ainsi PHP pourra placer des fichiers uploadés dans ce dossier.

Ce script est un début, mais en pratique il vous faudra sûrement encore l'améliorer. Par exemple, si le nom du fichier contient des espaces ou des accents ça posera un problème une fois envoyé sur le web. D'autre part, si quelqu'un envoie un fichier qui a le même nom que celui d'une autre personne, l'ancien sera écrasé !

La solution consiste en général à "choisir" nous-mêmes le nom du fichier stocké sur le serveur plutôt que de se servir du nom d'origine. Vous pouvez faire un compteur qui s'incrémente : 1.png, 2.png, 3.jpg, etc.

 Soyez toujours très vigilants sur la sécurité, vous devez éviter que quelqu'un puisse envoyer des fichiers PHP sur votre serveur.

Pour aller plus loin, je vous recommande de lire le [tutoriel de DHKold sur l'upload de fichiers par formulaire](#). Il est plus complet que le mien et vous expliquera mieux certains détails.

Bonne lecture 😊

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 vous 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 😊

Instructions pour réaliser le TP

Les 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 d'un formulaire
- 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 chapitre qui correspond, vous en aurez besoin pour traiter convenablement le TP. Vous verrez, ce TP ne nous demandera pas de faire des choses compliquées. Le but est simplement d'assembler toutes vos connaissances pour répondre à un problème précis.

Votre objectif

Voici le scénario : vous voulez mettre en ligne une page web pour donner des informations confidentielles à certaines personnes. Cependant, pour limiter l'accès à cette page, il faudra connaître un mot de passe.

Dans notre cas, les données confidentielles seront les codes d'accès au serveur central de la NASA (soyons fous ! ). Le mot de passe pour pouvoir visualiser les codes d'accès sera kangourou.

Sauriez-vous réaliser une page qui n'affiche ces codes secrets que si on a rentré le bon mot de passe ?

Comment procéder ?

Pour travailler correctement, 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).

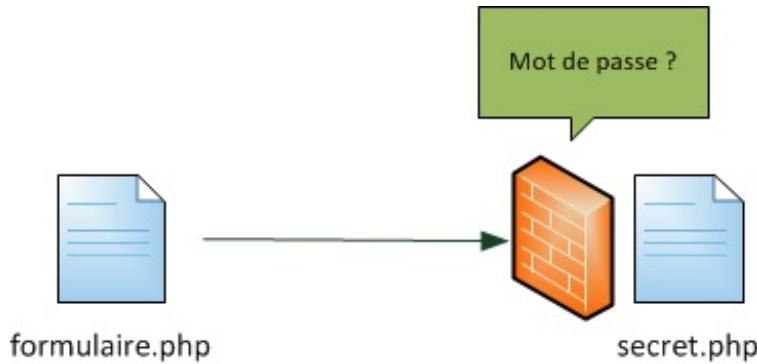
Et pour montrer l'exemple, nous allons suivre cette liste pour notre TP.

Problème posé

On doit protéger l'accès à une page par un mot de passe. La page ne doit pas s'afficher si on n'a pas le mot de passe.

Schéma du code

Pour que l'utilisateur puisse rentrer le mot de passe, le plus simple est de créer un formulaire. Celui-ci appellera la page protégée et lui enverra le mot de passe.
L'accès au contenu de la page ne sera autorisé que si le mot de passe est kangourou.



Vous devez donc créer 2 pages web :

- **formulaire.php** : contient un simple formulaire comme vous savez les faire.
- **secret.php** : contient les "codes secrets" mais ne les affiche que si on lui donne le mot de passe.

Connaissances requises

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

A 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 ne marche jamais du premier coup ! 😊

Bon code !

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.

Comme vous le savez, il y a 2 pages à créer. Commençons par la plus simple, `formulaire.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" 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="secret.php" method="post">
    <p>
      <input type="password" 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>
```

Si vous avez bien suivi le chapitre sur les formulaires, vous ne devriez avoir eu aucun mal à réaliser ce formulaire. J'ai choisi un champ de type "password" puisqu'il s'agit d'un mot de passe. A part ça, rien de bien particulier.

Maintenant, intéressons-nous à la page `secret.php` qui est appelée par le formulaire.

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>Codes d'accès au serveur central de la NASA</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
</head>
<body>

  <?php
  if (isset($_POST['mot_de_passe'])) AND $_POST['mot_de_passe'] ==
"kangourou" // Si le mot de passe est bon
  {
  // On affiche les codes
  ?>
  <h1>Voici les codes d'accès :</h1>
  <p><strong>CRD5-GTFT-CK65-JOPM-V29N-24G1-HH28-LLFV</strong></p>
  <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>
<?php
}
else // Sinon, on affiche un message d'erreur
{
    echo '<p>Mot de passe incorrect</p>';
}
?>

</body>
</html>
```

Dans la page secrète, on vérifie d'abord si on a envoyé un mot de passe (avec `isset`) et si ce mot de passe correspond bien à celui que l'on attendait (`kangourou`). Si ces 2 conditions sont remplies, on affiche alors les codes d'accès.

Comme vous le voyez, je n'ai pas fait un `echo` pour afficher tout ce texte. Quand il y a beaucoup de texte à afficher, il est préférable de fermer les balises PHP après l'accolade du `if`, c'est plus simple et plus lisible.
En revanche, pour le cas du `else` comme il y avait une seule petite phrase à afficher, j'ai choisi de l'afficher avec un `echo`.

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 !



Cette protection est-elle vraiment efficace ?

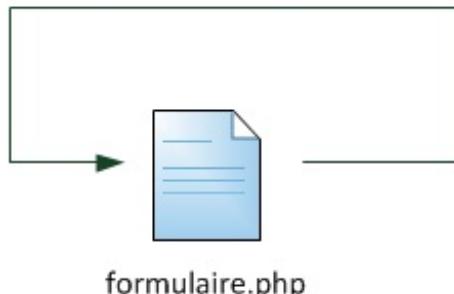
Oui, honnêtement elle l'est. Du moins, elle est efficace si vous mettez un mot de passe compliqué (pas "kangourou" 😊).
Pour moi, un bon mot de passe c'est long, avec plein de caractères bizarres, des majuscules, des minuscules, des chiffres, etc.
Par exemple `k7hYTe40Lm8Mf` est un bon mot de passe qui a peu de chances d'être trouvé "par hasard".

Aller plus loin

Si vous le souhaitez, sachez qu'il est possible de réaliser ce TP en une seule page au lieu de deux.

Imaginez pour cela que le formulaire, sur la page `formulaire.php`, s'appelle lui-même. En clair, l'attribut `action` du formulaire serait `action="formulaire.php"`. Cela voudrait dire que les données seraient renvoyées sur la même page :

Envoi des données du formulaire



Dans ce mode de fonctionnement, la page `formulaire.php` contiendrait à la fois le formulaire et le message secret.



Comment peut-on faire ça ? Ce n'est pas dangereux ? Ce ne serait pas très sécurisé non ?

On peut très bien faire cela de façon tout à fait sécurisée, c'est juste un peu plus "dur" à imaginer.

Il faut construire le code de votre page `formulaire.php` en 2 grandes parties :

- Si aucun mot de passe n'a été envoyé (ou s'il est faux) : afficher le formulaire
- Si le mot de passe a été envoyé et qu'il est bon : afficher les codes secrets

Toute votre page PHP sera donc architecturée autour d'un grand `if` qui pourrait ressembler à quelque chose comme ça :

Code : PHP

```
<?php

// Le mot de passe n'a pas été envoyé ou n'est pas bon
if (!isset($_POST['mot_de_passe'])) OR $_POST['mot_de_passe'] != "kangourou"
{
    // Afficher le formulaire de saisie du mot de passe
}
// Le mot de passe a été envoyé et il est bon
else
{
    // Afficher les codes secrets
}

?>
```

Voilà dans les grandes lignes comment on ferait. A chaque fois que la page `formulaire.php` est appelée, elle détermine (grâce au `if`) si on l'appelle pour afficher la partie secrète ou si on l'appelle pour afficher le formulaire de saisie du mot de passe.

Voici alors ce qui se passera :

1. La première fois que le visiteur charge la page `formulaire.php`, aucune donnée POST n'est envoyée à la page. C'est donc le formulaire qui s'affiche.
2. Une fois qu'on a envoyé le formulaire, la page `formulaire.php` est rechargée et cette fois elle reçoit les données POST qu'on vient d'envoyer. Elle peut donc les analyser et, si le mot de passe est bon, elle affiche les codes secrets.

Sauriez-vous refaire ce TP en une seule page en vous basant sur mes indices ? Essayez ! Ce sera un très bon exercice ! 😊
Et si vous avez des difficultés, n'hésitez pas à demander de l'aide sur le [forum PHP](#) du site.



Vous pourriez même aller plus loin, car dans mon schéma de code précédent je n'ai pas prévu de cas pour afficher "Mot de passe incorrect". Cela peut se faire facilement en découplant votre page en 3 à l'aide d'un `elseif` : `formulaire, mot de passe incorrect, codes secrets.`

Voilà qui devrait vous avoir permis de vous entraîner ! Que ce TP vous ait paru facile ou difficile, l'essentiel est de lui avoir accordé du temps. C'est dans les TP que vous apprenez en fait le plus de choses, donc n'hésitez pas à y revenir, à les relire et à refaire l'exercice !



Variables superglobales, sessions et cookies

Vous avez probablement remarqué que les array `$_GET` et `$_POST` sont des variables un peu particulières : leur nom est écrit en majuscules et commence par un underscore, mais surtout ces variables sont générées automatiquement par PHP. Ce sont ce qu'on appelle des variables superglobales.

Il existe d'autres types de variables superglobales que nous allons découvrir dans ce chapitre. Parmi elles, certaines permettent de stocker des informations pendant la durée d'une visite (c'est le principe des sessions), mais aussi de stocker des informations sur l'ordinateur de vos visiteurs pendant plusieurs mois (c'est le principe des cookies).

Les variables superglobales

Les variables superglobales sont des variables un peu particulières pour 3 raisons :

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

Pour afficher le contenu d'une superglobale et voir ce qu'elle contient, le plus simple est d'utiliser la fonction `print_r` puisqu'il s'agit d'un array. Exemple :

Code : PHP

```
<pre>
<?php
print_r($_GET);
?>
</pre>
```

Je vous propose de faire le tour des principales variables superglobales qui existent. Nous ne les utiliserons pas toutes, mais nous aurons fait un petit tour d'horizon des superglobales pour pouvoir nous concentrer ensuite sur les plus utiles d'entre elles.

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

Vous connaissez déjà une bonne partie de ces variables superglobales comme vous pouvez le constater. Je vous propose d'étudier plus en détail les sessions et les cookies. Avec ça nous aurons fait le tour des principaux moyens de transmettre des variables de page en page ! 😊

Les sessions

Les sessions sont un moyen de conserver des variables sur toutes les pages de votre site. Jusqu'ici, nous étions parvenus 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 (à l'aide d'un formulaire).

Mais imaginez maintenant que vous souhaitez transmettre des variables sur toutes les pages de votre site pendant la durée d'une visite d'un visiteur. Ce ne serait pas facile avec GET et POST car ils sont plutôt faits pour transmettre les informations une seule fois, d'une page à une autre. On sait ainsi envoyer d'une page à une autre le nom et le prénom du visiteur, mais dès qu'on charge une autre page ces informations sont "oubliées". C'est pour cela qu'on a inventé les sessions.

Fonctionnement des sessions

Comment sont gérées les sessions en PHP ? Voici les 3 étapes à connaître :

1. Un visiteur arrive sur votre site. On demande à créer une session pour lui. PHP génère alors un numéro unique. Ce numéro est souvent très gros et généralement écrit en hexadécimal. Par exemple :
a02bbff6198e6e0cc2715047bc3766f.



Ce numéro sert d'identifiant et est appelé "ID de session" (ou PHPSESSID). PHP transmet automatiquement cet ID de page en utilisant un cookie en règle générale.

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

Tout ceci peut vous sembler un peu compliqué, mais c'est en fait très simple à utiliser. Vous devez connaître 2 fonctions :

- `session_start()` : démarre le système de sessions. Si le visiteur vient d'arriver sur le site, alors un numéro de session est généré pour lui. Vous devez appeler cette fonction au tout début de chacune des pages où vous avez besoin des variables de session.
- `session_destroy()` : ferme la session du visiteur. Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page de votre site pendant plusieurs minutes (c'est le *timeout*), mais vous pouvez aussi créer une page "Déconnexion" si le visiteur souhaite se déconnecter manuellement.



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

Exemple d'utilisation des sessions

Je vous propose d'étudier un exemple concret pour que vous voyiez à quel point c'est simple à utiliser :

Code : PHP

```

<?php
// On démarre la session AVANT d'écrire du code HTML
session_start();

// On s'amuse à créer quelques variables de session dans $_SESSION
$_SESSION['prenom'] = 'Jean';
$_SESSION['nom'] = 'Dupont';
$_SESSION['age'] = 24;
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/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 les variables de session n'importe où dans le code (pas seulement au début comme je l'ai fait ici). La seule chose qui importe, c'est que le `session_start()` soit fait au tout début de la page.

Comme vous le voyez, j'ai créé 3 variables de session qui contiennent ici le nom, le prénom et l'âge du visiteur. J'ai fait des liens vers d'autres pages de mon site.

Notez quelque chose de très important : mes liens sont tous simples et ne transmettent aucune information. Je ne m'occupe de rien : ni de transmettre le nom, prénom, âge du visiteur, ni de transmettre l'ID de session. PHP gère tout pour nous. PHP s'occupe tout seul de transmettre les variables !

Maintenant, sur toutes les pages de mon site, je peux retrouver les variables `$_SESSION['prenom']`, `$_SESSION['nom']` et `$_SESSION['age']` ! Bien entendu, il faudra démarrer le système de session sur toutes les pages avec `session_start()`.

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 ça ? :-D
</p>

</body>
</html>

```

Essayer !

Vous voyez ? On a juste fait démarrer la session avec un `session_start()`, puis nous avons affiché 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, on peut créer des variables de session comme on crée des variables, à condition de les écrire dans l'array `$_SESSION` et d'avoir lancé le système de sessions avec `session_start()`. Ces variables sont ainsi conservées de page en page pendant toute la durée de la visite de votre visiteur. 

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

L'utilité des sessions en pratique

Concrètement, les sessions peuvent servir dans de nombreux cas sur votre site (et pas seulement pour retenir un nom et un prénom !). Voici quelques exemples :

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

 Si votre site est hébergé chez Free.fr, vous devrez créer un dossier appelé "sessions" à la racine de votre FTP pour activer les sessions.

Les cookies

Travailler avec des cookies fonctionne à peu près de la même façon qu'avec des sessions, à quelques petites différences près que nous allons voir. Voici ce que nous allons faire pour découvrir les cookies :

1. On va voir ce que c'est un cookie exactement... parce que si ça se trouve il y en a qui croient en ce moment même que je vais parler de recettes de cuisine ! 😊
2. Ensuite, nous verrons comment **écrire un cookie**. C'est facile à faire, si on respecte quelques règles.
3. Enfin, nous verrons comment **récupérer le contenu d'un cookie**. Ce sera le plus simple. 😊

Qu'est-ce qu'un cookie ?

Un cookie, c'est un petit fichier que l'on enregistre sur l'ordinateur du visiteur.

Ce fichier contient du texte et permet de "retenir" des informations sur le visiteur. Par exemple, vous inscrivez dans un cookie le pseudo du visiteur, comme ça la prochaine fois qu'il viendra sur votre site vous pourrez lire son pseudo en allant regarder ce que son cookie contient.

Parfois les cookies ont une mauvaise image. On fait souvent l'erreur de penser que les cookies sont "dangereux". Non, ce ne sont pas des virus, juste 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 c'est tout, ces petites bêtes sont inoffensives pour votre ordinateur.

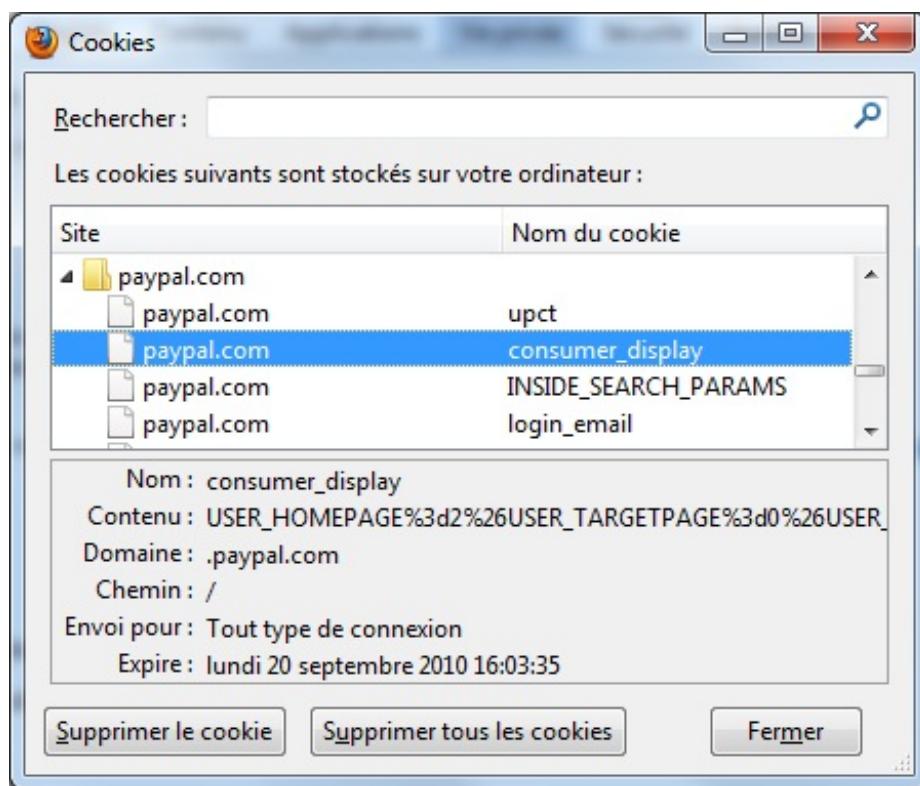
Chaque cookie stocke généralement une information à la fois. Si vous voulez stocker le pseudonyme du visiteur et sa date de naissance, il est donc recommandé de créer 2 cookies.



Où sont stockés les cookies sur mon disque dur ?

Cela dépend de votre navigateur web. Généralement on ne touche pas directement à ces fichiers, mais on peut afficher à l'intérieur du navigateur la liste des cookies qui sont stockés. On peut choisir de les supprimer à tout moment.

Si vous avez Mozilla Firefox, vous pouvez aller dans le menu Outils / Options / Vie privée et cliquer sur Supprimer des cookies spécifiques. Vous obtenez la liste et la valeur de tous les cookies stockés :



Les cookies sont classés par site web. Chaque site web peut écrire comme vous le voyez plusieurs cookies. Chacun d'eux a un nom et une valeur (que vous pouvez voir à la ligne Contenu sur ma capture). Vous noterez que comme tout cookie qui se respecte, ils ont chacun une date d'expiration. Après cette date, ~~ils ne sont plus bons à manger~~ ils sont automatiquement supprimés par le navigateur.

Les cookies sont donc des informations temporaires que l'on stocke sur l'ordinateur des visiteurs. La taille est limitée à quelques Ko, vous ne pouvez pas stocker beaucoup d'informations à la fois, mais c'est en général suffisant.

Ecrire un cookie

Comme une variable, un cookie a un nom et une 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`).

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

Si vous voulez supprimer le cookie dans un an, il vous faudra donc écrire : `time() + 365*24*3600`. Cela veut dire : timestamp actuel + nombre de secondes dans une année. Cela aura pour effet de supprimer votre cookie dans exactement un an.

Voici donc comment on peut créer un cookie :

Code : PHP

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

Sécuriser son cookie avec le mode httpOnly

Je recommande toutefois d'activer l'option `httpOnly` sur le cookie. Sans rentrer dans les détails, cela rendra votre cookie inaccessible en javascript sur tous les navigateurs qui supportent cette option (c'est le cas de tous les navigateurs récents). Cette option permet de réduire drastiquement les risques de faille XSS sur votre site, au cas où vous ayez oublié d'utiliser `htmlspecialchars` à un moment.

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

Code : PHP

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

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



Les paramètres au milieu sont des paramètres que nous n'utilisons pas, je leur ai donc envoyé `null`.

Créer le cookie avant d'écrire du HTML

Il y a un petit problème avec `setcookie`... Comme pour `session_start`, cette fonction ne marche **que** si vous lappelez avant tout code HTML (donc avant la balise `<!DOCTYPE>`).



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  
setcookie('pseudo', 'M@teo21', time() + 365*24*3600, null, null,  
false, true); // On écrit un cookie  
setcookie('pays', 'France', time() + 365*24*3600, null, null, false,  
true); // On écrit un autre cookie...  
  
// Et SEULEMENT MAINTENANT, on peut commencer à écrire du code html  
?>  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/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 ! 😊

Comme vous le voyez, pour écrire 2 cookies il faut appeler 2 fois `setcookie`.

Afficher un cookie

C'est la partie la plus simple. Avant de commencer à travailler sur une page, PHP lit les cookies du client pour récupérer toutes les informations qu'ils contiennent. Ces informations sont placées dans la superglobale `$_COOKIE`, sous forme d'array comme d'habitude.

De ce fait, si je veux ressortir le pseudo du visiteur que j'avais inscrit dans un cookie, il suffit 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 ça ?
</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. Il faut donc faire un `isset` pour vérifier si le cookie existe ou non.



Les cookies viennent du visiteur. Comme toute information qui vient du visiteur, elle **n'est pas sûre**. N'importe quel visiteur peut créer des cookies et envoyer ainsi de fausses informations à votre site. Souvenez-vous en lorsque vous lisez les cookies du visiteur : il peut les avoir modifiés, donc soyez prudents et ne faites pas une confiance aveugle au contenu de ces cookies !

Modifier un cookie existant

Vous vous demandez peut-être comment modifier un cookie déjà existant ? C'est là encore très simple : il faut refaire appel à `setcookie` en gardant le même nom de cookie. Cela "écrasera" l'ancien cookie.

Par exemple, si j'habite maintenant en Chine, je ferai :

Code : PHP

```
<?php setcookie('pays', 'Chine', time() + 365*24*3600, null, null,
```

```
false, true); ?>
```

Notez qu'alors le temps d'expiration du cookie est remis à zéro pour un an.

Vous connaissez maintenant tout ce qu'il faut savoir sur les superglobales ! Avec les cookies et les sessions, vous avez maintenant entre les mains 2 outils très puissants pour retenir des informations de page en page.

Lire et écrire dans un fichier

Les variables sont simples à utiliser, mais ce sont des informations temporaires. La durée de vie d'une variable n'est en effet jamais très longue. Or, vous aurez certainement besoin sur votre site de stocker des informations définitivement.

Par exemple, il est impossible de stocker les messages d'un forum dans des variables... puisque celles-ci seront supprimées à la fin de l'exécution de la page ! Pour stocker ces informations longtemps, il faut les écrire sur le disque dur. Quoi de plus logique pour cela que de créer des fichiers ?

PHP permet justement d'enregistrer des données dans des fichiers sur le disque dur du serveur.

Autoriser l'écriture de fichiers (chmod)

Pour que PHP puisse créer des fichiers, il doit avoir accès à un dossier qui autorise la création de fichiers par PHP. Il faut en effet donner le droit à PHP de créer et modifier les fichiers, sinon celui-ci ne pourra rien faire.

Pour créer ces droits, on dit en général qu'on doit modifier le CHMOD du fichier ou du dossier. C'est le nom de la commande qui permet de modifier les droits sous Linux.



Sous Windows, vous n'en avez probablement jamais entendu parler, tout simplement parce que ça n'existe pas. Mais le serveur de votre site, lui, est le plus souvent sous Linux. Et sous Linux, on utilise ce qu'on appelle le CHMOD pour gérer les droits.

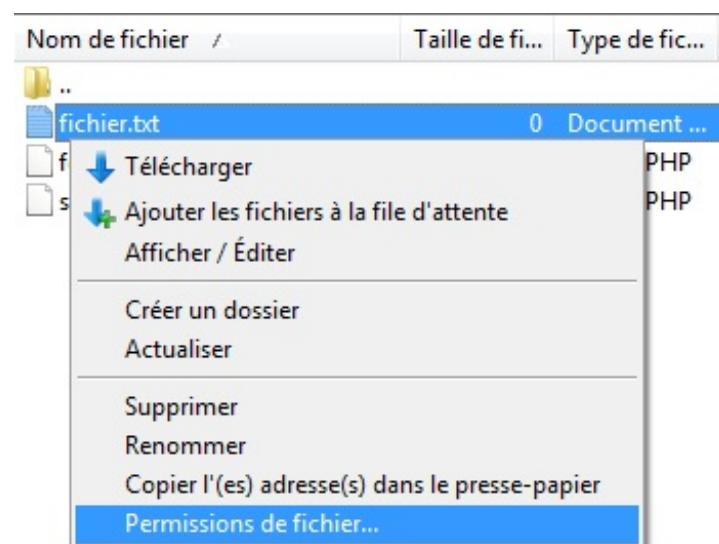
Le CHMOD est un nombre à 3 chiffres que l'on attribue à un fichier (par exemple 777). Selon la valeur de ce nombre, Linux autorisera (ou non) la modification du fichier.

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 ! 😊

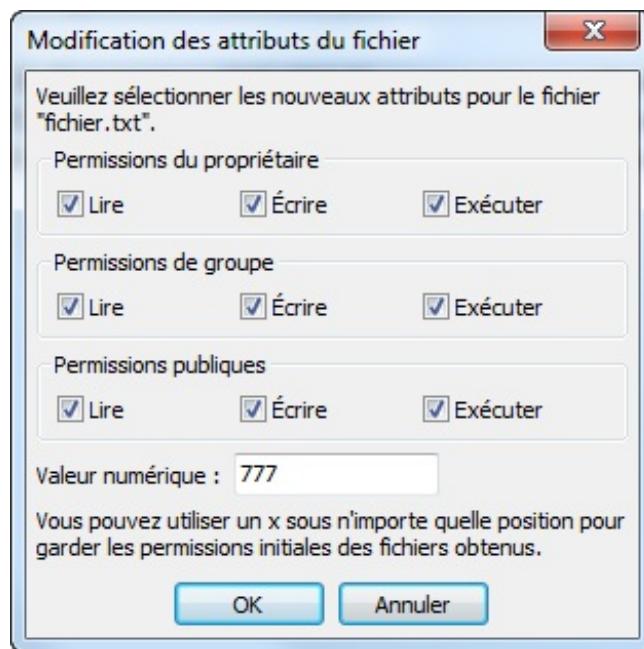
Il va falloir passer par... votre logiciel FTP ! Oui, celui-là même qui vous sert à envoyer vos pages sur le web. 😊

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 "Permissions de 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^e 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 (par défaut 5, il faut mettre cette valeur à 7).

Si vous rentrez 777 comme valeur pour le CHMOD, ça veut dire que tous les programmes du serveur ont le droit de modifier le fichier, notamment PHP. **Il faut donc rentrer 777 pour que PHP puisse modifier le fichier en question.**



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.

Pour ceux qui veulent en savoir plus sur les CHMOD, je traite le sujet beaucoup plus en détail dans mon cours sur Linux (car c'est un truc de Linux). N'hésitez pas à aller lire [le chapitre](#) si le sujet vous intéresse.

Ouvrir et fermer le fichier

Avant de lire/écrire dans un fichier, il faut d'abord l'ouvrir.

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 un fichier PHP qui va travailler sur `compteur.txt`.

Votre mission, si vous l'acceptez : compter le nombre de fois qu'une page a été vue sur votre site et enregistrer ce nombre dans ce fichier.

Voici comment on va procéder :

Code : 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. Nous allons voir comment ça fonctionne un peu plus loin.
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 absolument pas obligés de donner l'extension `.txt` à votre fichier. Vous pouvez l'appeler comme vous voulez : `compteur.cpt`, `compteur.num`, ou même `compteur` tout court.

Lire et écrire dans le fichier

Maintenant que nous savons ouvrir et fermer notre fichier, nous allons apprendre à le lire et à le modifier.

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 procéder comme ceci :

Code : 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 `$monfichier` qui lui permettra de l'identifier. `fgets` renvoie toute la ligne (la fonction arrête la lecture au premier saut de ligne). Donc, notre variable `$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. Au second tour de boucle, le prochain appel à `fgets` renverra la deuxième ligne, et ainsi de suite.

C'est un peu lourd, mais si on stocke assez peu d'informations dans le fichier cela peut suffire. Sinon, si on a beaucoup d'informations à stocker, on préférera utiliser une base de données (on en parlera dans la prochaine partie).

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 :

Code : PHP

```
<?php 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).

38472
↑

Si vous faites un `fputs` juste après, il va écrire à la suite ! Pour éviter ça, on va utiliser la fonction `fseek` qui va remplacer le curseur où on veut dans le fichier. En l'occurrence, on va remplacer le curseur au début du fichier en faisant : `fseek($monfichier, 0);`

Notre curseur sera alors repositionné au début :

38472
↑



Si vous avez ouvert le fichier avec le mode '`a`' ou '`a+`', toutes les données que vous écrirez seront *toujours* ajoutées à la fin du fichier. La fonction `fseek` n'aura donc aucun effet dans ce cas.

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).

Pour y voir un peu plus clair, je vous propose ce code source qui compte le nombre de fois que la page a été vue :

Code : 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 !

Ce n'était pas si dur, vous voyez. 😊

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 absolument 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 cela devient vite compliqué, mais pour un petit fichier comme celui-ci, cela convient 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. Si vous avez beaucoup d'informations à stocker, il est préférable de faire appel à une base de données. Nous allons justement découvrir les bases de données dans la prochaine partie. 

Partie 3 : Stocker des informations dans une base de données

En PHP, on peut difficilement se passer d'une base de données. Cet outil incontournable sert à enregistrer des données de façon efficace et organisée.

Tout ce que vous voulez *enregistrer* sur votre site va se retrouver stocké dans une base de données : liste des membres, messages des forums, news, etc.

Présentation des bases de données

Jusqu'ici, vous avez découvert le fonctionnement du langage PHP mais vous n'avez probablement pas encore le sentiment que vous êtes capables de créer de vrais sites web avec ce que vous avez appris. C'est parfaitement normal, car il vous manque un élément crucial : la base de données.

Une base de données permet d'enregistrer des données de façon organisée et hiérarchisée. Vous connaissez certes les variables, mais celles-ci restent en mémoire seulement le temps de la génération de la page. Vous avez aussi appris à écrire dans des fichiers, mais cela devient vite très compliqué dès que vous avez beaucoup de données à enregistrer.

Or, il va bien falloir stocker quelque part la liste de vos membres, les messages de vos forums, les options de navigation des membres... Les bases de données sont le meilleur moyen de faire cela de façon simple et propre. Nous allons les étudier durant toute cette partie du cours !

Le langage SQL et les bases de données

La **base de données** (BDD) est un 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 peut enregistrer dans un fichier comme on a appris à le faire... mais quand on a goûté aux bases de données on peut difficilement s'en passer ensuite !

Imaginez par exemple une armoire, dans laquelle chaque dossier est à sa place.

Quand tout est 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.

Les SGBD s'occupent du stockage

Je vous ai présenté brièvement les SGBD (Systèmes de Gestion de Base de Données) dans le premier chapitre du cours. Les SGBD sont les programmes qui se chargent du stockage de vos données.

Les plus connus sont, pour rappel :

- MySQL : libre et gratuit, c'est probablement le SGBD le plus connu. Nous l'utiliserons dans ce cours.
- PostgreSQL : libre et gratuit comme MySQL, avec plus de fonctionnalités mais un peu moins connu.
- SQLite : libre et gratuit, très léger mais très limité en fonctionnalités.
- Oracle : utilisé par les très grosses entreprises, sans aucun doute un des SGBD les plus complets mais il n'est pas libre et on le paie le plus souvent très cher.
- Microsoft SQL Server : le SGBD de Microsoft.

Il faut donc choisir le SGBD que vous allez utiliser pour stocker les données. Je vous recommande de travailler plutôt avec les SGBD libres et gratuits, tels que MySQL, PostgreSQL et SQLite. Après, tout est question de goût et des fonctionnalités que vous recherchez. MySQL est un bon compromis.



Nous allons utiliser MySQL mais sachez que l'essentiel de ce que vous allez apprendre fonctionnera de la même manière avec un autre SGBD. Ce cours est construit afin que vous ayez le moins de choses possible à réapprendre si vous choisissez de changer de SGBD.

Vous donnez les ordres au SGBD en langage SQL

Vous allez devoir communiquer avec le SGBD pour lui donner l'ordre de récupérer ou d'enregistrer des données. Pour "parler" avec le SGBD, on utilise le langage SQL.

La bonne nouvelle, c'est que le langage SQL est un standard, c'est-à-dire que quel que soit le SGBD que vous utilisez vous utiliserez le langage SQL. La mauvaise, c'est qu'il y a en fait quelques petites variantes d'un SGBD à l'autre, mais cela concerne généralement les commandes les plus avancées.

Comme vous vous en doutez, il va falloir apprendre le langage SQL pour travailler avec les bases de données. Ce langage n'a

rien à voir avec le PHP, mais nous allons impérativement en avoir besoin.

Voici un exemple de commande en langage SQL pour vous donner une idée :

Code : SQL

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

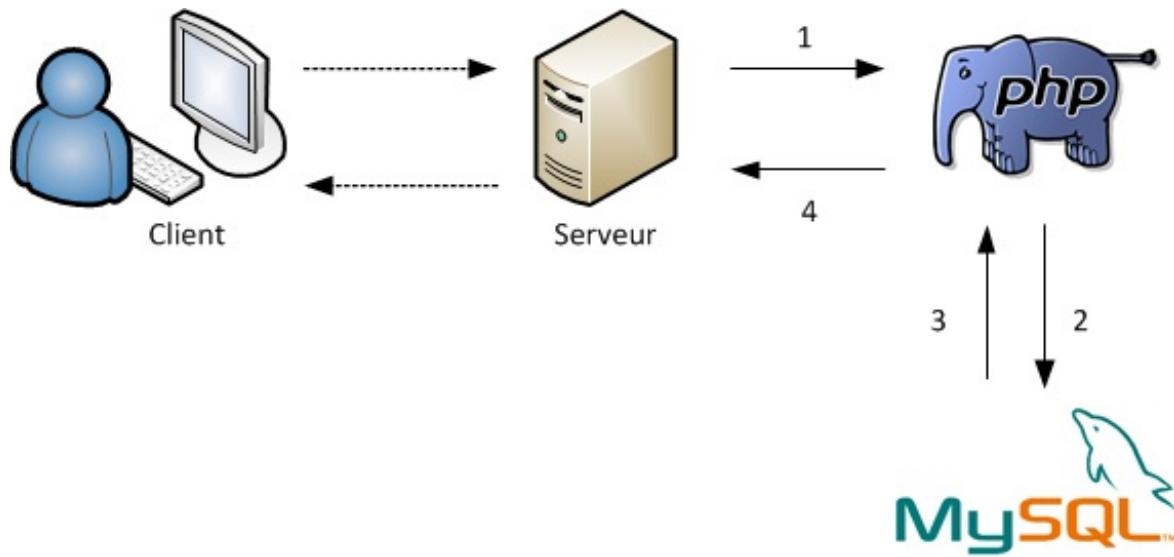
Le principal objectif de cette partie du cours sera d'apprendre à utiliser ce langage SQL pour que vous soyez capables de donner n'importe quel ordre à la base de données, comme par exemple : "Récupère-moi les 10 dernières news de mon site", "Supprime le dernier message posté dans ce forum", etc.

PHP fait la jonction entre vous et MySQL

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...



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 qui lui était demandé.

Maintenant que nous avons fait les présentations, il va falloir découvrir comment est organisée une base de données. Bien comprendre l'organisation d'une base de données est en effet absolument indispensable.

Structure d'une base de données

Avec les bases de données, il faut utiliser un vocabulaire **précis**. Heureusement, vous ne devriez pas avoir trop de mal à vous en souvenir, vu qu'on va se servir d'une image : celle d'une armoire. Ecoutez-moi attentivement, et n'hésitez pas à lire lentement, plusieurs fois si c'est nécessaire.

Je vous demande d'imaginer ceci :

- L'armoire est appelée **la base** dans le langage SQL. C'est le gros meuble dans lequel les secrétaires ont l'habitude de classer les informations.
- Dans une armoire, il y a plusieurs tiroirs. Un tiroir, en SQL, c'est ce qu'on appelle **une table**. Chaque tiroir contient des données différentes. Par exemple, on peut imaginer un tiroir qui contient les pseudonymes et 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**.

Une table est donc représentée sous la forme d'un tableau ; 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
...

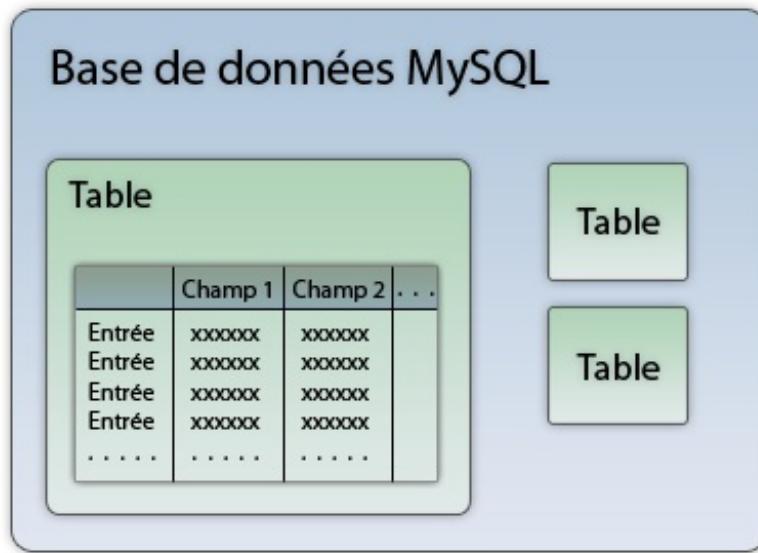
Ce tableau représente le contenu d'une table (c'est-à-dire le tiroir de l'armoire).

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" (identifiant). 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 :



La base de données contient plusieurs tables (on peut en mettre autant que l'on veut à l'intérieur). Chaque table est en fait un tableau où les colonnes sont appelées champs et où les lignes sont appelées entrées.

Pour vous donner quelques exemples concrets, voici quelques noms de tables que l'on peut être amenées à créer pour les besoins de son 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 tous seuls sur un bout de papier.

Mais où sont enregistrées les données ?

Avant de terminer le chapitre, voici une question que l'on se pose fréquemment quand on lit ce genre de chapitres sur les bases de données pour la première fois.

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 ! 😊



Ils sont bien jolis ces tableaux et ces schémas, ces bases, ces champs... 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 la base de données gère les informations sous forme de 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** !

Ces fichiers sont quelque part sur votre disque dur mais il ne faut jamais les ouvrir et encore moins les modifier directement. Il faut toujours parler avec MySQL qui va se charger d'extraire et de modifier les informations dans ces fichiers.

Chaque SGBD a sa propre façon d'enregistrer les données, mais aucun d'eux ne peut y échapper : pour que les données restent enregistrées, il faut les stocker dans des fichiers sur le disque dur.



Par exemple, avec MySQL sous Windows si vous utilisez WAMP, vous devriez trouver les fichiers où sont stockées les informations dans C :\wamp\mysql\data. Je vous recommande très fortement de ne pas y toucher car ce n'est pas prévu pour être modifié directement !

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.

Il existe plusieurs façons d'accéder à sa base de données et d'y faire des modifications. On peut utiliser une ligne de commande (console), exécuter les requêtes en PHP ou faire appel à un programme qui nous permet d'avoir rapidement une vue d'ensemble. Ici je vous propose de découvrir phpMyAdmin, un des outils les plus connus permettant de manipuler une base de données MySQL.

phpMyAdmin est livré avec WAMP, 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. Vous aurez très certainement besoin d'un login et d'un mot de passe.

Créer une table

La première chose que je vous demanderai de faire, c'est d'ouvrir phpMyAdmin. Pour cela, démarrez WAMP, faites un clic gauche sur l'icône de la barre des tâches et allez dans "phpMyAdmin". Vous y êtes 😊



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.

L'accueil de phpMyAdmin ressemble à ceci :

La capture d'écran montre l'interface de phpMyAdmin. À gauche, une liste de bases de données est affichée, incluant "information_schema (28)" et "mysql (23)". Un numéro "(1)" est placé près de cette liste. Au centre, une section "Actions" indique "MySQL localhost". Une sous-section "Créer une base de données" contient un champ de saisie "(2)" où l'on peut taper le nom de la nouvelle base. Ensuite, il y a des options pour l'interclassement et pour la connexion MySQL. En bas, une section "Interface" permet de configurer la langue (Français - French), le thème (Original), la couleur au choix et la taille du texte (82%).

Vous avez 2 endroits importants, signalés par des numéros sur ma capture d'écran :

- Liste des bases** : c'est la liste de vos bases de données. Le nombre entre parenthèses, c'est le nombre de tables qu'il y a dans la base.
Sur ma capture d'écran, on a donc 2 bases : `information_schema`, qui contient 28 tables, et `mysql`, qui contient 23 tables.
- Créer une base** : pour créer une nouvelle base de données, tapez un nom dans le champ de formulaire à droite, cliquez sur "Créer" et hop ! C'est fait. 😊

Pour le moment, 2 bases sont déjà créées : "`information_schema`" et "`mysql`". Ne touchez pas à ces bases, elles servent au fonctionnement interne de `mysql`.

Nous allons créer une nouvelle base "`test`" dans laquelle nous travaillerons dans toute la suite du cours. Utilisez le formulaire à droite pour créer cette base : rentrez le nom `test` et cliquez sur le bouton `Créer`.

L'écran suivant devrait alors s'afficher si la base a bien été créée :

The screenshot shows the phpMyAdmin interface. At the top, it says "Serveur: localhost" and "Base de données: test". Below that, there are tabs for "Structure", "SQL", "Rechercher", "Requête", "Exporter", "Importer", "Opérations", "Priviléges", and "Supprimer". A green message box at the top right says "La base de données test a été créée." with a checkmark. Below it, the SQL command "CREATE DATABASE 'test'" is shown. At the bottom right of the message box are "[Modifier]" and "[Créer source PHP]". On the left, under "Base de données", "test (0)" is selected. Below it, a message says "Aucune table n'a été trouvée dans cette base." In the center, there's a form titled "Créer une nouvelle table sur la base test" with fields for "Nom:" (set to "news") and "Nombre de champs:" (set to "3"). A "Exécuter" button is at the bottom right of the form.

On vous indique qu'aucune table n'a été trouvée dans la base. Et si on en créait une ? 😊

Dans le champ "Créer une nouvelle table sur la base test", indiquez le nom news et le nombre de champs 3 :

The screenshot shows the "Créer une nouvelle table sur la base test" form. It has two input fields: "Nom:" containing "news" and "Nombre de champs:" containing "3". Below the form is a large empty text area.

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. Je vous propose de faire simple car on cherche juste à tester phpMyAdmin ici. On va créer les 3 champs suivants pour cette table :

- **id** : comme bien souvent, vous allez devoir créer un champ appelé **id** (prononcez à l'anglaise "aille di"). 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. Pour le moment nous cherchons seulement à découvrir le fonctionnement de phpMyAdmin.

Vous devriez avoir ceci sous les yeux :

Champ	<input type="text" value="id"/>	titre	<input type="text" value="contenu"/>
Type <small>?</small>	<input type="text" value="INT"/> <small>▼</small>	<input type="text" value="VARCHAR"/> <small>▼</small>	<input type="text" value="TEXT"/> <small>▼</small>
Taille/Valeurs ^{*1}	<input type="text" value=""/>	<input type="text" value="255"/>	<input type="text" value=""/>
Défaut ²	<input type="text" value="Aucun"/> <small>▼</small>	<input type="text" value="Aucun"/> <small>▼</small>	<input type="text" value="Aucun"/> <small>▼</small>
Interclassement	<input type="text" value=""/> <small>▼</small>	<input type="text" value=""/> <small>▼</small>	<input type="text" value=""/> <small>▼</small>
Attributs	<input type="text" value=""/> <small>▼</small>	<input type="text" value=""/> <small>▼</small>	<input type="text" value=""/> <small>▼</small>
Null	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index	<input type="text" value="PRIMARY"/> <small>▼</small>	<input type="text" value="---"/> <small>▼</small>	<input type="text" value="---"/> <small>▼</small>
AUTO_INCREMENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Commentaires	<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>

Chaque colonne représente un champ. Nous avons demandé 3 champs, il y a donc 3 colonnes.

phpMyAdmin vous demande beaucoup d'informations mais rassurez-vous, il n'est pas nécessaire de tout remplir. La plupart du temps, les sections les plus intéressantes seront :

- **Champ** : permet de définir le nom du champ (important !)
- **Type** : le type de données que va stocker le champ (nombre entier, texte, date...)
- **Taille/Valeurs** : permet d'indiquer la taille maximale du champ, utile pour le type VARCHAR notamment afin de limiter le nombre de caractères autorisés.
- **Index** : active l'indexation du champ. Ce mot barbare signifie dans les grandes lignes que votre champ sera adapté aux recherches. Le plus souvent, on utilise l'index PRIMARY sur les champs de type `id`.
- **AUTO_INCREMENT** : permet au champ de s'incrémenter tout seul à chaque nouvelle entrée. On l'utilise fréquemment sur les champs de type "id".

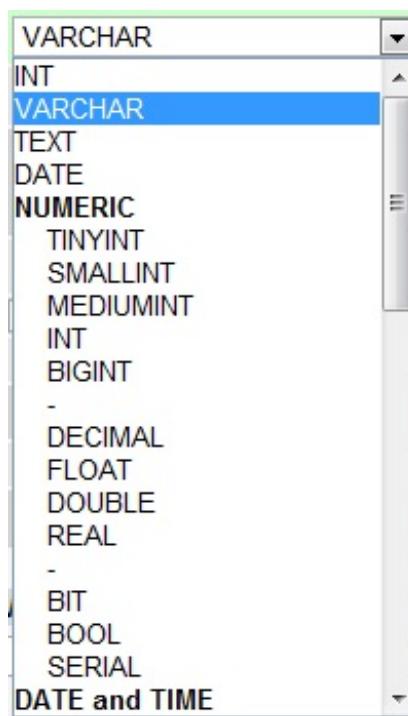
Je vous propose de remplir le formulaire comme je l'ai fait. Veillez à bien cocher AUTO_INCREMENT et à définir un index PRIMARY sur le champ `id`.

Une fois que c'est fait, cliquez sur le bouton **Sauvegarder** en bas de la page. Votre table est créée !

Avant d'aller plus loin, je voudrais revenir un peu plus en détail sur les types de champs et les index, notamment l'index PRIMARY qu'on a utilisé.

Les types de champs MySQL

Si vous déroulez la liste des types que vous propose MySQL, vous devriez tomber à la renverse :



Alors que PHP ne propose que quelques types de données que l'on connaît bien maintenant (`int, string, bool...`), MySQL propose une quantité très importante de types de données.

En fait, ceux-ci sont classés par catégories :

- NUMERIC : ce sont les nombres. On y trouve des types dédiés aux petits nombres entiers (TINYINT), aux gros nombres entiers (BIGINT), aux nombres décimaux, etc.
- DATE and TIME : ce sont les dates et les heures. De nombreux types différents permettent de stocker une date, une heure, ou les deux à la fois.
- STRING : ce sont les chaînes de caractères. Là encore, il y a des types adaptés à toutes les tailles.
- SPATIAL : cela concerne les bases de données spatiales, utile pour ceux qui font de la cartographie. Ce ne sera pas notre cas, donc nous n'en parlerons pas ici.

En fait, phpMyAdmin a eu la bonne idée de proposer au tout début de cette liste les 4 types de données les plus courants :

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

Nous n'aurons besoin de jongler qu'entre ces 4 types, donc ce sont eux qu'il faut retenir. Cela couvrira 99% des besoins. Vous pouvez aussi garder en tête le type DOUBLE qui permet de stocker des nombres décimaux.

 Une petite remarque à propos de VARCHAR : c'est un type adapté aux courts textes, comme le titre d'une news de votre site. Sa seule exigence est que vous devez indiquer la taille maximale du champ (entre 1 et 255). Si vous ne le faites pas, vous ne pourrez pas créer la table. Si vous ne savez pas à combien limiter votre champ, vous pouvez mettre la valeur maximale (255) comme je l'ai fait dans l'exemple précédent.

Les clés primaires

Toute table doit posséder un champ qui joue le rôle de clé primaire. La clé primaire permet d'identifier de manière unique une

entrée dans la table. En général, on utilise le champ "id" comme clé primaire comme on vient de le faire.

Chaque news de votre site doit pouvoir être identifiée de manière unique. Le moyen le plus simple pour cela est de lui donner un numéro unique, dans un champ nommé "id". **Il ne peut pas y avoir 2 news avec le même id !** Il en irait de même pour les autres tables de votre site : par exemple, chaque membre de votre site doit se voir attribuer un numéro unique. Si 2 membres ont le même numéro, on ne pourra pas les différencier !

Il est vital que chaque table possède sa clé primaire. On ne vous interdira pas de créer des tables sans clé primaire, mais leur performances seront extrêmement réduites. Je vous conseille donc de prendre le réflexe de créer à chaque fois ce champ "id" en lui donnant l'index PRIMARY, ce qui aura pour effet d'en faire une clé primaire. Vous en profiterez en général pour cocher la case AUTO_INCREMENT afin que ce champ gère lui-même les nouvelles valeurs automatiquement (1, 2, 3, 4...).

Modifier une table

A gauche de votre écran, la table "news" que vous venez de créer devient visible :

- Si vous cliquez sur le mot "news", le contenu de la table s'affiche à droite de l'écran.
- Si vous cliquez sur la petite image de tableau à gauche, phpMyAdmin vous présentera la structure de la table.

Actuellement, comme notre table est vide (elle ne contient aucune entrée), c'est la structure de la table qui s'affichera dans les deux cas :

	Champ	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
<input type="checkbox"/>	id	int(11)			Non	Aucun	auto_increment	
<input type="checkbox"/>	titre	varchar(255)	latin1_swedish_ci		Non	Aucun		
<input type="checkbox"/>	contenu	text	latin1_swedish_ci		Non	Aucun		

Tout cocher / Tout décocher Pour la sélection :

Version imprimable Suggérer des optimisations quant à la structure de la table

Ajouter champ(s) En fin de table En début de table Après Exécuter

Ce tableau vous rappelle de quels champs est constituée votre table : c'est sa structure. Notez que le champ `id` est souligné car c'est la clé primaire de la table.

Il n'y a rien de bien intéressant à faire ici, mais sachez qu'il est possible d'ajouter ou de supprimer des champs à tout moment. Ce n'est pas parce que votre table a été créée qu'elle est figée. Vous avez des options pour renommer les champs, les supprimer, en ajouter, etc.

Jetez déjà un oeil aux onglets en haut : "Structure", "Afficher", "SQL", etc. Cela vous amènera vers différentes options que nous verrons plus loin. Nous allons commencer par nous intéresser à l'onglet "Insérer" qui va nous permettre d'ajouter des entrées à la table.

Une page s'ouvre dans laquelle vous pouvez rentrer des valeurs pour chacun des champs. Cela va être pour nous l'occasion d'insérer notre première news :

Champ	Type	Fonction	Null	Valeur
id	int(11)	<input type="text"/>		<input type="text"/>
titre	varchar(255)	<input type="text"/>		<input type="text" value="Ma première news"/>
contenu	text	<input type="text"/>		<input type="text" value="Vous êtes en train de lire ma première news. Bravo !"/>

Exécuter

Ignorer

Champ	Type	Fonction	Null	Valeur
id	int(11)	<input type="text"/>		<input type="text"/>
titre	varchar(255)	<input type="text"/>		<input type="text"/>

Seule la colonne "Valeur" nous intéresse. Vous pouvez rentrer une valeur pour chacun des trois champs. Vous remarquerez que je n'ai pas mis de valeur pour l'id : c'est normal, le numéro d'id est automatiquement calculé grâce à l'option auto_increment. Ne vous en occupez pas et choisissez plutôt un titre puis inscrivez un contenu. L'id de la première news sera 1, celui de la seconde sera 2, etc.



Les id ne doivent pas obligatoirement se suivre de 1 en 1. S'il n'y a pas de news n°15 par exemple, cela ne pose aucun problème. Ce qui compte, c'est qu'il n'y ait pas deux news avec le même id. C'est d'ailleurs justement ce que permet d'éviter la clé primaire : elle interdit que deux entrées aient le même id.

Une fois que vous avez rentré le texte que vous vouliez, cliquez sur le premier bouton "Exécuter" de la page.



Il y a d'autres champs en-dessous : ignorez-les. Ils vous permettent d'ajouter plusieurs entrées à la fois mais nous n'en avons pas besoin.

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

Affichez maintenant le contenu de la table. Vous pouvez soit cliquer sur l'onglet "Afficher" en haut soit sur le nom de la table dans le menu à gauche.

	id	titre	contenu
<input type="checkbox"/>	1	Ma première news	Vous êtes en train de lire ma première news. Bravo...
<input type="checkbox"/>	2	Autre news	Ceci est une autre news !
<input type="checkbox"/>	3	Exclusif !	Ceci est une news !

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. 😊

Vous pouvez modifier ou supprimer chacun des éléments que vous voyez à l'écran. Il y a beaucoup d'autres options en-dessous que je vous laisse regarder. Pour l'instant, ce qui compte, c'est que vous ayez compris la procédure pour ajouter des éléments à la table et que vous soyez capables de lister le contenu de la table.



Mais... Je ne vais pas devoir passer par phpMyAdmin à chaque fois que je veux ajouter ou supprimer un élément quand même ? Il faudra passer par là pour ajouter chaque news de son site, mais aussi chaque membre, chaque

 message des forums ?

Non, bien sûr que non. Comme son nom l'indique, phpMyAdmin est un outil d'administration. Il permet de voir rapidement la structure et le contenu de vos tables. Il est aussi possible d'ajouter ou de supprimer des éléments comme on vient de le voir, mais on ne le fera que très rarement. Nous apprendrons à créer des pages en PHP qui insèrent ou suppriment des éléments directement depuis notre site web.

Il nous reste encore à découvrir quelques-unes des fonctionnalités offertes par phpMyAdmin et nous aurons terminé notre tour d'horizon de cet outil.

Autres opérations

Nous avons jusqu'ici découvert le rôle de 3 onglets :

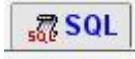
- Afficher : affiche le contenu de la table
- Structure : présente la structure de la table (liste des champs)
- Insérer : permet d'insérer de nouvelles entrées dans la table

Je souhaite vous présenter 6 autres onglets que nous n'avons pas encore découverts :

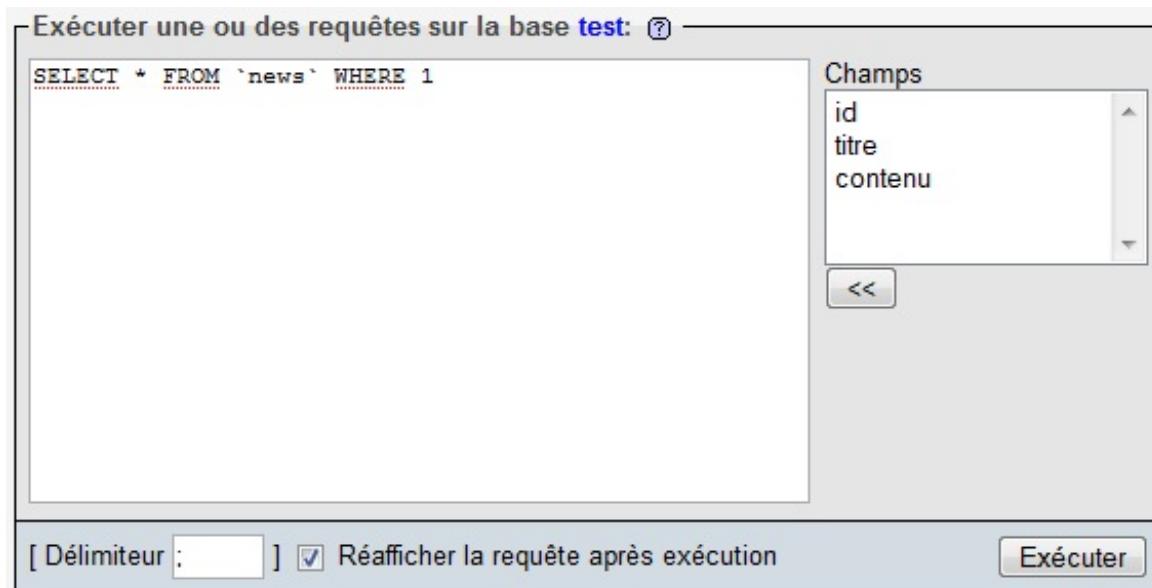
- SQL
- Importer
- Exporter
- Opérations
- Vider
- Supprimer

SQL

Cliquez sur l'onglet "SQL" :



Il s'affiche à l'écran :



Exécuter une ou des requêtes sur la base test: ⑦

```
SELECT * FROM `news` WHERE 1
```

Champs

id
titre
contenu

<<

[Délimiteur :] Réafficher la requête après exécution

C'est ici que vous pouvez exécuter ce que l'on appelle des requêtes SQL pour demander à MySQL de faire quelque chose.

Dans la grande zone de texte, vous pouvez taper des requêtes SQL. Par exemple on nous propose ici :

Code : SQL

```
SELECT * FROM `news` WHERE 1
```

Cela signifie : "Afficher tout le contenu de la table 'news'". C'est justement ce langage SQL que nous allons découvrir tout au long des prochains chapitres.

Notez qu'il est aussi possible d'écrire des requêtes SQL dans une nouvelle fenêtre. Pour ouvrir une nouvelle fenêtre de requête SQL, cliquez sur le bouton SQL en haut du menu à gauche :



Cette nouvelle fenêtre se révèlera souvent très pratique.

Importer

Il y a aussi un onglet "Importer" :



Dans la page qui s'affiche, vous pouvez envoyer un fichier de requêtes SQL (généralement un fichier .sql) à MySQL pour qu'il les exécute :

Fichier à importer

Emplacement du fichier texte (Taille maximum: 2 048 Kio)

Jeu de caractères du fichier:

Ces modes de compression seront détectés automatiquement : aucune, gzip, zip

Importation partielle

Permettre l'interruption de l'importation si la limite de temps est sur le point d'être atteinte. Ceci pourrait aider à importer des fichiers volumineux, au détriment du respect des transactions.

Nombre d'enregistrements (requêtes) à ignorer à partir du début

Format du fichier d'importation

CSV

CSV via LOAD DATA

SQL

Options

Mode de compatibilité SQL

Seul le premier champ en haut devrait nous intéresser : il nous permet d'indiquer un fichier sur notre disque dur contenant des requêtes SQL à exécuter.

Cliquez ensuite sur le bouton "Exécuter" tout en bas sans vous préoccuper des autres champs.



Quelle différence y a-t-il entre écrire la requête SQL (comme on vient de le voir juste avant) et envoyer un fichier contenant des requêtes SQL ?

C'est la même chose, sauf que parfois quand on doit envoyer un très grand nombre de requêtes, il est plus pratique d'utiliser un fichier. Dans les prochains chapitres du cours d'ailleurs, je vous donnerai un fichier de requêtes à exécuter, et il faudra utiliser cette méthode.

Exporter

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 .sql (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:\wamp\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 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 lorsque vous voudrez héberger votre site sur internet, il faudra utiliser la base de données en ligne de votre hébergeur ! Le fichier .sql que vous allez générer vous permettra de *reconstruire* la base de données grâce à l'outil d'importation de phpMyAdmin (en général les hébergeurs proposent eux aussi phpMyAdmin pour que vous puissiez effectuer facilement des opérations sur votre base en ligne).
- **Faire une copie de sauvegarde de la base de données** : on ne sait jamais, si vous faites une bêtise ou si quelqu'un réussit à détruire toutes les informations sur votre site (dont la base de données), vous serez bien content d'avoir une copie de secours sur votre disque dur !

Votre écran doit ressembler à ceci :

Afficher le schéma de la table

Exporter

CodeGen
 CSV
 CSV pour MS Excel
 Microsoft Excel 2000
 Microsoft Word 2000
 LaTeX
 Tableur "Open Document"
 Texte "Open Document"
 PDF
 SQL
 Texte Texy!
 XML
 YAML

Options

Commentaires mis en en-tête (\n sépare les lignes)

Commentaires
 Utiliser le mode transactionnel
 Désactiver la vérification des clés étrangères

Mode de compatibilité SQL

Structure

Ajouter DROP TABLE
 Ajouter IF NOT EXISTS
 Inclure la valeur courante de l'AUTO_INCREMENT
 Protéger les noms des tables et des champs par des ""
 Ajouter CREATE PROCEDURE / FUNCTION / EVENT

Inclure sous forme de commentaires

Dates de création/modification/vérification

Données

Insertions complètes
 Insertions étendues
 Taille maximum de la requête générée
 Insertions avec délais (DELAYED)
 Ignorer les erreurs de doublons (INSERT IGNORE)
 Utiliser l'hexadécimal pour les BLOB

Type d'exportation

Exporte enregistrement(s) à partir du rang n° .

Transmettre

Modèle de nom de fichier¹ : (se souvenir du modèle)

Compression: aucune "zippé" "gzippé"

>

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.



Pensez à cocher la case "Transmettre" en bas, sinon il ne se passera rien. 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 contient plusieurs requêtes SQL. C'est ce langage que je vais vous apprendre dans les chapitres qui suivent !



Comment dois-je faire 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 "Importer" qu'on a vu tout à l'heure. Cliquez sur "Parcourir" pour indiquer où se trouve le fichier SQL que vous venez de créer 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).
- **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, à moins d'avoir fait une sauvegarde au préalable avec l'outil d'exportation.

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 ? Ni une ni deux, on retourne coder !

Lire des données

Nous retournons dans ce chapitre à nos pages PHP. Nous allons à partir de maintenant apprendre à communiquer avec une base de données via PHP. Ce sera l'occasion de découvrir le langage SQL, que nous étudierons tout au long des prochains chapitres.

Nous allons ici nous entraîner à lire des données dans une table. Il est vivement conseillé d'avoir un peu manipulé phpMyAdmin au préalable : cet outil sera le moyen pour vous de vérifier si les manipulations que vous faites en PHP ont bien l'impact que vous attendiez dans votre base de données.

Se connecter à la base de données en PHP

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

Nous allons apprendre dans ce chapitre à lire des données dans une BDD (base de données). 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'elle contient (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 toutes les opérations que vous voudrez sur votre base de données !

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

Bonne question ! En effet, PHP propose plusieurs moyens de se connecter à une base de données MySQL :

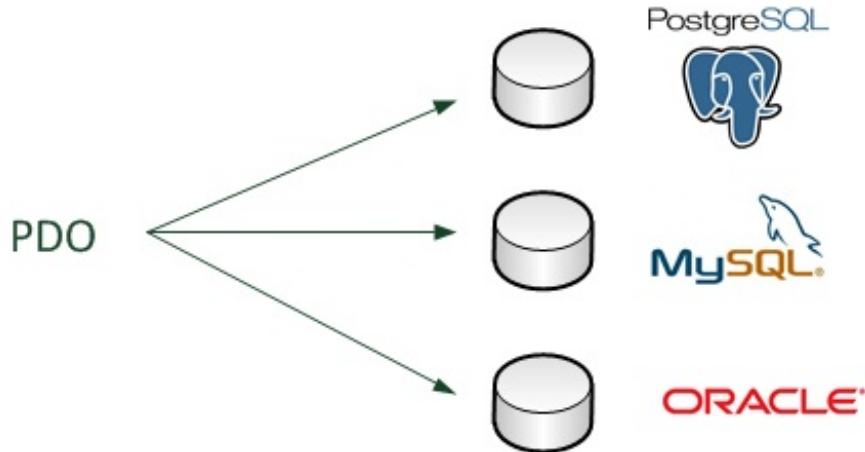
- L'extension `mysql_` : ce sont des fonctions qui permettent d'accéder à une base de données MySQL et donc de communiquer avec MySQL. Leur nom commence toujours par `mysql_`. Toutefois, ces fonctions sont vieilles et on recommande de ne plus les utiliser aujourd'hui.
- L'extension `mysqli_` : ce sont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

Ce sont toutes des extensions car PHP est très modulaire. On peut ajouter ou supprimer des éléments à PHP très facilement, car tout le monde n'a pas forcément besoin de toutes les fonctionnalités.



Quel moyen choisir parmi tous ceux-là ?

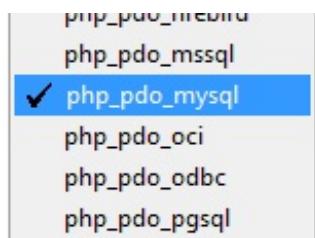
Vous l'aurez compris, les fonctions `mysql_` ne sont plus à utiliser (on dit qu'elles sont obsolètes). Il reste à choisir entre `mysqli_` et PDO. Nous allons ici utiliser PDO car c'est cette méthode d'accès aux bases de données qui va devenir la plus utilisée dans les prochaines versions de PHP. D'autre part, le gros avantage de PDO est que vous pouvez l'utiliser de la même manière pour vous connecter à n'importe quel autre type de base de données (PostgreSQL, Oracle...).



Vous pourrez donc réutiliser ce que vous allez apprendre si vous choisissez d'utiliser une autre base de données que MySQL.

Activer PDO

Normalement, PDO est activé par défaut. Pour le vérifier, faites un clic gauche sur l'icône de WAMP dans la barre des tâches, puis allez dans le menu PHP / Extensions PHP et vérifiez que `php_pdo_mysql` est bien coché.



Et si je n'utilise pas WAMP ?

Vous pouvez ouvrir le fichier de configuration de PHP (qui s'appelle généralement `php.ini`) et rechercher la ligne qui contient `php_pdo_mysql`. Enlevez le point-virgule devant s'il y en a un pour activer l'extension :

Code : Autre

```
;extension=php_pdo_firebird.dll
;extension=php_pdo_mssql.dll
extension=php_pdo_mysql.dll
;extension=php_pdo_oci.dll
;extension=php_pdo_odbc.dll
```

Si vous êtes sous Linux et que vous utilisez XAMPP, recherchez la ligne `pdo_mysql.default_socket` et complétez-la comme ceci

Code : Autre

```
pdo_mysql.default_socket = /opt/lampp/var/mysql/mysql.sock
```

Enregistrez le fichier puis redémarrez PHP. Il suffit pour cela de relancer votre logiciel favori, type WAMP, MAMP, XAMPP...

Se connecter à MySQL avec PDO

Maintenant que nous sommes certains que PDO est activé, nous pouvons nous connecter à MySQL. Nous allons avoir besoin de 4 renseignements :

- **Le nom de l'hôte** : c'est l'adresse de l'ordinateur où MySQL est installé (comme une adresse IP). Le plus souvent, MySQL est installé sur le même ordinateur que PHP : dans ce cas, mettez la valeur `localhost` (qui signifie "sur le même ordinateur"). Néanmoins, il est possible que votre hébergeur web vous indique une autre valeur à renseigner (par exemple `sql.hebergeur.com`). Dans ce cas il faudra modifier cette valeur lorsque vous enverrez votre site sur le web.
- **La base** : c'est le nom de la base de données à laquelle vous voulez vous connecter. Dans notre cas, la base s'appelle `test`. Nous l'avons créée avec phpMyAdmin dans le chapitre précédent.
- **Le login** : il 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** : il y a des chances pour que le mot de passe soit le même que celui que vous utilisez pour accéder au FTP. Renseignez-vous auprès de votre hébergeur.

Pour l'instant, nous faisons des tests sur notre ordinateur à la maison. On dit qu'on travaille "en local". Par conséquent, le nom de l'hôte sera `localhost`.

Quant au login et au mot de passe, par défaut le login est `root` et il n'y a pas de mot de passe.

Voici donc comment on doit faire pour se connecter à MySQL via PDO sur la base test :

Code : PHP

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



Je ne comprends rien à ce code, c'est normal ?

Oui, il faut reconnaître qu'il contient quelques nouveautés. En effet, PDO est ce qu'on appelle une extension **orientée objet**. C'est une façon de programmer un peu différente des fonctions classiques que l'on a appris à utiliser jusqu'ici.



Nous aurons l'occasion d'en apprendre plus au sujet de la programmation orientée objet (POO) plus loin dans le cours. Pour l'instant, je vous invite à réutiliser les codes que je vous propose en suivant mes exemples. Vous comprendrez les détails de leur mode de fonctionnement un peu plus tard.

La ligne de code qu'on vient de voir crée ce qu'on appelle un *objet* `$bdd`. Ce n'est pas vraiment une variable (même si ça y ressemble fortement), c'est un objet qui représente la connexion à la base de données. On crée la connexion en indiquant dans l'ordre dans les paramètres :

- Le nom d'hôte (`localhost`)
- La base de données (`test`)
- Le login (`root`)
- Le mot de passe (ici il n'y a pas de mot de passe, j'ai donc mis une chaîne vide)

Lorsque votre site sera en ligne, vous aurez sûrement un nom d'hôte différent ainsi qu'un login et un mot de passe comme ceci :

Code : PHP

```
<?php  
$bdd = new PDO('mysql:host=sql.hebergeur.com;dbname=mabase',  
'pierre.durand', 's3cr3t');  
?>
```

Il faudra donc penser à changer cette ligne pour l'adapter à votre hébergeur en modifiant les informations en conséquence lorsque vous enverrez votre site sur le web.



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

Tester la présence d'erreurs

Si vous avez renseigné les bonnes informations (nom d'hôte, base, login, mot de passe), rien ne devrait s'afficher à l'écran. Toutefois, s'il y a une erreur (vous vous êtes trompé de mot de passe ou de nom de base de données par exemple), PHP risque d'afficher toute la ligne qui pose l'erreur, ce qui inclut le mot de passe !

Vous ne voudrez pas que vos visiteurs puissent voir le mot de passe si une erreur survient lorsque votre site est en ligne. Il est préférable de *traiter* l'erreur. En cas d'erreur, PDO renvoie ce qu'on appelle une **exception** qui permet de "capturer" l'erreur.

Voici comment je vous propose de faire :

Code : PHP

```
<?php
try
{
    $pdo_options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
    $pdo_options);
}
catch (Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
?>
```

Voilà encore un code un peu nouveau pour nous. Je ne vais pas rentrer dans le détail, car nous aurons l'occasion de parler de tout cela plus tard.

Néanmoins, je me doute que vous voudriez savoir comment ce code fonctionne au moins dans les grandes lignes. Voici le principe : PHP exécute les instructions à l'intérieur du bloc `try{ }`. Si une erreur se produit, il s'arrête et "saute" directement au niveau du `catch{ }` pour savoir quoi faire. Ici, on choisit d'afficher l'erreur et d'arrêter le script si on arrive dans le bloc `catch`.



Si au contraire tout se passe bien, PHP continue l'exécution du code et ne lit pas ce qu'il y a dans le bloc `catch`. Votre page PHP ne devrait donc rien afficher pour le moment.

Enfin, vous noterez qu'on crée un array `$pdo_options` que l'on envoie en paramètre sur la ligne suivante. L'objectif de cette manipulation est d'activer les exceptions PDO. Grâce à ce petit ajout (que je vous recommande d'effectuer systématiquement), nous pourrons récupérer les erreurs lorsqu'elles se produisent.



Oulah ! Tout ceci semble bien compliqué, je n'y comprends pas grand chose ! C'est grave docteur ?

Non pas du tout ! En fait, et j'insiste là-dessus, PDO nous fait utiliser des fonctionnalités de PHP que l'on n'a pas étudiées jusqu'à présent (programmation orientée objet, exceptions...). Contentez-vous pour le moment de réutiliser les codes que je vous propose et n'ayez crainte : nous reviendrons sur ces codes-là plus tard pour les expliquer dans le détail.

Si vous avez une page blanche, vous pouvez continuer. Si vous avez une erreur, lisez le message et essayez de comprendre ce qu'il signifie. Si vous êtes bloqué, n'hésitez pas à demander de l'aide sur [les forums](#) sinon vous ne pourrez pas aller plus loin.

Ce que vous devez retenir : utilisez ce modèle de blocs try/catch pour récupérer les erreurs relatives à la base de données, renvoyées par PDO. Cela vous permettra d'avoir un message clair (bon d'accord, en anglais) qui vous permettra de comprendre vos erreurs.

Récupérer les données

Nous allons dans un premier temps apprendre à lire des informations dans la base de données, puis nous verrons dans le chapitre suivant comment ajouter et modifier des données.

Pour travailler ici, il nous faut une base de données "toute prête" qui va nous servir de support pour travailler. Je vous invite à télécharger la table que j'ai créée pour vous :

Télécharger la table

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

Pour cet exemple, plusieurs amis ont voulu répertorier tous les jeux vidéo 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, qu'est-ce que je dois en faire de ce fichier `jeux_video.sql` ?

Inutile d'essayer de l'ouvrir, ça n'a pas d'intérêt. Il faut l'importer via l'onglet "Importer" de phpMyAdmin. Nous avons appris à le faire dans le chapitre précédent. Pensez à sélectionner votre base de données "test" au préalable.

Et voilà ! Vous devriez voir une nouvelle table apparaître à gauche : "jeux_video". Vous pouvez vous amuser à regarder ce qu'elle contient, pour vous faire une idée.



Voici les 5 premières entrées qu'elle contient (il y en a une cinquantaine en tout !) :

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

Pour le moment ne modifiez pas cette table. Notre objectif est de créer une page PHP qui va afficher ce que contient la table `jeux_video`.

Faire une requête

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

Pour récupérer des informations de la base de données, nous avons besoin de notre objet qui représente la connexion à la base. Vous vous souvenez, il s'agit de `$bdd`. Nous allons effectuer la requête comme ceci :

Code : PHP

```
<?php  
$reponse = $bdd->query('Tapez votre requête SQL ici');  
?>
```

On demande ainsi à effectuer une requête sur la base de données.



"query" en anglais signifie "requête".

On récupère ce que la base de données nous a renvoyé dans un autre objet, que l'on a appelé ici `$reponse`.

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 la première requête SQL que nous allons utiliser :

Code : SQL

```
SELECT * FROM jeux_video
```

Ceci peut se traduire par : « Prendre tout ce qu'il y a dans la table "jeux_video" ». 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_video`
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_video** : c'est le nom de la table dans laquelle il faut aller piocher.

Effectuons la requête avec la méthode que l'on vient de découvrir :

Code : PHP

```
<?php  
$reponse = $bdd->query('SELECT * FROM jeux_video');  
?>
```

\$reponse contient maintenant la réponse de MySQL. 😊

Afficher le résultat d'une requête

Le problème, c'est que \$reponse contient quelque chose d'inexploitable. MySQL nous renvoie beaucoup d'informations et il faut les organiser.

Vous imaginez toutes les informations qui sont dedans ? Si c'est une table à 10 champs, avec 200 entrées, cela représente plus de 2000 informations ! Pour ne pas tout traiter d'un coup, on extrait cette réponse ligne par ligne, c'est-à-dire entrée par entrée.

Pour récupérer une entrée, on prend la réponse de MySQL et on y exécute fetch(), ce qui nous renvoie la première ligne.

Code : PHP

```
<?php
$donnees = $reponse->fetch();
?>
```



"fetch" en anglais signifie "va chercher".

\$donnees est un array qui contient champ par champ les valeurs de la première entrée. 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 appellerez \$reponse->fetch(), vous passez à l'entrée suivante. La boucle est donc répétée autant de fois qu'il n'y a d'entrées dans votre table.

Ouf ! Cela fait beaucoup d'informations à la fois. Je vous propose de résumer tout ce qu'on vient d'apprendre, de la connexion via PDO à l'affichage du résultat de la requête :

Code : PHP

```
<?php
try
{
    // On se connecte à MySQL
    $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
    $pdo_options);

    // On récupère tout le contenu de la table jeux_video
    $reponse = $bdd->query('SELECT * FROM jeux_video');

    // On affiche chaque entrée une à une
    while ($donnees = $reponse->fetch())
    {
        ?>
        <p>
        <strong>Jeu</strong> : <?php echo $donnees['nom']; ?><br />
        Le possesseur de ce jeu est : <?php echo
        $donnees['possesseur']; ?>, et il le vend à <?php echo
        $donnees['prix']; ?> euros !<br />
        Ce jeu fonctionne sur <?php echo $donnees['console']; ?> et
        on peut y jouer à <?php echo $donnees['nbre_joueurs_max']; ?> au
```

```

maximum<br />
    <?php echo $donnees['possesseur']; ?> a laissé ses
commentaires sur <?php echo $donnees['nom']; ?> : <em><?php echo
$donnees['commentaires']; ?></em>
</p>
<?php
}

$reponse->closeCursor(); // Termine le traitement de la requête

}

catch(Exception $e)
{
    // En cas d'erreur précédemment, on affiche un message et on
arrête tout
    die('Erreur : '. $e->getMessage());
}

?>

```

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 !

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 une autre entrée.



Quelle est la différence entre \$reponse et \$donnees ?

\$reponse contenait la réponse de MySQL en vrac, sous forme d'objet.

\$donnees est un array renvoyé par le `fetch()`. A chaque fois qu'on fait une boucle, `fetch` va chercher dans \$reponse l'entrée suivante et organise les champs dans l'array \$donnees.



Je ne comprends pas la ligne `while ($donnees = $reponse->fetch()) ?`

En effet c'est un peu curieux et nouveau pour vous. Cette ligne fait 2 choses à la fois :

- Elle va récupérer une nouvelle entrée et place son contenu dans \$donnees
- Elle vérifie si \$donnees vaut vrai ou faux

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

Vous noterez à la fin la présence de la ligne

Code : PHP

```
<?php $reponse->closeCursor(); ?>
```

Elle provoque la "fermeture du curseur d'analyse des résultats". Cela signifie, en d'autres termes plus humains, que vous devez effectuer cet appel à `closeCursor()` à chaque fois que vous avez fini de traiter le retour d'une requête afin d'éviter d'avoir des problèmes à la requête suivante. Cela signifie qu'on a terminé le travail sur la requête.

Afficher seulement le contenu de quelques champs

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 utilisé la requête SQL suivante :

Code : SQL

```
SELECT nom FROM jeux_video
```

Reprenons le code complet précédent et adaptons-le pour afficher un nom de jeu par ligne :

Code : PHP

```
<?php
try
{
    $pdo_options [ PDO::ATTR_ERRMODE ] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
$pdo_options);

    $reponse = $bdd->query('SELECT nom FROM jeux_video');

    while ($donnees = $reponse->fetch())
    {
        echo $donnees['nom'] . '<br />';
    }

    $reponse->closeCursor();
}
catch (Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
?>
```

Essayer !

Ce code est très semblable au précédent mais c'est l'occasion pour vous de vous familiariser avec MySQL et PDO. Retenez bien en particulier les choses suivantes :

- La connexion à la base de données n'a besoin d'être faite qu'une seule fois, au début de la page.
- Il faut fermer les résultats de recherche avec `closeCursor()` après avoir traité chaque requête.

Les critères de sélection

Imaginons que je souhaite obtenir uniquement la liste des jeux disponibles de la console "Nintendo 64" et que je souhaite les trier par prix croissant. Ca paraît compliqué à faire ? Pas en SQL !

Vous allez voir qu'en modifiant nos requêtes SQL, il est possible de filtrer et trier très facilement vos données. Nous allons nous intéresser ici aux mots-clé suivants du langage SQL :

- 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_video WHERE possesseur='Patrick'
```

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



Vous noterez que les chaînes de caractères doivent être placées entre apostrophes pour les délimiter, comme c'est ici le cas pour 'Patrick'. Elles ne sont en revanche pas nécessaires pour les nombres.

Un petit code pour voir ce que ça donne ?

Code : PHP

```
<?php
try
{
    $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
    $pdo_options);

    $reponse = $bdd->query('SELECT nom, possesseur FROM jeux_video
    WHERE possesseur=\''Patrick\'');

    while ($donnees = $reponse->fetch())
    {
        echo $donnees['nom'] . ' appartient à ' .
        $donnees['possesseur'] . '<br />';
    }

    $reponse->closeCursor();
}
catch (Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
?>
```

[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 combiner plusieurs conditions. Par exemple, si je veux lister les jeux de Patrick qu'il vend à moins de 20 euros, je combinerai les critères de sélection à l'aide du mot-clé AND (qui signifie "et") :

Code : SQL

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

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



Il existe aussi le mot-clé OR pour signifier "ou".

ORDER BY

ORDER BY nous permet d'ordonner nos résultats. Nous pourrions ainsi classer les résultats en fonction de leur prix ! La requête SQL serait :

Code : SQL

```
SELECT * FROM jeux_video ORDER BY prix
```

Traduction : "Sélectionner tous les champs de jeux_video et ordonner les résultats par prix croissant.".

Application :

Code : PHP

```
<?php
try
{
    $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
$pdo_options);

    $reponse = $bdd->query('SELECT nom, prix FROM jeux_video ORDER
BY prix');

    while ($donnees = $reponse->fetch())
    {
        echo $donnees['nom'] . ' coûte ' . $donnees['prix'] . '
EUR<br />';
    }

    $reponse->closeCursor();
}
```

```

catch(Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
?>

```

[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_video ORDER BY prix DESC
```

Traduction : "Sélectionner tous les champs de jeux_video, 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

LIMIT nous permet de ne sélectionner qu'une partie des résultats (par exemple les 20 premiers). C'est très utile lorsqu'il y a beaucoup de résultats et que vous souhaitez les paginer (c'est-à-dire par exemple afficher les 30 premiers résultats sur la page 1, les 30 suivants sur la page 2, etc.).

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

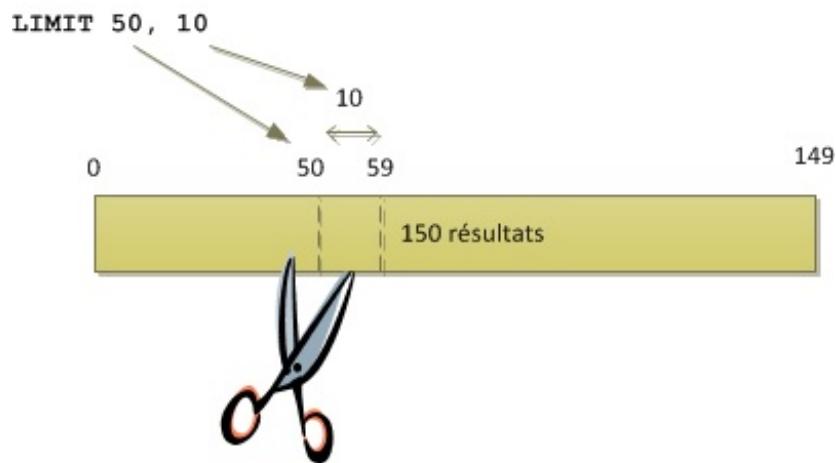
```
SELECT * FROM jeux_video LIMIT 0, 20
```

Ces deux nombres ont un sens bien précis :

- On indique tout d'abord à partir de **quelle entrée** on commence à lire la table. Ici, j'ai mis 0, ce qui correspond à la première entrée. Attention, cela n'a rien à voir avec le champ ID ! Imaginez qu'une requête retourne 100 résultats : LIMIT tronquera à partir du premier résultat si vous indiquez 0, à partir du 21ème si vous indiquez 20, etc.
- Ensuite, le deuxième nombre indique combien d'entrées on doit sélectionner. Ici, j'ai mis 20, on prendra donc 20 entrées.

Quelques exemples :

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



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

Code : PHP

```
<?php
try
{
    $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
$pdo_options);

    $reponse = $bdd->query('SELECT nom FROM jeux_video LIMIT 0,
10');

    echo '<p>Voici les 10 premières entrées de la table jeux_video
:</p>';
    while ($donnees = $reponse->fetch())
    {
        echo $donnees ['nom'] . '<br />';
    }

    $reponse->closeCursor();
}
catch (Exception $e)
{
    die('Erreur : '. $e->getMessage());
}
?>
```

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_video 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 dans un premier temps, 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.

Construire des requêtes en fonction de variables

Les requêtes que nous avons étudiées jusqu'ici étaient simples et effectuaient toujours la même opération. Or, les choses deviennent intéressantes quand on utilise des variables de PHP dans les requêtes.

La mauvaise idée : concaténer une variable dans une requête

Prenons cette requête qui récupère la liste des jeux appartenant à Patrick :

Code : PHP

```
<?php  
$reponse = $bdd->query('SELECT nom FROM jeux_video WHERE  
possesseur=\''Patrick'\');  
?>
```

Au lieu de toujours afficher les jeux de Patrick, on aimerait que cette requête soit capable de s'adapter au nom de la personne défini dans une variable, par exemple `$_GET['possesseur']`. Ainsi la requête pourrait s'adapter en fonction de la demande de l'utilisateur !

On pourrait être tenté de concaténer la variable dans la requête comme ceci :

Code : PHP

```
<?php  
$reponse = $bdd->query('SELECT nom FROM jeux_video WHERE  
possesseur=' . $_GET['possesseur'] . '\'');
```



Il est nécessaire d'entourer la chaîne de caractères d'apostrophes comme je vous l'ai indiqué précédemment, d'où la présence des antislash pour insérer les apostrophes : \ '

Bien que ce code fonctionne, c'est l'**illustration parfaite de ce qu'il ne faut pas faire** et que pourtant beaucoup de sites font encore. En effet, si la variable `$_GET['possesseur']` a été modifiée par un visiteur (et nous savons à quel point il ne faut pas faire confiance à l'utilisateur !), il y a un gros risque de faille de sécurité qu'on appelle **Injection SQL**. Un visiteur pourrait s'amuser à insérer une requête SQL au milieu de la vôtre et pourrait potentiellement lire tout le contenu de votre base de données, comme par exemple la liste des mots de passe de vos utilisateurs !



Le sujet des injections SQL est un peu complexe pour être détaillé ici. Si vous souhaitez en apprendre plus à ce sujet, je vous invite à consulter [Wikipédia](#).

Nous allons utiliser un autre moyen plus sûr d'adapter nos requêtes en fonction de variables : les requêtes préparées.

La solution : les requêtes préparées

Le système de requêtes préparées a l'avantage d'être beaucoup plus sûr mais aussi plus rapide pour la base de données si la requête est exécutée plusieurs fois. C'est ce que je vous préconise d'utiliser si vous voulez adapter une requête en fonction d'une ou plusieurs variables.

Avec des marqueurs "?"

On va dans un premier temps "préparer" la requête sans sa partie variable, que l'on représentera avec un marqueur sous forme de point d'interrogation :

Code : PHP

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

Au lieu d'exécuter la requête avec `query()` comme la dernière fois, on appelle ici `prepare()`.

La requête est alors prête, sans sa partie *variable*. Maintenant, nous allons exécuter la requête en appelant `execute` et en lui transmettant la liste des paramètres :

Code : PHP

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

La requête est alors exécutée à l'aide des paramètres que l'on a indiqués sous forme d'array.

S'il y a plusieurs marqueurs, il faut indiquer les paramètres dans le bon ordre :

Code : PHP

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

Le premier point d'interrogation de la requête sera remplacé par le contenu de la variable `$_GET['possesseur']` et le second point d'interrogation sera remplacé par le contenu de `$_GET['prix_max']`. Le contenu de ces variables aura été automatiquement sécurisé pour prévenir les risques d'injection SQL.

Essayons de construire une page capable de lister les jeux appartenant à une personne et dont le prix ne dépasse pas une certaine somme :

Code : PHP

```
<?php
try
{
    $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
    $pdo_options);
```

```

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

echo '<ul>';
while ($donnees = $req->fetch())
{
    echo '<li>' . $donnees['nom'] . ' (' . $donnees['prix'] . ' EUR)</li>';
}
echo '</ul>';

$req->closeCursor();
}
catch(Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}
?>

```



Bien que la requête soit "sécurisée" (ce qui élimine les risques d'injection SQL), il faudrait néanmoins vérifier quand même que `$_GET['prix_max']` contient bien un nombre et qu'il est compris dans un intervalle correct. Vous n'êtes donc pas dispensés d'effectuer des vérifications supplémentaires si vous estimez cela nécessaire.

Essayez d'appeler cette page (que l'on nommera par exemple `selection_jeux.php`) en modifiant les valeurs des paramètres, vous allez voir que la liste des jeux qui ressort change en fonction des paramètres envoyés ! Quelques exemples :

- `selection_jeux.php?possesseur=Michel&prix_max=20`
- `selection_jeux.php?possesseur=Michel&prix_max=40`
- `selection_jeux.php?possesseur=Patrick&prix_max=45`
- `selection_jeux.php?possesseur=Florent&prix_max=15`
- ... A vous de jouer !

Avec des marqueurs nominatifs

Si la requête contient beaucoup de parties variables, il peut être plus pratique de nommer les marqueurs plutôt que d'utiliser des points d'interrogation.

Voici comment on s'y prendrait :

Code : PHP

```

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

```

Les points d'interrogations ont été remplacés par les marqueurs nominatifs `:possesseur` et `:prixmax` (ils commencent par le symbole deux-points comme vous le voyez).

Ces marqueurs sont remplacés par les variables à l'aide cette fois d'un array associatif. Cela permet parfois plus de clarté quand

il y a beaucoup de paramètres. De plus, contrairement aux points d'interrogations, nous ne sommes cette fois plus obligés d'envoyer les variables dans le même ordre que la requête.

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 commencerez à être capables de créer des scripts de news, de livre d'or, un forum, 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 l'on pouvait facilement récupérer des informations de notre base de données. Nous avons aussi pu constater que le langage SQL était très puissant, car il propose de nombreux critères de sélection et de tri (WHERE, ORDER BY, etc.).

Il est maintenant temps de découvrir comment on peut ajouter et modifier des données dans la base. Pour cela, nous allons aborder de nouvelles requêtes SQL fondamentales à connaître : INSERT, UPDATE et DELETE.

INSERT : ajouter des données

Votre mission, si vous l'acceptez : ajouter une nouvelle entrée à la table "jeux_video" 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. Mais ce qui nous intéresse ici, c'est de le faire via un script PHP et une requête SQL !

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

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 !
...

La requête INSERT INTO permet d'ajouter une entrée

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



Les nombres (tels que 45 et 50 ici) n'ont pas besoin d'être entourés d'apostrophes. Seules les chaînes de caractères les nécessitent.

Etudions un peu cette requête :

- D'abord, vous devez commencer par les mots-clé **INSERT INTO** qui indiquent que vous voulez insérer une entrée.
- Vous précisez ensuite le nom de la table (ici **jeux_video**), puis listez entre parenthèses les noms des champs dans lesquels vous souhaitez placer des informations.
- Enfin, et c'est là qu'il ne faut pas se tromper, vous inscrivez **VALUES** suivi des valeurs à insérer **dans le même ordre que les champs que vous avez indiqués**.

Vous remarquerez que pour le premier champ (ID), j'ai laissé des apostrophes vides. C'est voulu : le champ a la propriété "auto_increment", MySQL mettra donc le numéro d'ID lui-même. On pourrait même se passer du champ ID dans la requête :

Code : SQL

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

C'est encore plus simple ! Le champ ID sera de toute façon automatiquement rempli par MySQL, il est donc inutile de le lister.

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

Code : SQL



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

Il faut lister les valeurs pour tous les champs sans exception (ID compris) dans le bon ordre.

Application en PHP

Utilisons cette requête SQL au sein d'un script PHP. Cette fois, au lieu de faire appel à `query()` (que l'on utilisait dans le chapitre précédent pour récupérer des données), on va utiliser `exec()` qui est prévue pour exécuter des modifications sur la base de données :

Code : PHP

```
<?php
try
{
    $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
$pdo_options);

    // On ajoute une entrée dans la table jeux_video
    $bdd->exec('INSERT INTO jeux_video(nom, possesseur, console,
prix, nbre_joueurs_max, commentaires) VALUES(\'Battlefield 1942\",
\'Patrick\', \'PC\', 45, 50, \'2nde guerre mondiale\')');

    echo 'Le jeu a bien été ajouté !';
}
catch(Exception $e)
{
    die('Erreur : '. $e->getMessage());
}
?>
```

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...

La présence de multiples apostrophes rend la requête un peu difficile à lire et à écrire à cause des antislash (\) que l'on doit rajouter devant. De plus, cette requête insère toujours les mêmes données. Comme on l'a vu dans le chapitre précédent, si on veut rendre une partie de la requête variable, le plus rapide et le plus sûr est de faire appel aux requêtes préparées.

Insertion de données variables grâce à une requête préparée

Si on choisit d'utiliser une requête préparée (ce que je vous recommande si vous souhaitez insérer des variables), le fonctionnement est en fait exactement le même que dans le chapitre précédent :

Code : PHP

```
<?php
$req = $bdd->prepare('INSERT INTO jeux_video(nom, possesseur,
console, prix, nbre_joueurs_max, commentaires) VALUES (:nom,
:possesseur, :console, :prix, :nbre_joueurs_max, :commentaires)');
$req->execute(array(
    'nom' => $nom,
    'possesseur' => $possesseur,
    'console' => $console,
    'prix' => $prix,
    'nbre_joueurs_max' => $nbre_joueurs_max,
    'commentaires' => $commentaires
));
echo 'Le jeu a bien été ajouté !';
?>
```



Je ne mets plus désormais l'étape de la connexion à MySQL avec PDO dans mes codes pour les simplifier. Bien entendu, il faut toujours se connecter au préalable si on veut que la requête fonctionne !

Pour plus de clarté, j'ai utilisé ici des marqueurs nominatifs. Comme vous le voyez, j'ai créé l'array sur plusieurs lignes : c'est autorisé et surtout c'est bien plus lisible.

Les variables telles que \$nom et \$possesseur doivent avoir été définies précédemment. Généralement, on récupèrera des variables de \$_POST (issues d'un formulaire) pour insérer une entrée dans la base de données. Nous découvrirons un cas pratique dans le TP suivant.

UPDATE : 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).

La requête UPDATE permet de modifier une entrée

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_video 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_video`).
- Le mot-clé SET, qui sépare le nom de la table de la liste des champs à modifier.
- Viennent ensuite les champs qu'il faut modifier, séparés par des virgules. Ici, on modifie le champ "prix", on lui affecte la valeur "10" (`prix = 10`), et de même pour le champ `nbre_joueurs_max`. Les autres champs ne seront pas modifiés.
- Enfin, le mot-clé WHERE est tout simplement indispensable. Il nous permet de dire à MySQL quelle entrée il doit modifier (sinon toutes les entrées seraient affectées !). On se base très souvent sur le champ ID pour indiquer *quelle entrée* doit être modifiée. Ici, on suppose que Battlefield a été enregistré sous l'ID n°51.

Si vous voulez, vous pouvez vous baser sur le nom du jeu au lieu de l'ID pour effectuer votre sélection :

Code : SQL

```
UPDATE jeux_video 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, il va falloir modifier chaque entrée une à une ? 🤔

Non ! Il n'est pas question de passer des heures à modifier chaque entrée une à une pour ça ! En réfléchissant environ 0,5 seconde vous allez trouver tous seuls la requête SQL qui permet de faire ce qu'on souhaite.

C'est bon vous avez trouvé ? Allez, je vous donne la réponse dans le mille :

Code : SQL

```
UPDATE jeux_video SET possesseur = 'Florent' WHERE possesseur = 'Michel'
```

Traduction : Dans la table jeux_video, 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.



Application en PHP

De la même manière, en PHP on fait appel à `exec()` pour effectuer des modifications :

Code : PHP

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

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

Code : PHP

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

Cela affichera quelque chose comme : 13 entrées ont été modifiées !

Avec une requête préparée

Si vous insérez des données variables, par exemple envoyées par l'utilisateur, je vous recommande là encore de faire appel à une requête préparée :

Code : PHP

```
<?php
$req = $bdd->prepare('UPDATE jeux_video SET prix = :nvprix,
nbre_joueurs_max = :nv_nb_joueurs WHERE nom = :nom_jeu');
$req->execute(array(
    'nvprix' => $nvprix,
    'nv_nb_joueurs' => $nv_nb_joueurs,
    'nom_jeu' => $nom_jeu
));
?>
```

DELETE : supprimer des données

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

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

Code : SQL

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

Il n'y a rien de plus facile :

- DELETE FROM : pour dire "supprimer dans".
- jeux_video : le nom de la table.
- WHERE : indispensable pour indiquer quelle(s) entrée(s) doivent être supprimée(s).



Si vous oubliez le WHERE, toutes les entrées seront supprimées ! Cela équivaut à vider la table.

Je vous laisse essayer cette requête en PHP. Vous pouvez là encore passer par `exec()` si vous voulez exécuter une requête bien précise, ou bien utiliser une requête préparée si votre requête dépend de variables.

Vous êtes maintenant capables de lire, ajouter, modifier et supprimer des informations dans une base de données. Ce sont toutes les opérations de base. On les résume parfois sous le nom CRUD (Create, Read, Update, Delete).

En théorie, une découverte minimale du langage SQL pourrait (déjà) s'arrêter là. Cependant, c'est un langage bien plus riche que ça : nous aurons l'occasion de découvrir d'autres fonctionnalités du SQL dans les prochains chapitres. Mais avant cela, un petit TP va nous permettre de nous exercer avec ce que nous avons appris. 😊

TP : un Mini-Chat

Voilà un TP qui va nous permettre de mettre en pratique tout ce que l'on vient d'apprendre avec le langage SQL. Il faut dire qu'on a enchaîné beaucoup de nouveautés dans les chapitres précédents (base de données, extraction des informations contenues dans une table, etc.).

Avec les connaissances que vous avez maintenant, vous êtes en mesure de réaliser de vrais scripts qui pourront vous être utiles pour votre site, comme un Mini-Chat (ce qu'on va faire) ou encore un livre d'or, un système de news (ou blog), etc. Ces scripts sont en fait assez similaires, mais le plus simple d'entre eux est le Mini-Chat.

Le Mini-Chat permettra d'ajouter facilement une touche de dynamisme à votre site, mais il faut encore le construire. A nous de jouer !

Instructions pour réaliser le TP

Pour préparer ce TP, nous allons voir les points suivants :

- Prérequis
- Objectifs
- Structure de la table MySQL
- Structure des pages PHP
- Rappels sur les consignes de sécurité

Prérequis

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

- Transmission de variables via un formulaire
- Lire dans une table
- Ecrire dans une table
- Utilisation de PDO et des requêtes préparées

Objectifs

Avant de commencer à attaquer notre script PHP, qu'est-ce que je vous avais dit qu'il fallait absolument faire en premier ? 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. En particulier, il faut se demander ce qu'on veut *exactement* faire.



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.

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 :

Pseudo :	<input type="text"/>
Message :	<input type="text"/>
<input type="button" value="Envoyer"/>	

Tom : Oui, il a appris ça sur www.siteduzero.com :-p

John : C'est pas mal du tout ! C'est toi qui l'a fait ?

M@teo21 : Qu'en penses-tu John ?

etc ...

Tom : Ouais ! C'est génial on va pouvoir discuter !

Message 3

M@teo21 : Mon script marche bien, n'est-ce pas ? :o)

Message 2

M@teo21 : Bonjour tout le monde !

Message 1

Maintenant que l'on sait ce que l'on veut obtenir, il nous sera beaucoup plus facile de le réaliser ! Et ne rigolez pas, trop de gens se lancent dans un script sans vraiment savoir ce qu'ils veulent faire, ce qui les conduit bien souvent dans un mur.

Structure de la table MySQL

Comme à chaque fois que l'on se servira d'une base de données, on va commencer par étudier la forme de la table, c'est-à-dire la liste des champs. Voici un petit tableau que j'ai réalisé en 1 minute 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 !

On distingue les champs suivants :

- ID (type INT) : 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 tout seuls, et ne pas oublier de sélectionner "Primaire" (ça dit à MySQL que c'est le champ qui numérote les entrées).
- pseudo (type VARCHAR) : pensez à indiquer la taille maximale du champ (je vous conseille de mettre le maximum, 255).
- message (type VARCHAR) : de même, on indiquera une taille maximale de 255 caractères. Si vous pensez que vos messages seront plus longs, utilisez plutôt le type TEXT qui est beaucoup moins limité.

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

Structure des pages PHP

Comme pour le TP "Page protégée par mot de passe", nous allons utiliser 2 fichiers PHP :

- `minichat.php` : contient le formulaire permettant d'ajouter un message et liste les 10 derniers messages.
- `minichat_post.php` : insère le message reçu avec `$_POST` dans la base de données puis redirige vers `minichat.php`.

Il aurait été possible de tout faire sur une seule page PHP, mais pour bien séparer le code il est préférable d'utiliser 2 fichiers comme ici.



Vous avez toutes les connaissances nécessaires pour réaliser un Mini-Chat basé sur la structure du schéma précédent... à l'exception de la redirection. En effet, il existe plusieurs moyens de rediriger le visiteur vers une autre page (via une balise `<meta>` par exemple) mais le plus propre et le plus rapide consiste à faire une **redirection HTTP**. Voici comment il faut procéder pour faire cela sur la page `minichat_post.php`:

Code : PHP

```
<?php
// Effectuer ici la requête qui insère le message
// Puis rediriger vers minichat.php comme ceci :
header('Location: minichat.php');
?>
```

Le visiteur ne verra jamais la page `minichat_post.php`. Celle-ci n'affiche rien mais commande en revanche au navigateur du visiteur de retourner sur `minichat.php`.

La fonction `header()` permet d'envoyer ce qu'on appelle des en-têtes HTTP. C'est le protocole utilisé sur le web entre le serveur et le client pour échanger des pages web. Ici, on utilise une des possibilités de HTTP qui commande une redirection via la commande `Location`.

Par rapport à d'autres types de redirection (comme la balise `<meta>`), cette technique a l'avantage d'être instantanée et transparente pour l'utilisateur. De plus, s'il rafraîchit ensuite la page `minichat.php`, il ne risque pas d'avoir le message souvent gênant et déroutant : "Pour afficher cette page, les informations précédemment transmises doivent être renvoyées. Etes-vous sûr de vouloir le faire ?".

Rappels sur les consignes de sécurité

Un petit rappel ne peut pas faire de mal : **ne faites jamais confiance aux données de l'utilisateur !** Tout ce qui vient de l'utilisateur doit être traité avec la plus grande méfiance.

Ici, on a une page `minichat_post.php` assez simple qui reçoit 2 champs : le pseudo et le message. A priori, il n'y a pas de vérification supplémentaire à faire, si ce n'est qu'il faudra veiller à l'affichage à protéger les chaînes de caractères contre la faille XSS (celle qui permet d'insérer du HTML et du Javascript dans la page). Il faudra donc bien veiller à appeler `htmlspecialchars()` pour protéger les chaînes.

A vous de jouer !

Allez, j'en ai assez dit. 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 ). Faites une pause et revenez-y plus tard.

Si vous coincez vraiment, vous pouvez demander de l'aide sur les forums ou regarder la correction pour vous aider. Faites l'effort dans tous les cas de travailler ce script, ce sera très formateur vous verrez !

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 !

Il y avait deux fichiers, commençons par minichat.php.

minichat.php : formulaire et liste des derniers messages

Cette page contient le formulaire d'ajout de message ainsi que la liste des derniers messages.

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>

    <form action="minichat_post.php" method="post">
        <p>
            <label for="pseudo">Pseudo</label> : <input type="text"
name="pseudo" id="pseudo" /><br />
            <label for="message">Message</label> : <input type="text"
name="message" id="message" /><br />

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

<?php
// Connexion à la base de données
try
{
    $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
$pdo_options);

    // Récupération des 10 derniers messages
    $reponse = $bdd->query('SELECT pseudo, message FROM minichat
ORDER BY ID DESC LIMIT 0, 10');

    // Affichage de chaque message (toutes les données sont
protégées par htmlspecialchars)
    while ($donnees = $reponse->fetch())
    {
        echo '<p><strong>' . htmlspecialchars($donnees['pseudo']) . 
'</strong> : ' . htmlspecialchars($donnees['message']) . '</p>';
    }
}

```

```

        $reponse->closeCursor();
    }
catch(Exception $e)
{
    die('Erreur : '. $e->getMessage());
}

?>
</body>
</html>

```

Le code de cette page est séparé en deux parties :

- Le formulaire (en HTML)
- La liste des messages (affichée en PHP à l'aide d'une requête SQL)

Il n'y avait pas de piège particulier, à l'exception du `htmlspecialchars()` à ne pas oublier sur le message ET le pseudo. Toutes les données issues du formulaire doivent être protégées pour éviter la faille XSS dont nous avons parlé dans un chapitre précédent.

minichat_post.php : enregistrement et redirection

La page `minichat_post.php` reçoit les données du formulaire, enregistre le message et redirige ensuite le visiteur sur la liste des derniers messages.

Code : PHP

```

<?php
// Connexion à la base de données
try
{
    $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
$pdo_options);

    // Insertion du message à l'aide d'une requête préparée
    $req = $bdd->prepare('INSERT INTO minichat (pseudo, message)
VALUES (?, ?)');
    $req->execute(array($_POST['pseudo'], $_POST['message']));

    // Redirection du visiteur vers la page du minichat
    header('Location: minichat.php');
}
catch(Exception $e)
{
    die('Erreur : '. $e->getMessage());
}
?>

```

Ce code est relativement court et sans surprises. On se connecte à la base, on insère les données et on redirige le visiteur vers la page `minichat.php` comme on vient d'apprendre à le faire.



En fait, ce code peut être amélioré (je vais en parler un peu plus loin). En effet, on ne teste pas si le pseudo et le message existent bien, s'ils sont vides ou non, etc. Il est donc en théorie possible d'enregistrer des messages vides, ce



qui idéalement ne devrait pas être autorisé.

Vous voulez tester le minichat ? Pas de problème ! Ce lien ouvre la page minichat .php :

[Essayer !](#)

Aller plus loin

Il serait dommage d'en rester là... Le script de Mini Chat que je vous ai fait faire est certes amusant, 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, ce 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 :

- **Retenir le pseudo.** On doit actuellement saisir à nouveau son pseudo à chaque nouveau message. 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 le pseudonyme du visiteur. Ce pseudonyme peut être issu d'un cookie par exemple : lorsqu'il poste un message, vous inscrivez son pseudo dans un cookie, ce qui vous permet ensuite de préremplir le champ.

- **Proposez d'actualiser le Mini Chat.** Le Mini Chat ne s'actualise pas automatiquement s'il y a de nouveaux messages. C'est normal, ce serait difficile à faire à notre niveau (le web n'a pas vraiment été prévu pour ce type d'applications à la base). En revanche, ce que vous pouvez facilement faire, c'est proposer un lien "Rafraîchir" qui charge à nouveau la page minichat.php. Ainsi, s'il y a de nouveaux messages, ils apparaîtront après un clic sur le lien.
- **Afficher les anciens messages.** On ne voit actuellement que les 10 derniers messages. Sauriez-vous trouver un moyen d'afficher les anciens messages ? Bien sûr, tous les afficher d'un coup sur la même page n'est pas une bonne idée. Vous pourriez imaginer un paramètre `$_GET['page']` qui permet de choisir le numéro de page des messages à afficher.

Au travail !

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 très 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.

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



Les fonctions SQL

Vous connaissez déjà les fonctions en PHP, mais vous allez découvrir dans ce chapitre que le SQL propose lui aussi toute une série de fonctions ! Le langage SQL permet en effet d'effectuer des calculs directement sur ses données à l'aide de fonctions toutes prêtes.

Celles-ci sont moins nombreuses qu'en PHP mais elles sont spécialement dédiées aux bases de données et se révèlent très puissantes dans la pratique. Pour reprendre notre exemple de la table `jeux_video`, elles permettent de récupérer très simplement le prix moyen de l'ensemble des jeux, de compter le nombre de jeux que possède chaque personne, d'extraire le jeu le plus cher ou le moins cher, etc. Les fonctions se révèlent aussi indispensables lorsqu'on doit travailler avec des dates en SQL, comme nous le ferons dans le chapitre suivant.

Les fonctions SQL peuvent être classées en 2 catégories :

- **Les fonctions scalaires** : elles agissent sur chaque entrée. Par exemple, vous pouvez transformer en majuscules la valeur de chacune des entrées d'un champ.
 - **Les fonctions d'agrégat** : lorsque vous utilisez ce type de fonction, des calculs sont faits sur l'ensemble de la table pour retourner **une** valeur. Par exemple, calculer la moyenne des prix retourne une valeur : le prix moyen.
-

Les fonctions scalaires

Nous allons d'abord découvrir le mode d'emploi d'une fonction SQL de type *scalaire* : la fonction **UPPER**. Lorsque vous aurez appris à vous en servir, vous serez capables de faire de même avec toutes les autres fonctions scalaires. Je vous proposerai alors une petite sélection de fonctions scalaires à connaître, sachant qu'il en existe d'autres mais que nous ne pouvons pas toutes les passer en revue, ce serait bien trop long. 😊

Utiliser une fonction scalaire SQL

Nous allons pour nos exemples nous baser sur la table `jeux_video` que nous connaissons bien maintenant. Pour rappel, voici à quoi elle ressemble :

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 !

On écrit les noms des fonctions SQL en majuscules, comme on le fait déjà pour la plupart des mots-clé comme **SELECT**, **INSERT**, etc. Ce n'est pas une obligation mais plutôt une convention, une habitude qu'ont prise les programmeurs.

Pour vous montrer comment on utilise les fonctions SQL scalaires, je vais me baser sur la fonction **UPPER()** qui permet de convertir l'intégralité d'un champ en majuscules. Supposons que nous souhaitions obtenir en majuscules les noms de tous les jeux. Voici comment on écrirait la requête SQL :

Code : SQL

```
SELECT UPPER(nom) FROM jeux_video
```

La fonction **UPPER** est utilisée sur le champ `nom`. On récupère ainsi tous les noms des jeux en majuscules.



Cela modifie-t-il le contenu de la table ?

Non ! La table reste la même. La fonction **UPPER** modifie seulement la valeur envoyée à PHP. On ne touche donc pas au contenu de la table.

Cela crée en fait un "champ virtuel" qui n'existe que le temps de la requête. Il est conseillé de donner un nom à ce champ virtuel qui représente les noms des jeux en majuscules. Il faut utiliser pour cela le mot-clé **AS** comme ceci :

Code : SQL

```
SELECT UPPER(nom) AS nom_maj FROM jeux_video
```

On récupère les noms des jeux en majuscules via un champ virtuel appelé nom_maj.



Ce champ virtuel est appelé **alias**.

Voici le tableau que retournera MySQL d'après la requête précédente :

nom_maj
SUPER MARIO BROS
SONIC
ZELDA : OCARINA OF TIME
MARIO KART 64
SUPER SMASH BROS MELEE

On peut s'en servir en PHP pour afficher les noms des jeux en majuscules :

Code : PHP

```
<?php
$reponse = $bdd->query('SELECT UPPER(nom) AS nom_maj FROM
jeux_video');

while ($donnees = $reponse->fetch())
{
    echo $donnees['nom_maj'] . '<br />';
}

$reponse->closeCursor();

?>
```

[Essayer !](#)

Comme vous le voyez, PHP ne récupère qu'un champ nommé nom_maj (même s'il n'existe pas dans la table). En affichant le contenu de ce champ, on ne récupère que les noms des jeux en majuscules.

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

Code : SQL

```
SELECT UPPER(nom) AS nom_maj, possesseur, console, prix FROM
jeux_video
```

On récupèrera alors les données suivantes :

nom_maj	possesseur	console	prix
SUPER MARIO BROS	Florent	NES	4

SONIC	Patrick	Megadrive	2
ZELDA : OCARINA OF TIME	Florent	Nintendo 64	15
MARIO KART 64	Florent	Nintendo 64	25
SUPER SMASH BROS MELEE	Michel	GameCube	55

Vous savez maintenant utiliser une fonction SQL scalaire. 😊

Passons en revue quelques fonctions du même type, qui s'utilisent donc de la même manière.

Présentation de quelques fonctions scalaires utiles

Je vais vous présenter une sélection de fonctions scalaires qu'il peut être utile de connaître. Il en existe bien d'autres comme nous le verrons à la fin de cette liste, mais il serait trop long et peu utile de toutes les présenter ici.

UPPER : convertir en majuscules

Cette fonction convertit le texte d'un champ en majuscules. Nous l'avons découverte pour introduire les fonctions SQL :

Code : SQL

```
SELECT UPPER(nom) AS nom_ma_j FROM jeux_video
```

Ainsi, le jeu "Sonic" sera renvoyé sous la forme "SONIC" dans un champ nommé nom_ma_j.

LOWER : convertir en minuscules

Cette fonction a l'effet inverse : le contenu sera entièrement écrit en minuscules.

Code : SQL

```
SELECT LOWER(nom) AS nom_min FROM jeux_video
```

Cette fois, le jeu "Sonic" sera renvoyé sous la forme "sonic" dans un champ nommé nom_min.

LENGTH : compter le nombre de caractères

Vous pouvez obtenir la longueur d'un champ avec la fonction LENGTH () :

Code : SQL

```
SELECT LENGTH(nom) AS longueur_nom FROM jeux_video
```

Pour "Sonic", on récupèrera donc la valeur 5 dans un champ longueur_nom.

ROUND : arrondir un nombre décimal

La fonction ROUND () s'utilise sur des champs comportant des valeurs décimales. Il n'y en a pas dans la table `jeux_video`, mais si on avait des prix décimaux, on pourrait arrondir les valeurs avec cette fonction.

Celle-ci prend cette fois 2 paramètres : le nom du champ à arrondir et le nombre de chiffres après la virgule que l'on souhaite obtenir. Exemple :

Code : SQL

```
SELECT ROUND(prix, 2) AS prix_arrêté FROM jeux_video
```

Ainsi, si un jeu coûte 25,86999 €, il sera retourné sous forme de la valeur 25,87 € dans un champ `prix_arrêté`.

Et bien d'autres !

Il existe beaucoup d'autres fonctions SQL du même type mais je ne peux pas toutes vous les présenter. La documentation de MySQL vous propose une liste bien plus complète de [fonctions mathématiques](#) (comme ROUND) et de fonctions sur les [chaînes de caractères](#) (comme UPPER). Si vous voulez en découvrir d'autres, c'est par là qu'il faut aller !

Les fonctions d'agrégat

Comme précédemment, nous allons d'abord voir comment on utilise une fonction d'agrégat dans une requête SQL et comment on récupère le résultat en PHP, puis je vous présenterai une sélection de fonctions à connaître. Bien entendu, il en existe bien d'autres que vous pourrez découvrir dans la documentation. L'essentiel est de comprendre comment s'utilise ce type de fonction : vous pourrez ensuite appliquer ce que vous connaissez à n'importe quelle autre fonction du même type. 😊

Utiliser une fonction d'agrégat SQL

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

Par exemple, ROUND permettait d'arrondir chaque prix. On récupérait autant d'entrées qu'il y en avait dans la table. En revanche, une fonction d'agrégat comme AVG renvoie **une seule entrée** : la valeur moyenne de tous les prix.

Regardons de près la fonction d'agrégat AVG. Elle calcule la moyenne d'un champ contenant des nombres. Utilisons-la sur le champ `prix` :

Code : SQL

```
SELECT AVG(prix) AS prix_moyen FROM jeux_video
```

On donne là encore un alias au résultat donné par la fonction. La particularité, c'est que cette requête ne va retourner qu'une seule entrée, le prix moyen de tous les jeux :

prix_moyen
28.34

Pour afficher cette information en PHP, on pourrait faire comme on en a l'habitude (cela fonctionne) :

Code : PHP

```
<?php
$reponse = $bdd->query('SELECT AVG(prix) AS prix_moyen FROM
jeux_video');

while ($donnees = $reponse->fetch())
{
    echo $donnees['prix_moyen'];
}

$reponse->closeCursor();

?>
```

Néanmoins, pourquoi s'embêterait-on à faire une boucle étant donné qu'on *sait* qu'on ne va récupérer qu'une seule entrée, puisqu'on utilise une fonction d'agrégat ?

On peut se permettre d'appeler `fetch()` une seule fois, en dehors d'une boucle, étant donné qu'il n'y a qu'une seule entrée. Le code suivant est donc un peu plus adapté dans le cas présent :

Code : PHP

```
<?php  
$reponse = $bdd->query('SELECT AVG(prix) AS prix_moyen FROM  
jeux_video');  
  
$donnees = $reponse->fetch();  
echo $donnees['prix_moyen'];  
  
$reponse->closeCursor();  
  
?>
```

Essayer !

Ce code est plus simple et plus logique. On récupère la première (et seule) entrée une fois avec `fetch()` et on affiche ce qu'elle contient, puis on ferme le curseur. Inutile de le faire dans une boucle étant donné qu'il n'y a pas de seconde entrée.

N'hésitez pas à filtrer !

Bien entendu, vous pouvez profiter de toute la puissance du langage SQL pour obtenir, par exemple, le prix moyen des jeux appartenant à Patrick. Voici comment on s'y prendrait :

Code : SQL

```
SELECT AVG(prix) AS prix_moyen FROM jeux_video WHERE  
possesseur='Patrick'
```

Le calcul de la moyenne ne sera fait que sur la liste des jeux qui appartiennent à Patrick. Vous pourriez même combiner les conditions pour obtenir le prix moyen des jeux de Patrick qui se jouent à 1 seul joueur. Essayez !

Ne pas mélanger une fonction d'agrégat avec d'autres champs

Soyez attentifs à ce point car il n'est pas forcément évident à comprendre : vous ne devez *pas* récupérer d'autres champs de la table quand vous utilisez une fonction d'agrégat, contrairement à tout à l'heure avec les fonctions scalaires. En effet, quel sens cela aurait-il de faire :

Code : SQL

```
SELECT AVG(prix) AS prix_moyen, nom FROM jeux_video
```

On récupèrerait d'un côté le prix moyen de tous les jeux et de l'autre la liste des noms de tous les jeux ? Ca ne peut pas se représenter dans un seul et même tableau.

Comme vous le savez, SQL renvoie les informations sous la forme d'un tableau. Or on ne peut pas représenter la moyenne des prix (qui tient en une seule entrée) en même temps que la liste des jeux. Il faudrait faire 2 requêtes si on voulait avoir ces 2 informations.

Présentation de quelques fonctions d'agrégat utiles

AVG : calculer la moyenne

C'est la fonction que l'on vient d'étudier pour découvrir les fonctions d'agrégat. Elle retourne la moyenne d'un champ contenant des nombres :

Code : SQL

```
SELECT AVG(prix) AS prix_moyen FROM jeux_video
```

SUM : additionner les valeurs

La fonction SUM permet d'additionner toutes les valeurs d'un champ. Ainsi, on pourrait connaître la valeur totale des jeux appartenant à Patrick :

Code : SQL

```
SELECT SUM(prix) AS prix_total FROM jeux_video WHERE possesseur='Patrick'
```

MAX : retourner la valeur maximale

Cette fonction analyse un champ et retourne la valeur maximale trouvée. Pour obtenir le prix du jeu le plus cher :

Code : SQL

```
SELECT MAX(prix) AS prix_max FROM jeux_video
```

MIN : retourner la valeur minimale

De même on peut obtenir le prix du jeu le moins cher :

Code : SQL

```
SELECT MIN(prix) AS prix_min FROM jeux_video
```

COUNT : compter le nombre d'entrées

La fonction COUNT permet de compter le nombre d'entrées. Elle est très intéressante mais plus complexe. On peut en effet l'utiliser de plusieurs façons différentes.

L'utilisation la plus courante consiste à lui donner * en paramètre :

Code : SQL

```
SELECT COUNT(*) AS nb_jeux FROM jeux_video
```

On obtient ainsi le nombre total de jeux dans la table.

On peut bien entendu filtrer avec une clause WHERE, pour obtenir le nombre de jeux de Florent par exemple :

Code : SQL

```
SELECT COUNT(*) AS nbjeux FROM jeux_video WHERE possesseur='Florent'
```

Il est possible de compter uniquement les entrées pour lesquelles l'un des champs n'est pas vide, c'est-à-dire qu'il ne vaut pas NULL. Il n'y a pas de jeu de ce type dans notre table `jeux_video`, mais supposons que pour certains jeux on ne connaisse pas le nombre de joueurs maximum. On laisserait certaines entrées vides, ce qui aurait pour effet d'afficher NULL (pas de valeur) sur la colonne `nbre_joueurs_max` comme ceci :

ID	nom	possesseur	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	Florent	NES	4	NULL	Un jeu d'anthologie !
2	Sonic	Patrick	Megadrive	2	NULL	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	NULL	Un jeu de baston délirant !

Dans ce cas, on peut compter uniquement les jeux qui ont un nombre de joueurs maximum défini. On doit indiquer en paramètre le nom du champ à analyser :

Code : SQL

```
SELECT COUNT(nbre_joueurs_max) AS nbjeux FROM jeux_video
```

Dans notre exemple, seuls les jeux Zelda et Mario Kart seront comptés car on connaît leur nombre de joueurs maximum. Donc on obtiendra 2 en réponse.

Enfin, il est possible de compter le nombre de valeurs distinctes sur un champ précis. Par exemple dans la colonne `possesseur`, Florent apparaît plusieurs fois, Patrick aussi, etc. Mais combien y a-t-il de personnes différentes dans la table ? On peut le savoir en utilisant le mot-clé DISTINCT devant le nom du champ à analyser, comme ceci :

Code : SQL

```
SELECT COUNT(DISTINCT possesseur) AS nbpossesseurs FROM jeux_video
```

On peut ainsi facilement savoir combien de personnes différentes sont référencées dans la table. Essayez de faire de même pour connaître le nombre de consoles différentes dans la table !

GROUP BY et HAVING : le groupement de données

Je vous disais un peu plus tôt qu'on ne pouvait pas récupérer d'autres champs lorsqu'on utilisait une fonction d'agrégat. Prenons par exemple la requête suivante :

Code : SQL

```
SELECT AVG(prix) AS prix_moyen, console FROM jeux_video
```

Ca n'a pas de sens de récupérer le prix moyen de tous les jeux et le champ "console" à la fois. Il n'y a qu'un seul prix moyen pour tous les jeux, mais plusieurs consoles. MySQL ne peut pas renvoyer un tableau correspondant à ces informations-là.

Si vous essayez d'exécuter la requête, vous obtiendrez des résultats incohérents :

prix_moyen	console
28.34	NES

On a bien le bon prix moyen de tous les jeux, mais on récupère juste une console (pourquoi celle-là plutôt qu'une autre ?). Bref, ça ne veut rien dire.

GROUP BY : grouper des données

Par contre, ce qui pourrait avoir du sens, ce serait de demander le **prix moyen des jeux pour chaque console** ! Pour faire cela, on doit utiliser un nouveau mot-clé : GROUP BY. Cela signifie "grouper par". On utilise cette clause en combinaison d'une fonction d'agrégat (comme AVG) pour obtenir des informations intéressantes sur des groupes de données.

Voici un exemple d'utilisation de GROUP BY :

Code : SQL

```
SELECT AVG(prix) AS prix_moyen, console FROM jeux_video GROUP BY console
```

[Essayer !](#)

Il faut utiliser GROUP BY en même temps qu'une fonction d'agrégat, sinon il ne sert à rien. Ici, on récupère le prix moyen et la console, et on choisit de grouper par console. Par conséquent, on obtiendra la liste des différentes consoles de la table et le prix moyen des jeux qu'elles contiennent !

prix_moyen	console
12.67	Dreamcast
5.00	Gameboy
47.50	GameCube

Cette fois les valeurs sont cohérentes ! On a la liste des consoles et le prix moyen des jeux associés.

Exercice : essayez d'obtenir de la même façon la valeur totale des jeux que possède chaque personne.

HAVING : filtrer les données regroupées

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

Voyez la requête suivante :

Code : SQL

```
SELECT AVG(prix) AS prix_moyen, console FROM jeux_video GROUP BY
console HAVING prix_moyen <= 10
```

Essayer !

Avec cette requête, on récupère uniquement la liste des consoles et leur prix moyen si ce prix moyen ne dépasse pas 10 euros.

HAVING ne doit s'utiliser que sur le résultat d'une fonction d'agrégat. Voilà pourquoi on l'utilise ici sur `prix_moyen` et non sur `console`.



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

Oui, mais pas au même moment. WHERE agit en premier, avant le regroupement des données, tandis que HAVING agit en second, après le regroupement des données. On peut d'ailleurs très bien combiner les deux, regardez l'exemple suivant :

Code : SQL

```
SELECT AVG(prix) AS prix_moyen, console FROM jeux_video WHERE
possesseur='Patrick' GROUP BY console HAVING prix_moyen <= 10
```

Ca commence à faire de la requête maousse costaud.



Essayer !

Ici, on demande à récupérer le prix moyen par console de tous les jeux de Patrick (WHERE), à condition que le prix moyen des jeux de la console ne dépasse pas 10 euros (HAVING).

Les fonctions SQL ouvrent de nombreuses possibilités comme vous avez pu le constater.

On pourrait se dire que certaines fonctionnalités auraient aussi bien pu être traitées en PHP. Par exemple, mettre en majuscules le nom des jeux aurait pu être fait avec une fonction PHP adaptée au moment de l'affichage. C'est vrai, mais les fonctions SQL sont souvent plus souples et plus indiquées car on récupère ainsi un résultat déjà mis en forme. D'autre part, si on se met à utiliser des fonctions d'agrégat en même temps qu'un regroupement de données, on peut rapidement obtenir des informations très intéressantes à partir d'une table toute simple.

L'utilisation de GROUP BY et HAVING n'est d'ailleurs pas toujours très bien comprise, j'ai parfois vu des gens les utiliser avec des fonctions scalaires ce qui n'a pas de sens. Prenez donc bien le temps de comprendre ce qu'elles font pour ne pas les utiliser n'importe comment. Demandez-vous toujours "Qu'est-ce que je veux obtenir" et construisez la requête bout par bout car tout

faire d'un coup peut parfois se révéler vraiment difficile.

Les dates en SQL

Lorsque nous avons construit nos tables, nous avons utilisé différents types de champs, notamment INT (nombre entier), VARCHAR (texte court) et TEXT (texte long). Vous avez pu découvrir dans phpMyAdmin qu'il existait de nombreux autres types. La plupart ne sont que des variations de ces types, pour stocker par exemple des très petits nombres ou des très grands nombres. La plupart du temps, vous n'aurez pas à vous soucier de tous ces types : INT suffit par exemple amplement pour les nombres entiers.

Les dates sont plus délicates à manier en SQL, et pourtant on en a très souvent besoin. Par exemple, dans le TP du Mini Chat, on pourrait s'en servir pour stocker le jour et l'heure précise où chaque message a été posté. Il en va de même si vous construisez un système de forum ou de news pour votre site : vous aurez besoin d'enregistrer la date à chaque fois.

Nous ferons d'abord le tour des types de champs à connaître pour stocker des dates avec MySQL et nous verrons comment les utiliser. Nous pourrons ensuite découvrir de nouvelles fonctions SQL dédiées aux dates. 

Les champs de type date

Dans ce chapitre, je vous propose d'améliorer un peu le Mini Chat que nous avons créé dans un TP précédent.

Saviez-vous qu'il était possible de modifier la structure d'une table après sa création ? Il est en effet possible d'y ajouter ou d'y supprimer des champs à tout moment. Ouvrez la table `mini_chat` dans phpMyAdmin, onglet "Structure". Cherchez en bas de la page le formulaire "Ajouter 1 champ en fin de table" et cliquez sur le bouton "Exécuter".

The screenshot shows a top navigation bar with links for 'Version imprimable', 'Suggérer des optimisations quant à la structure de la table', and a search bar. Below this is a form titled 'Ajouter 1 champ(s)'. The 'En fin de table' radio button is selected. The 'Type' dropdown is set to 'DATETIME'. A dropdown for 'Après' has 'ID' selected. At the bottom right is a large 'Exécuter' (Execute) button.

Un formulaire apparaît, vous proposant de créer un nouveau champ. C'est l'occasion de passer en revue les différents types de champs qui permettent de stocker des dates.

Les différents types de dates

Voici les différents types de dates que peut stocker MySQL :

- DATE : stocke une date au format AAAA-MM-JJ (Année-Mois-Jour).
- TIME : stocke un moment de la journée au format HH:MM:SS (Heures:Minutes:Secondes).
- DATETIME : stocke une combinaison d'une date et d'un moment de la journée au format AAAA-MM-JJ HH:MM:SS. Ce type de champ est donc plus précis.
- TIMESTAMP : stocke une date et un moment de la journée sous le format AAAAMMJJHHMMSS.
- YEAR : stocke une année, soit au format AA, soit au format AAAA.

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

Créez un champ nommé `date` de type DATETIME comme ceci :

Champ	<input type="text" value="date"/>
Type	<input type="text" value="DATETIME"/>
Taille/Valeurs^{*1}	<input type="text"/>
Défaut²	<input type="text" value="Aucun"/>
Interclassement	<input type="text"/>
Attributs	<input type="text"/>
Null	<input type="checkbox"/>
Index	<input type="text" value="--"/>
AUTO_INCREMENT	<input type="checkbox"/>
Commentaires	<input type="text"/>

Bien que cela fonctionne avec MySQL, il est parfois préférable de donner un autre nom au champ que `date`. En effet, c'est un mot-clé du langage SQL, ce qui peut provoquer des erreurs avec d'autres systèmes de bases de données comme Oracle.



Vous pourriez par exemple nommer le champ comme ceci : `date_creation`, ou encore `date_modification`.

Utilisation des champs de date en SQL

Les champs de type date s'utilisent comme des chaînes de caractères : il faut donc les entourer d'apostrophes. Vous devez écrire la date dans le format du champ.

Par exemple, pour un champ de type DATE :

Code : SQL

```
SELECT pseudo, message, date FROM minichat WHERE date = '2010-04-02'
```

... vous renverra la liste des messages postés le 02/04/2010 (2 avril 2010).

Si le champ est de type DATETIME (comme c'est le cas pour notre nouveau Mini Chat), il faut aussi indiquer précisément les heures, minutes et secondes :

Code : SQL

```
SELECT pseudo, message, date FROM minichat WHERE date = '2010-04-02  
15:28:22'
```

... vous renverra la liste des messages postés le 02/04/2010 à 15h28min22s.

Bon je reconnais que c'est un peu précis, il est peu probable que beaucoup de messages aient été postés à ce moment exact. 🍪

En revanche, et c'est là que les champs de date deviennent réellement intéressants, vous pouvez utiliser d'autres opérateurs que le signe égal. Par exemple, on peut obtenir la liste de tous les messages postés *après* cette date :

Code : SQL

```
SELECT pseudo, message, date FROM minichat WHERE date >= '2010-04-02  
15:28:22'
```

Ou même la liste de tous les messages postés entre le 02/04/2010 et le 18/04/2010 :

Code : SQL

```
SELECT pseudo, message, date FROM minichat WHERE date >= '2010-04-02  
00:00:00' AND date <= '2010-04-18 00:00:00'
```

En SQL, pour récupérer des données comprises entre 2 intervalles comme ici, il y a une syntaxe plus simple et plus élégante avec le mot-clé `BETWEEN` qui signifie "Entre". On pourrait écrire la requête précédente comme ceci :

Code : SQL



```
SELECT pseudo, message, date FROM minichat WHERE date BETWEEN  
'2010-04-02 00:00:00' AND '2010-04-18 00:00:00'
```

Cela signifie : récupérer tous les messages dont la date est comprise entre 2010-04-02 00:00:00 et 2010-04-18 00:00:00. Vous pouvez aussi utiliser cette syntaxe sur les champs contenant des nombres.

Si vous voulez insérer une entrée contenant une date, il suffit là encore de respecter le format de date de la base de données :

Code : SQL

```
INSERT INTO minichat(pseudo, message, date) VALUES('Mateo', 'Message  
!', '2010-04-02 16:32:22')
```

Les fonctions de gestion des dates

Il existe de très nombreuses fonctions de manipulation des dates. Utilisées sur des champs de type DATE ou DATETIME par exemple, elles permettent d'extraire très facilement toutes sortes d'informations utiles sur les dates, comme l'année, le numéro du jour du mois, le numéro du jour dans l'année, etc. Il est aussi possible d'effectuer des opérations sur les dates.

Il est impossible de lister toutes les fonctions de gestion des dates, mais vous trouverez la liste complète dans la [documentation de MySQL sur les dates](#) au besoin. Cette introduction aux dates devrait être suffisante pour que vous puissiez vous débrouiller tous seuls par la suite. 😊

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

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

Code : SQL

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

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

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

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

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

Code : SQL

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

On récupèrera 3 champs : le pseudo, le message et le numéro du jour où il a été posté.

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

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

Code : SQL

```
SELECT pseudo, message, HOUR(date) AS heure FROM minichat
```

DATE_FORMAT : formater une date

Avec les fonctions que l'on vient de découvrir à l'instant, on pourrait extraire tous les éléments de la date comme ceci :

Code : SQL

```
SELECT pseudo, message, DAY(date) AS jour, MONTH(date) AS mois,
YEAR(date) AS annee, HOUR(date) AS heure, MINUTE(date) AS minute,
SECOND(date) AS seconde FROM minichat
```

On pourrait ensuite afficher la date dans l'ordre que l'on souhaite en PHP à l'aide du découpage en champs qu'on vient de faire :

Code : PHP

```
<?php
echo $donnees['jour'] . '/' . $donnees['mois'] . '/' .
$donnees['annee'] . '...';
?>
```

C'est cependant un peu compliqué et il y a plus simple. La fonction DATE_FORMAT vous permet d'adapter directement la date au format que vous préférez. Il faut dire que le format par défaut de MySQL (AAAA-MM-JJ HH:MM:SS) n'est pas très courant en France. 

Voici comment on pourrait l'utiliser :

Code : SQL

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

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



Comment ça marche ce bazar ? 

Les symboles %d, %m, %Y (etc.) sont remplacés par le jour, le mois, l'année, etc. Les autres symboles et lettres sont affichés tels quels.

Il existe beaucoup d'autres symboles, pour extraire par exemple le nom du jour (la plupart du temps en anglais malheureusement, comme "Saturday"), le numéro du jour dans l'année, etc. La liste des symboles disponibles est dans la doc à la section DATE_FORMAT.

DATE_ADD et DATE_SUB : ajouter ou soustraire des dates

Il est possible d'ajouter ou soustraire des heures, minutes, secondes, jours, mois ou années à une date. Il faut envoyer 2 paramètres à la fonction : la date sur laquelle travailler et le nombre à ajouter ainsi que son type.

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

Code : SQL

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

Le champ date_expiration correspond à la date de l'entrée + 15 jours. Le mot-clé INTERVAL ne doit pas être changé, en revanche vous pouvez remplacer DAY par MONTH, YEAR, HOUR, MINUTE, SECOND, etc. Par conséquent, si vous souhaitez que les messages expirent dans 2 mois :

Code : SQL

```
SELECT pseudo, message, DATE_ADD(date, INTERVAL 2 MONTH) AS  
date_expiration FROM minichat
```

 Juste un petit rappel pour être sûr qu'on se comprend bien : cette requête ne commande pas la suppression des messages dans 2 mois ! On crée juste un champ virtuel date_expiration qui correspond à la date du message + 2 mois. Cela serait utile pour informer l'utilisateur quand son message va expirer, mais ce serait à vous de faire en sorte que votre site supprime le message au bout de 2 mois !

Ce tour d'horizon des dates vous sera rapidement très utile. On peut effectuer de nombreuses opérations sur les dates à l'aide de fonctions comme vous venez de le découvrir, ce qui nous fait gagner du temps et nous permet d'obtenir en PHP directement les informations que l'on souhaite dans la forme qui nous convient le mieux.

TP : un blog avec des commentaires

Le blog est probablement l'application la plus courante que l'on réalise en PHP avec MySQL. Bien qu'il soit conseillé d'utiliser un système tout prêt (en téléchargeant Wordpress ou Dotclear par exemple), en créer un de toutes pièces est un excellent exercice.

Le but de ce TP n'est pas de vous faire créer un blog de A à Z, car ce serait un peu long, mais plutôt d'appliquer les dernières notions de SQL que vous venez d'apprendre sur les fonctions et les dates en SQL.

Chaque billet du blog possèdera ses propres commentaires. Nous nous concentrerons dans ce TP uniquement sur l'affichage des billets et des commentaires, ce sera à vous ensuite de compléter le blog pour y insérer des formulaires d'ajout et de modification du contenu. 

Instructions pour réaliser le TP

Pour ce TP comme pour le précédent, nous allons nous préparer ensemble en passant en revue les points suivants :

- Prérequis
- Objectifs
- Structure de la table MySQL
- Structure des pages PHP

Prérequis

Dans ce TP, nous allons nous concentrer sur la base de données. Nous aurons besoin des notions suivantes :

- Lire dans une table
- Utilisation de PDO et des requêtes préparées
- Utilisation de fonctions SQL
- Manipulation des dates en SQL

Objectifs

Commençons par définir ce qu'on veut arriver à faire. Un système de blog avec des commentaires, oui, mais encore ? Il faut savoir jusqu'où on veut aller, ce qu'on a l'intention de réaliser et ce qu'on va mettre de côté.

Si on est trop ambitieux, on risque de le regretter : on pourrait en effet y passer des jours et ce TP deviendrait long, complexe et fastidieux. Je vous propose donc de réaliser l'affichage de base d'un blog et des commentaires associés aux billets, et je vous inviterai par la suite à l'améliorer pour créer l'interface de gestion des billets et d'ajout de commentaires.

L'ajout de billets et de commentaires ne sont donc pas au programme de ce TP, ce qui va nous permettre de nous concentrer sur leur affichage.

Les pages à développer

Il y aura 2 pages à réaliser :

- index.php : liste des 5 derniers billets
- commentaires.php : affichage d'un billet et de ses commentaires

Voici à quoi devrait ressembler la liste des 5 derniers billets (index.php) :

Mon super blog !

Derniers billets du blog :

Le PHP à la conquête du monde ! le 27/03/2010 à 18h31min11s

C'est officiel, l'élePHPant a annoncé à la radio hier soir "J'ai l'intention de conquérir le monde !".

Il a en outre précisé que le monde serait à sa botte en moins de temps qu'il n'en fallait pour dire "éléPHPant". Pas dur, ceci dit entre nous...

[Commentaires](#)

Bienvenue sur mon blog ! le 25/03/2010 à 16h28min41s

Je vous souhaite à toutes et à tous la bienvenue sur mon blog qui parlera de... PHP bien sûr !

[Commentaires](#)

Voici à quoi devrait ressembler l'affichage d'un billet et de ses commentaires (`commentaires.php`) :

Mon super blog !

[Retour à la liste des billets](#)

Bienvenue sur mon blog ! le 25/03/2010 à 16h28min41s

Je vous souhaite à toutes et à tous la bienvenue sur mon blog qui parlera de... PHP bien sûr !

Commentaires

M@teo21 le 25/03/2010 à 16h49min53s

Un peu court ce billet !

Maxime le 25/03/2010 à 16h57min16s

Oui, ça commence pas très fort ce blog...

MultiKiller le 25/03/2010 à 17h12min52s

+1 !

Comme vous pouvez le constater, l'affichage est minimaliste. Le but n'est pas de réaliser le design de ce blog mais bel et bien d'avoir quelque chose de fonctionnel. 😊

Le CSS

Voici le fichier CSS (très simple) que j'utilisera pour ce TP :

Code : 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;
}

a
{
    text-decoration: none;
    color: blue;
}
```

Libre à vous de l'utiliser ou non, de le modifier, bref faites-en ce que vous voulez. 😊

Structure des tables MySQL

Eh oui, cette fois nous allons travailler avec non pas une mais deux tables :

- `billets` : liste des billets du blog
- `commentaires` : liste des commentaires du blog pour chaque billet



On va vraiment stocker tous les commentaires dans une seule table, même s'ils concernent des billets différents ?

Oui. C'est la bonne façon de faire. Tous les commentaires, quel que soit le billet auquel ils se rapportent, seront stockés dans la même table. On pourra faire le tri facilement à l'aide d'un champ `id_billet` qui indiquera pour chaque commentaire le numéro du billet associé.

Voici la structure que je propose pour la table `billets` :

- `id` (int) : identifiant du billet, clé primaire et auto_increment
- `titre` (varchar 255) : titre du billet
- `contenu` (text) : contenu du billet
- `date_creation` (datetime) : date et heure de création du billet

De même, voici la structure que l'on va utiliser pour la table `commentaires` :

- `id` (int) : identifiant du commentaire, clé primaire et auto_increment
- `id_billet` (int) : identifiant du billet auquel correspond ce commentaire
- `auteur` (varchar 255) : auteur du commentaire
- `commentaire` (text) : contenu du commentaire
- `date_commentaire` (datetime) : date et heure à laquelle le commentaire a été posté.

C'est vraiment la base. Vous pouvez rajouter d'autres champs si vous le désirez. Par exemple, on n'a pas défini de champ `auteur` pour les billets.



Notez qu'il est possible d'ajouter des champs à tout moment, comme nous l'avons vu il y a peu. L'interface phpMyAdmin propose des options pour cela.

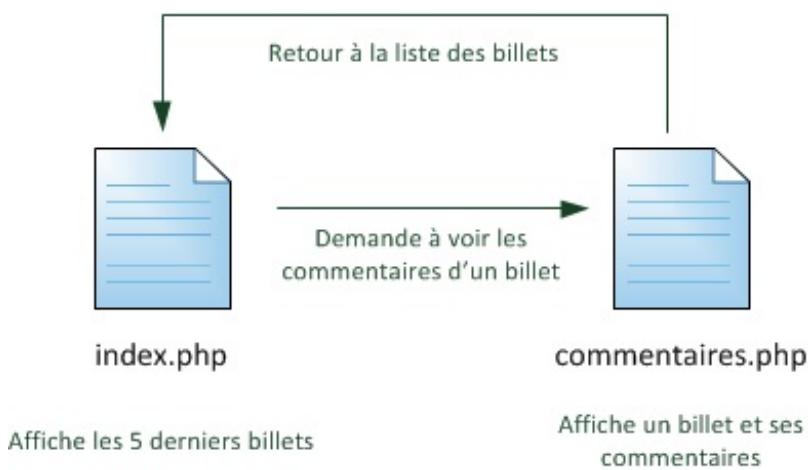
Comme nous n'allons pas créer les formulaires d'ajout de billets et de commentaires dans un premier temps, je vous conseille de remplir vous-mêmes les tables à l'aide de phpMyAdmin après les avoir créées.

Si vous êtes du genre flemmard, vous pouvez aussi télécharger mes tables toutes prêtes avec quelques données à l'intérieur, mais je vous recommande de vous entraîner à les créer vous-mêmes.

[Télécharger les tables du blog](#)

Structure des pages PHP

Etant donné que nous nous concentrons sur l'affichage, la structure des pages reste très simple :



Le visiteur arrive d'abord sur l'`index` où sont affichés les derniers billets. S'il choisit d'afficher les commentaires de l'un d'eux, il charge la page `commentaires.php` qui affichera le billet sélectionné ainsi que tous ses commentaires. Bien entendu, il faudra envoyer un paramètre à la page `commentaires.php` pour qu'elle sache quoi afficher, je vous laisse deviner lequel.



Il sera possible de revenir à la liste des billets depuis les commentaires à l'aide d'un lien de retour.

A vous de jouer !

Je vous en ai assez dit, la réalisation de ce TP devrait être relativement simple pour vous si vous avez bien suivi jusqu'ici.

N'oubliez pas les éléments essentiels de sécurité, notamment la protection de tous les textes par `htmlspecialchars()`. Et ne faites jamais confiance à l'utilisateur ! 😊

Correction

Si vous lisez ces lignes, c'est que vous devez être venu à bout de ce TP. Celui-ci ne présentait pas de difficultés particulières mais il était l'occasion de vous exercer un peu plus avec MySQL tout en faisant appel aux fonctions et dates en SQL.

index.php : la liste des derniers billets

Le TP est constitué de 2 pages, voici la correction que je propose pour la page index.php qui liste les derniers billets du blog :

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>Mon blog</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
        <link href="style.css" rel="stylesheet" type="text/css" />
    </head>

    <body>
        <h1>Mon super blog !</h1>
        <p>Derniers billets du blog :</p>

        <?php
        // Connexion à la base de données
        try
        {
            $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
            $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
$pdo_options);

            // On récupère les 5 derniers billets
            $req = $bdd->query('SELECT id, titre, contenu,
DATE_FORMAT(date_creation, \'%d/%m/%Y à %H%imin%ss\') AS
date_creation_fr FROM billets ORDER BY date_creation DESC LIMIT 0,
5');

            while ($donnees = $req->fetch())
            {
                ?>
                <div class="news">
                    <h3>
                        <?php echo htmlspecialchars($donnees['titre']); ?>
                        <em>le <?php echo $donnees['date_creation_fr']; ?></em>
                    </h3>

                    <p>
                    <?php
                    // On affiche le contenu du billet
                    echo nl2br(htmlspecialchars($donnees['contenu']));
                    ?>
                    <br />
                    <em><a href="commentaires.php?billet=<?php echo
$donnees['id']; ?>">Commentaires</a></em>
                    </p>
                </div>
                <?php
            } // Fin de la boucle des billets
            $req->closeCursor();
        }
    
```

```

catch(Exception $e)
{
    die('Erreur : ' . $e->getMessage());
}

?>

</body>
</html>

```

Vous constaterez que tous les textes, y compris les titres, sont protégés par `htmlspecialchars()`. J'utilise par ailleurs une fonction qui doit être nouvelle pour vous : `nl2br()`. Elle permet de convertir les retours à la ligne en balises HTML `
`. C'est une fonction dont vous aurez sûrement besoin pour conserver facilement les retours à la ligne saisis dans les formulaires.

Côté SQL, cette page ne fait qu'une seule requête : celle qui récupère les 5 derniers billets.

Code : SQL

```

SELECT id, titre, contenu, DATE_FORMAT(date_creation, '%d/%m/%Y à
%H%imin%ss') AS date_creation_fr FROM billets ORDER BY
date_creation DESC LIMIT 0, 5

```

On récupère toutes les données qui nous intéressent dans cette table, en mettant en forme au passage la date. Pour cela, on utilise la fonction scalaire `DATE_FORMAT` qui nous permet d'obtenir une date dans un format français.

Les billets sont ordonnés par date décroissante, le plus récent étant donc en haut de la page.

Enfin, chaque billet est suivi d'un lien vers la page `commentaires.php` en transmettant le numéro du billet dans l'URL :

Code : PHP

```

<a href="commentaires.php?billet=<?php echo $donnees['id']; ?>">Commentaires</a>

```

commentaires.php : affichage d'un billet et de ses commentaires

Cette page présente des similitudes avec la précédente mais elle est un peu plus complexe. En effet, pour afficher un billet ainsi que ses commentaires, nous avons besoin de faire 2 requêtes SQL :

- Une requête pour récupérer le contenu du billet
- Une requête pour récupérer les commentaires associés au billet

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>Mon blog</title>
        <meta http-equiv="Content-Type" content="text/html;

```

```
charset=iso-8859-1" />
<link href="style.css" rel="stylesheet" type="text/css" />
</head>

<body>
    <h1>Mon super blog !</h1>
    <p><a href="index.php">Retour à la liste des billets</a></p>

    <?php
    // Connexion à la base de données
    try
    {
        $pdo_options [PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
        $bdd = new PDO ('mysql:host=localhost;dbname=test', 'root', '',
        $pdo_options);

        // Récupération du billet
        $req = $bdd->prepare('SELECT id, titre, contenu,
DATE_FORMAT(date_creation, \'%d/%m/%Y à %H%imin%ss\') AS
date_creation_fr FROM billets WHERE id = ?');
        $req->execute(array($_GET['billet']));
        $donnees = $req->fetch();
    ?>

        <div class="news">
            <h3>
                <?php echo htmlspecialchars($donnees['titre']); ?>
                <em>le <?php echo $donnees['date_creation_fr']; ?></em>
            </h3>

            <p>
                <?php
                    echo nl2br(htmlspecialchars($donnees['contenu']));
                ?>
            </p>
        </div>

        <h2>Commentaires</h2>

        <?php
            $req->closeCursor(); // Important : on libère le curseur pour la
prochaine requête

            // Récupération des commentaires
            $req = $bdd->prepare('SELECT auteur, commentaire,
DATE_FORMAT(date_commentaire, \'%d/%m/%Y à %H%imin%ss\') AS
date_commentaire_fr FROM commentaires WHERE id_billet = ? ORDER BY
date_commentaire');
            $req->execute(array($_GET['billet']));

            while ($donnees = $req->fetch())
            {
                ?>
                <p><strong><?php echo htmlspecialchars($donnees['auteur']); ?>
                </strong> le <?php echo $donnees['date_commentaire_fr']; ?></p>
                <p><?php echo nl2br(htmlspecialchars($donnees['commentaire'])); ?>
            ?></p>
                <?php
            } // Fin de la boucle des commentaires
            $req->closeCursor();

        }
    catch (Exception $e)
    {
        die('Erreur : '. $e->getMessage());
    }
?>
</body>
```

```
</html>
```

Ce code est un peu gros mais on peut le découper en 2 parties :

- Affichage du billet
- Affichage des commentaires

La requête qui récupère le billet ressemble à celle de la page précédente, à la différence près qu'il s'agit d'une requête préparée car elle dépend d'un paramètre : l'id du billet (fourni par `$_GET['billet']`) qu'on a reçu dans l'URL).

Comme on récupère forcément un seul billet, il est inutile de faire une boucle. L'affichage est identique à celui qu'on faisait pour chaque billet sur la page précédente, à l'exception du lien vers la page des commentaires qui ne sert plus à rien (puisque nous sommes sur la page des commentaires 😊).

On pense à libérer le curseur après l'affichage du billet avec :

Code : PHP

```
<?php  
$req->closeCursor();  
?>
```

En effet, cela permet de "terminer" le traitement de la requête pour pouvoir traiter la prochaine requête sans problème.

La récupération des commentaires se fait ensuite via la requête suivante :

Code : SQL

```
SELECT auteur, commentaire, DATE_FORMAT(date_commentaire, '%d/%m/%Y  
à %H%imin%ss') AS date_commentaire_fr FROM commentaires WHERE  
id_billet = ? ORDER BY date_commentaire
```



Vous noterez que l'on ne se connecte à la base de données qu'une fois par page. Pas besoin donc de se reconnecter pour effectuer cette seconde requête.

On récupère avec cette requête tous les commentaires liés au billet correspondant à l'id reçu dans l'URL. Les commentaires sont triés par date croissante comme c'est habituellement le cas sur les blogs, mais vous pouvez changer cet ordre si vous le désirez, c'est facile. 😊

Aller plus loin

Ce TP ne concernait que la structure de base d'un blog avec commentaires. Comme il est relativement simple, les possibilités d'extension ne manquent pas. 😊

Alors, comment pourrait-on améliorer notre blog ? Voici quelques suggestions que je vous conseille d'étudier, cela vous fera progresser. 😊

Un formulaire d'ajout de commentaire

Sur la page `commentaires.php`, rajoutez un formulaire pour que n'importe quel visiteur puisse poster un commentaire.

Ce formulaire redirigera vers une page qui enregistrera le commentaire puis qui redirigera vers la liste des commentaires, de la même façon qu'on l'avait fait avec le Mini-Chat.



C'est de votre niveau vous avez déjà réussi à le faire, allez-y ! 😊

Utiliser les includes

Certaines portions de codes sont un peu répétitives. Par exemple, on retrouve le même bloc affichant un billet sur la page des billets sur celle des commentaires.

Il serait peut-être plus simple d'avoir un seul code dans un fichier que l'on inclutrait ensuite depuis `index.php` et `commentaires.php`. 😊

Vérifier si le billet existe sur la page des commentaires

Imaginez que le visiteur s'amuse à modifier l'URL de la page des commentaires. Par exemple s'il essaie d'accéder à `commentaires.php?billet=819202` et que le billet n° 819202 n'existe pas, il n'aura pas de message d'erreur (en fait le contenu de la page sera vide). Pour que votre site paraisse un peu plus sérieux, vous devriez afficher une erreur.

Pour cela, regardez si la requête qui récupère le contenu du billet renvoie des données. Pour cela, le plus simple est de vérifier après le `fetch()` si la variable `$donnees` est vide ou non grâce à la fonction `empty()`.

Ainsi, si la variable est vide, vous pourrez afficher un message d'erreur comme "Ce billet n'existe pas". Sinon, vous afficherez le reste de la page normalement. 😊

Paginer les billets et commentaires

Quand vous commencerez à avoir beaucoup de billets (et beaucoup de commentaires) vous voudrez peut-être ne pas tout afficher sur la même page. Pour cela, il faut créer un système de pagination.

Supposons que vous souhaitez afficher uniquement 5 billets par page. Si vous voulez afficher des liens vers chacune des pages, il faut savoir combien votre blog comporte de billets.

Par exemple, si vous avez 5 billets, il n'y aura qu'une seule page. Si vous avez 12 billets, il y aura 3 pages. Pour connaître le nombre de billets, une requête SQL avec COUNT (*) est indispensable :

Code : SQL

```
SELECT COUNT(*) AS nb_billets FROM billets
```

Une fois ce nombre de billets récupéré, vous pouvez trouver le nombre de pages et créer des liens vers chacune d'elles :

Page : 1 2 3 4

Chacun de ces nombres amènera vers la page correspondante et ajoutera dans l'url le numéro de celle-ci :

Code : HTML

```
<a href="index.php?page=2">2</a>
```

A l'aide du paramètre `$_GET['page']` vous pourrez déterminer quelle page vous devez afficher. A vous d'adapter la requête SQL pour commencer uniquement à partir du billet n° X. Par exemple, si vous demandez à afficher la page 2, vous voudrez afficher uniquement les billets n° 5 à 9 (n'oubliez pas qu'on commence à compter à partir de 0 !). Revoyez la section sur LIMIT au besoin.



Et si aucun numéro de page n'est défini dans l'URL, lorsqu'on arrive la première fois sur le blog ?

Dans ce cas, si `$_GET['page']` n'est pas définie, vous devrez considérer que le visiteur veut afficher la page 1 (la plus récente).

Ca demande un peu de réflexion mais le jeu en vaut la chandelle ! N'hésitez pas à demander de l'aide sur les forums au besoin.

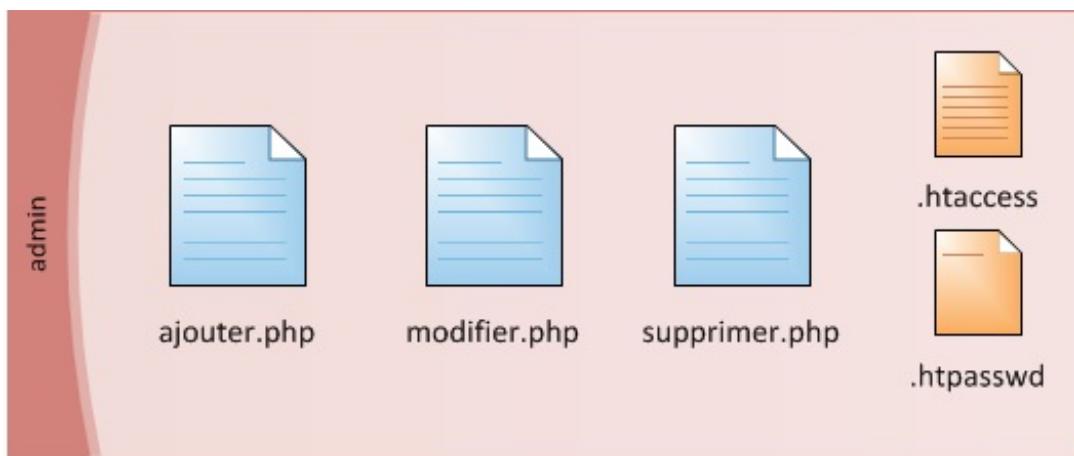


Réaliser une interface d'administration du blog

C'est probablement l'amélioration la plus longue. Il faudra créer des pages qui permettent de modifier, supprimer et ajouter des nouveaux billets.

Un problème cependant : comment protéger l'accès à ces pages ? En effet, vous devriez être le seul à avoir accès à votre interface d'administration, sinon n'importe qui pourra ajouter des billets s'il connaît l'URL de la page d'administration !

Plusieurs techniques existent pour protéger l'accès à l'administration. Le plus simple dans ce cas est de créer un sous-dossier `admin` qui contiendra tous les fichiers d'administration du blog (`ajouter.php`, `modifier.php`, `supprimer.php`...). Ce dossier `admin` sera entièrement protégé à l'aide des fichiers `.htaccess` et `.htpasswd`, ce qui fait que personne ne pourra charger les pages qu'il contient à moins de connaître le login et le mot de passe.



Pour en savoir plus sur la protection d'un dossier avec ces fichiers `.htaccess` et `.htpasswd`, je vous invite à consulter [cette annexe au cours](#). 😊

Allez, au boulot ! 😊

Je vous avais prévenu : ce TP ne vous montre que les bases du blog (l'affichage) mais c'est à vous de jouer si vous voulez construire le reste du blog ! Réalisez les améliorations que je propose, cela vous prendra sûrement un peu de temps mais vous progresserez beaucoup en faisant cela.

N'hésitez pas à vous faire aider si une amélioration vous bloque trop longtemps. N'abandonnez pas à la première difficulté !



📘 Les jointures entre tables

MySQL permet de travailler avec plusieurs tables à la fois. Un des principaux intérêts d'une base de données est de pouvoir créer des relations entre les tables, de pouvoir les lier entre elles.

Pour le moment, nous n'avons travaillé que sur une seule table à la fois. Dans la pratique, vous aurez certainement plusieurs tables dans votre base, dont la plupart seront interconnectées. Cela vous permettra de mieux découper vos informations, d'éviter des répétitions et de rendre ainsi vos données plus faciles à gérer.

Tenez, par exemple, dans notre table `jeux_video`, on répète à chaque fois le nom du possesseur du jeu. Le mot "Patrick" est écrit de nombreuses fois dans la table. Imaginez que l'on souhaite stocker aussi son nom de famille, son adresse, son numéro de téléphone... On ne va quand même pas recopier ces informations pour chaque jeu qu'il possède ! Il est temps de créer une autre table et de la lier ! 😊

Modélisation d'une relation

Si je voulais stocker le nom, prénom et numéro de téléphone de chaque propriétaire de jeu vidéo dans notre table `jeux_video`, il n'y aurait pas d'autre solution que de dupliquer ces informations sur chaque entrée... Cependant ce serait bien trop répétitif, regardez ce que ça donnerait :

ID	nom	prenom	nom_posesseur	tel	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	Florent	Dugommier	01 44 77 21 33	NES	4	1	Un jeu d'anthologie !
2	Sonic	Patrick	Lejeune	03 22 17 41 22	Megadrive	2	1	Pour moi, le meilleur jeu au monde !
3	Zelda : ocarina of time	Florent	Dugommier	01 44 77 21 33	Nintendo 64	15	1	Un jeu grand, beau et complet comme on en voit rarement de nos jours
4	Mario Kart 64	Florent	Dugommier	01 44 77 21 33	Nintendo 64	25	4	Un excellent jeu de kart !
5	Super Smash Bros Melee	Michel	Doussand	04 11 78 02 00	GameCube	55	4	Un jeu de baston délirant !

Comme vous le voyez, le nom, le prénom et le numéro de téléphone de Florent apparaissent autant de fois qu'il possède jeux vidéo, et il en irait de même pour Patrick et Michel. Il faut à tout prix éviter ces répétitions.

Ce que je vous propose, c'est de créer une autre table, que l'on nommera par exemple `proprietaires`, qui centralisera les informations des propriétaires des jeux vidéo :

Table propriétaires

ID	prenom	nom	tel
1	Florent	Dugommier	01 44 77 21 33
2	Patrick	Lejeune	03 22 17 41 22
3	Michel	Doussand	04 11 78 02 00

Cette table liste tous les propriétaires de jeux connus et leur attribue à chacun un ID. Les propriétaires n'apparaissent qu'une seule fois, il n'y a pas de doublon.

Maintenant, il faut modifier la structure de la table `jeux_video` pour faire référence aux propriétaires. Pour cela, le mieux est de créer un champ `ID_proprietaire` qui indique le numéro du propriétaire dans l'autre table :

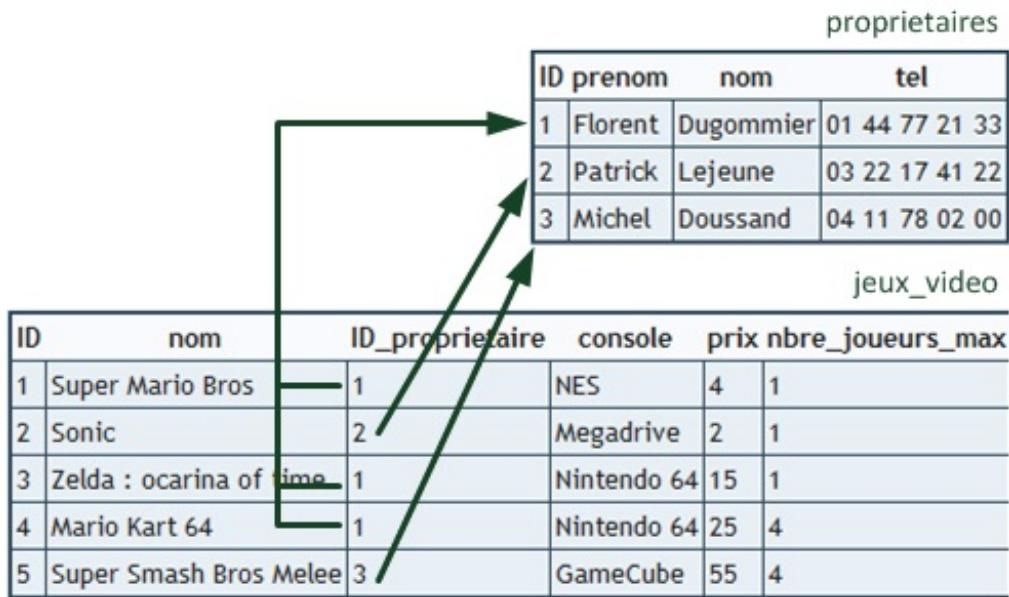
Table jeux_video

ID	nom	ID_proprietaire	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	1	NES	4	1	Un jeu d'anthologie !
2	Sonic	2	Megadrive	2	1	Pour moi, le meilleur jeu au monde !
3	Zelda : ocarina of time	1	Nintendo 64	15	1	Un jeu grand, beau et complet comme on en voit rarement de nos jours

4	Mario Kart 64	1	Nintendo 64	25	4	Un excellent jeu de kart !
5	Super Smash Bros Melee	3	GameCube	55	4	Un jeu de baston délirant !

Le nouveau champ `ID_proprietaire` est de type `INT`. Il permet de faire référence à une entrée précise de la table `proprietaires`.

On peut maintenant considérer que les tables sont reliées à travers ces ID de propriétaires.



MySQL sait donc que l'`ID_proprietaire` n°1 dans la table des `jeux_video` correspond à Florent ?

Non, il ne le sait pas. Il ne voit que des nombres et il ne fait pas la relation entre les 2 tables. Il va falloir lui expliquer cette relation dans une requête SQL : on va faire ce qu'on appelle une **jointure** entre les 2 tables.

Qu'est-ce qu'une jointure ?

Nous avons donc maintenant 2 tables :

- jeux_video
- proprietaires

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

Cependant, lorsqu'on récupère la liste des jeux, si on souhaite obtenir le nom du propriétaire, il va falloir adapter la requête pour récupérer aussi les informations issues de la table proprietaires. Pour cela, on doit faire ce qu'on appelle une **jointure**.

Il existe plusieurs types de jointures, qui nous permettent de choisir exactement les données que l'on veut récupérer. Je vous propose d'en découvrir 2, les plus importantes :

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

Il est important de bien comprendre la différence entre une jointure interne et une jointure externe.

Pour cela, imaginons que nous ayons une 4ème personne dans la table des propriétaires, un certain Romain Vipelli, qui ne possède aucun jeu :

Table proprietaires			
ID	prenom	nom	tel
1	Florent	Dugommier	01 44 77 21 33
2	Patrick	Lejeune	03 22 17 41 22
3	Michel	Doussand	04 11 78 02 00
4	Romain	Vipelli	01 21 98 51 01

Romain Vipelli est référencé dans la table proprietaires mais il n'apparaît nulle part dans la table jeux_video car il ne possède aucun jeu.

Si vous récupérez les données des 2 tables à l'aide :

- D'une **jointure interne** : Romain Vipelli n'apparaîtra pas dans les résultats de la requête. La jointure interne force les données d'une table à avoir une correspondance dans l'autre.
- D'une **jointure externe** : vous aurez toutes les données de la table des propriétaires même s'il n'y a pas de correspondance dans l'autre table des jeux vidéo, donc Romain Vipelli, qui pourtant ne possède aucun jeu vidéo, apparaîtra.

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

Voici par exemple les données que l'on récupérerait avec une jointure interne :

Jointure interne

nom_jeu	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...

On obtient les jeux et leurs propriétaires, mais Romain qui ne possède pas de jeu n'apparaît pas du tout. En revanche, avec une jointure externe :

Jointure externe	
nom_jeu	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...
NULL	Romain

Romain apparaît maintenant. Comme il ne possède pas de jeu, il n'y a aucun nom de jeu indiqué (NULL).

Nous allons maintenant voir dans la pratique comment réaliser ces deux types de jointures.

Les jointures internes

Une jointure interne peut être effectuée de 2 façons différentes :

- A l'aide du mot-clé WHERE : c'est l'ancienne syntaxe, toujours utilisée aujourd'hui, qu'il faut donc connaître mais que vous devriez éviter d'utiliser si vous avez le choix.
- A l'aide du mot-clé JOIN : c'est la nouvelle syntaxe qu'il est recommandé d'utiliser. Elle est plus efficace et plus lisible.

Ces deux techniques produisent exactement le même résultat, mais il faut les connaître toutes les deux. 😊

Jointure interne avec WHERE (ancienne syntaxe)

Construction d'une jointure interne pas à pas

Pour réaliser ce type de jointure, on va sélectionner des champs des 2 tables et indiquer le nom de ces 2 tables dans la clause FROM :

Code : SQL

```
SELECT nom, prenom FROM proprietaires, jeux_video
```

Cependant ça ne fonctionnera pas car ce n'est pas suffisant. En effet, le champ nom apparaît dans les 2 tables : une fois pour le nom du propriétaire, et une autre fois pour le nom du jeu vidéo. C'est ce qu'on appelle une **colonne ambiguë** car MySQL ne sait pas s'il doit récupérer un nom de personne (comme Dugommier) ou un nom de jeu (comme Super Mario Bros). Bref, il est un peu perdu. 🤔

L'astuce consiste à marquer le nom de la table devant le nom du champ comme ceci :

Code : SQL

```
SELECT jeux_video.nom, proprietaires.prenom FROM proprietaires,  
jeux_video
```

Ainsi, on demande clairement de récupérer le nom du jeu et le prénom du propriétaire avec cette requête.



Le champ prenom n'est pas ambigu, car il n'apparaît que dans la table proprietaires. On pourrait donc se passer d'écrire le préfixe proprietaires devant, mais ça ne coûte rien de le faire et c'est plus clair : on voit immédiatement en lisant la requête de quelle table est issu ce champ.

Il reste encore à lier les 2 tables entre elles. En effet, les jeux et leurs propriétaires ont une correspondance via le champ ID_proprietaire (de la table jeux_video) et le champ ID (de la table proprietaires). On va indiquer cette liaison dans un WHERE comme ceci :

Code : SQL

```
SELECT jeux_video.nom, proprietaires.prenom  
FROM proprietaires, jeux_video  
WHERE jeux_video.ID_proprietaire = proprietaires.ID
```



Comme la requête devient longue, je me permets de l'écrire sur plusieurs lignes. Cette écriture est tout à fait autorisée et a l'avantage d'être plus lisible.

On indique bien que le champ `ID_proprietaire` de la table `jeux_video` correspond au champ `ID` de la table `proprietaires`. Cela établit la correspondance entre les 2 tables telle qu'on l'avait définie dans le schéma au début du chapitre.

Notre requête est enfin complète, vous pouvez l'essayer. 😊

Vous devriez récupérer les données suivantes :

nom	prenom
Super Mario Bros	Florent
Sonic	Patrick
...	...

Utilisez les alias !

Nous avons appris à utiliser les alias lorsque nous avons découvert les fonctions SQL. Cela nous permettait de créer ce que j'appelais des "champs virtuels" pour représenter le résultat des fonctions.

Il est fortement conseillé d'utiliser des alias lorsqu'on fait des jointures. On peut utiliser des alias sur les noms de champs (comme on l'avait fait) :

Code : SQL

```
SELECT jeux_video.nom AS nom_jeu, proprietaires.prenom AS
prenom_proprietaire
FROM proprietaires, jeux_video
WHERE jeux_video.ID_proprietaire = proprietaires.ID
```

On récupèrera donc 2 champs : `nom_jeu` et `prenom_proprietaire`. Ces alias permettent de donner un nom plus clair aux champs que l'on récupère.

nom_jeu	prenom_proprietaire
Super Mario Bros	Florent
Sonic	Patrick
...	...

Il est possible aussi de donner un alias aux noms des tables, ce qui est fortement recommandé pour donner un nom plus court et plus facile à écrire. En général, on crée des alias de tables d'une lettre ou deux correspondant à leurs initiales, comme ceci :

Code : SQL

```
SELECT j.nom AS nom_jeu, p.prenom AS prenom_proprietaire
FROM proprietaires AS p, jeux_video AS j
WHERE j.ID_proprietaire = p.ID
```

Comme vous le voyez, la table `jeux_video` a pour alias la lettre `j` et `proprietaires` la lettre `p`. On réutilise ces alias dans toute la requête, ce qui la rend plus courte à écrire (et plus lisible aussi au final 😊).

Notez que le mot-clé `AS` est en fait facultatif, les développeurs ont tendance à l'omettre. Vous pouvez donc tout simplement le retirer de la requête :

Code : SQL

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p, jeux_video j
WHERE j.ID_proprietaire = p.ID
```

Jointure interne avec JOIN (nouvelle syntaxe)

Bien qu'il soit possible de faire une jointure interne avec un `WHERE` comme on vient de le voir, c'est une ancienne syntaxe et on recommande aujourd'hui d'utiliser plutôt `JOIN`. Il faut dire que nous étions habitués à utiliser le `WHERE` pour filtrer les données, alors que nous l'utilisons ici pour associer des tables et récupérer plus de données.

Pour éviter de confondre le `WHERE` "traditionnel" qui filtre les données et le `WHERE` de jointure que l'on vient de découvrir, on va utiliser la syntaxe `JOIN`.

Pour rappel, voici la requête qu'on utilisait avec un `WHERE` :

Code : SQL

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p, jeux_video j
WHERE j.ID_proprietaire = p.ID
```

Avec un `JOIN`, on écrirait cette même requête de la façon suivante :

Code : SQL

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p
INNER JOIN jeux_video j
ON j.ID_proprietaire = p.ID
```

Cette fois, on récupère les données depuis une table principale (ici `proprietaires`) et on fait une jointure interne (`INNER JOIN`) avec une autre table (`jeux_video`). La liaison entre les champs est faite dans la clause `ON` un peu plus loin.

Le fonctionnement reste le même, on récupère les mêmes données que tout à l'heure avec la syntaxe `WHERE`.

Si vous voulez filtrer (`WHERE`), ordonner (`ORDER BY`) ou limiter les résultats (`LIMIT`), vous devez le faire à la fin de la requête, après le "`ON j.ID_proprietaire = p.ID`". Par exemple :

Code : SQL

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p
INNER JOIN jeux_video j
ON j.ID_proprietaire = p.ID
WHERE j.console = 'PC'
ORDER BY prix DESC
LIMIT 0, 10
```

Traduction (inspirez un grand coup avant de lire) : "Récupère le nom du jeu et le prénom du propriétaire dans les tables propriétaires et jeux_video, la liaison entre les tables se fait entre les champs ID_proprietaire et ID, prends uniquement les jeux qui tournent sur PC, trie-les par prix décroissant et ne prends que les 10 premiers."

Faut s'accrocher avec des requêtes de cette taille-là. 😎

Les jointures externes

Les jointures externes permettent de récupérer toutes les données, même celles qui n'ont pas de correspondance. On pourra ainsi obtenir Romain Vipelli dans la liste même s'il ne possède pas de jeu vidéo.

Cette fois, la seule syntaxe disponible est à base de JOIN. Il y a deux écritures à connaître : LEFT JOIN et RIGHT JOIN. Cela revient pratiquement au même, avec une subtile différence que nous allons voir.

LEFT JOIN : récupérer toute la table de gauche

Reprendons la jointure à base de INNER JOIN et remplaçons tout simplement INNER par LEFT :

Code : SQL

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p
LEFT JOIN jeux_video j
ON j.ID_proprietaire = p.ID
```

propriétaires est appelée la "table de gauche" et jeux_video la "table de droite". Le LEFT JOIN demande à récupérer tout le contenu de la table de gauche (donc tous les propriétaires) même s'ils n'ont pas d'équivalence dans la table jeux_video.

nom_jeu	prenom_proprietaire
Super Mario Bros	Florent
Sonic	Patrick
...	...
NULL	Romain

Romain apparaît désormais dans les résultats de la requête grâce à la jointure externe. Comme il ne possède aucun jeu, la colonne du nom du jeu est vide.

RIGHT JOIN : récupérer toute la table de droite

Le RIGHT JOIN demande à récupérer toutes les données de la table dite "de droite", même si celle-ci n'a pas d'équivalent dans l'autre table. Prenons la requête suivante :

Code : SQL

```
SELECT j.nom nom_jeu, p.prenom prenom_proprietaire
FROM proprietaires p
RIGHT JOIN jeux_video j
ON j.ID_proprietaire = p.ID
```

La table de droite est jeux_video. On récupèrerait donc tous les jeux, même ceux qui n'ont pas de propriétaire associé.



Comment est-ce possible qu'un jeu n'ait pas de propriétaire associé ?

Il y a 2 cas possibles :

- Soit le champ ID_proprietaire contient une valeur qui n'a pas d'équivalent dans la table des propriétaires, par exemple "56".
- Soit le champ ID_proprietaire vaut NULL, c'est-à-dire que personne ne possède ce jeu. C'est le cas notamment du jeu Bomberman dans la table que vous avez téléchargée (voir tableau ci-dessous).

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

Dans ce cas, Bomberman n'appartient à personne. Avec la requête RIGHT JOIN que l'on vient de voir, on obtiendra toutes les lignes de la table de droite (`jeux_video`) même si elles n'ont aucun lien avec la table `proprietaires`, comme c'est le cas ici pour Bomberman.

On obtiendra donc les données suivantes :

nom_jeu	prenom_proprietaire
Super Mario Bros	Florent
Sonic	Patrick
...	...
Bomberman	NULL

Cette introduction aux jointures devrait vous avoir permis de comprendre comment on peut séparer les informations dans différentes tables pour éviter de répéter inutilement des informations. Cette plus grande séparation des données nous donne plus de flexibilité : ainsi, on peut récupérer seulement la liste des propriétaires puisqu'elle est stockée dans une table à part.

Vous devriez, pour vous entraîner, faire la même séparation par exemple pour la liste des consoles. Vous pourriez créer une table `consoles` qui contiendrait le nom de la console, son année de sortie, le nom de son fabricant (Nintendo, Sega, Microsoft...). Et pour aller plus loin vous pourriez aussi stocker les fabricants dans une table `fabricants` !

Enfin, sachez qu'il est possible d'aller plus loin dans l'association des tables avec MySQL. Si on utilise des tables de type InnoDB (contrairement au type MyISAM qu'on a utilisé par défaut jusqu'ici), MySQL peut connaître la relation entre les tables à tout instant, et pas seulement au moment de la requête. Cela permet de faire par exemple en sorte que si on supprime un propriétaire, cela supprime automatiquement tous les jeux vidéo qu'il possède dans l'autre table.

Partie 4 : Utilisation avancée de PHP

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

Créer des images en PHP

Vous savez quoi ? Il y a des gens qui croient que le PHP n'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 demander de l'activer tout simplement. 😊

La procédure à suivre est exactement la même que celle qu'on avait vue pour [activer PDO](#) lorsque nous avons découvert les bases de données. Sous WAMP par exemple, faites un clic gauche sur l'icône de WAMP dans la barre des tâches, puis allez dans le menu PHP / Extensions PHP et cochez php_gd2.



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

Cela 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

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

1. Nous allons découvrir ce qu'est un *header*.
2. Ensuite, nous allons créer l'image de base.
3. Enfin, nous verrons comment on affiche l'image concrètement.

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

Voilà, c'est assez simple. Ce code signifiera pour le navigateur que l'on envoie 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.

A partir d'une image vide

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 renvoie une information, que vous devez mettre dans une variable (par exemple `$image`). Ce qui nous donne :

Code : 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 je n'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 `open`. Tiens tiens, ça vous rappelle quelque chose ?

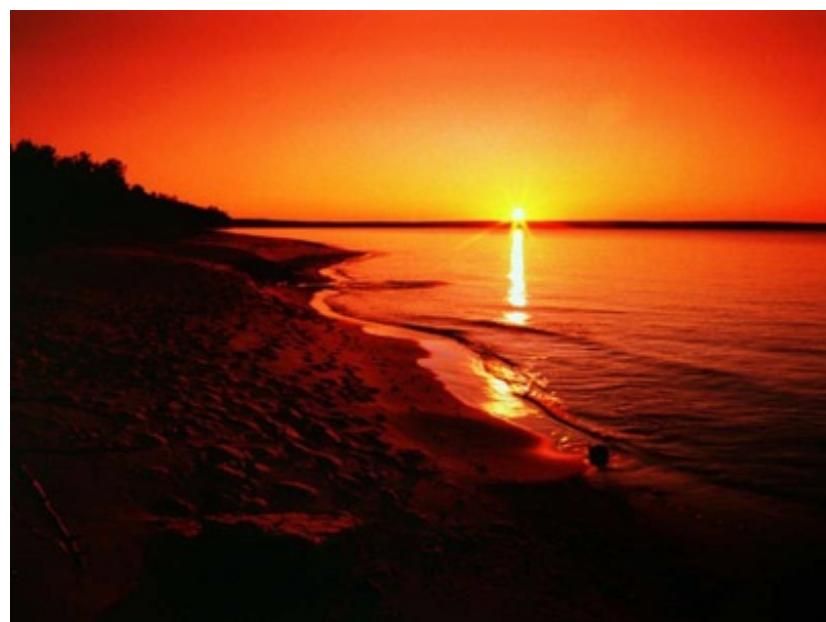
A partir d'une image existante

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 écrire du texte dedans, à dessiner des cercles, des carrés etc... Nous allons apprendre tout cela juste après.

Je souhaite vous montrer ici comment faire 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 s'utilisent 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 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
header ("Content-type: image/png"); // 1 : on indique qu'on va
envoyer une image PNG
$image = imagecreate(200,50); // 2 : on crée une nouvelle image de
taille 200x50
// 3 : on s'amuse avec notre image (on va apprendre à le faire)
imagepng($image); // 4 : on a fini 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 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 :

Code : HTML

```

```

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 😊

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
$image = imagecreate(200,50);
// on fait joujou avec notre image
imagepng($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 :

Code : HTML

```

```

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.



Mais... Mais ??? Si je teste ces codes, ça crée une image toute blanche ! C'est nul, il ne 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 :

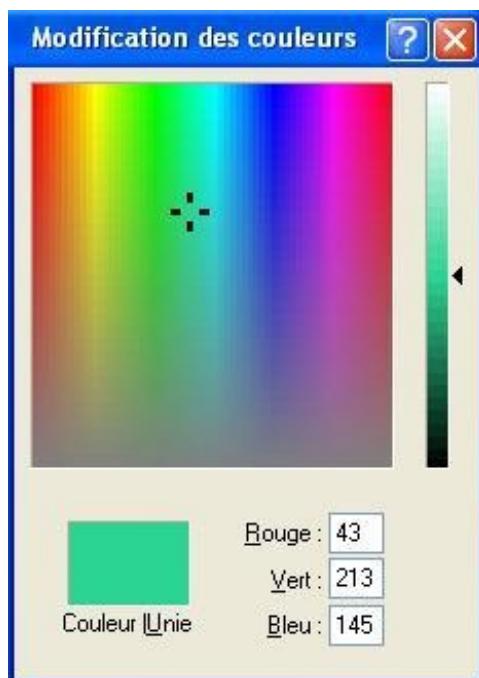
255 128 0

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 renvoie la couleur dans une variable. Grâce à cette fonction, on va pouvoir se créer des "variables-couleur" qui vont nous être utiles pour indiquer la couleur ensuite.

Voici quelques exemples de création de couleur :

Code : 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);
?>
```

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 placer `$blanc` en premier.

Voilà, vous savez maintenant créer toutes les couleurs de l'arc-en-ciel en PHP. 😊

Ecrire du texte

Nous voici enfin dans le vif du sujet (ouf !). Nous avons une belle image avec un magnifique fond orange, et nous voulons écrire du texte dedans.

Avec la fonction `imagestring`, c'est facile !

Cette fonction prend beaucoup de paramètres. Elle s'utilise comme ceci :

Code : PHP

```
<?php  
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 sent 🍪). 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 :



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

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 199, 49 se trouve tout en bas à droite.

- `$texte_a_ecrire`, c'est le... texte que vous voulez écrire. Non non, y'a pas de piège. 😊
- `$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
header ("Content-type: image/png");
$image = imagecreate(200,50);

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

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

imagepng ($image);
?>
```

Essayer !

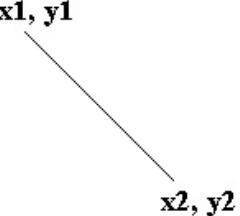
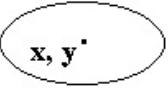
La ligne avec `imagestring` peut se traduire par : *Mets dans l'image \$image, avec la police de taille 4, aux coordonnées (35, 15), le texte "Salut les Zéros !", de couleur blanche.*

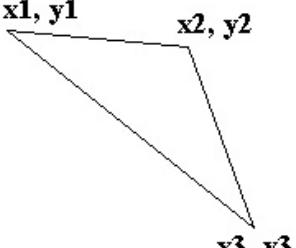
Dessiner une forme

Dessiner du texte c'est bien, mais ça serait bête si on était limité 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...

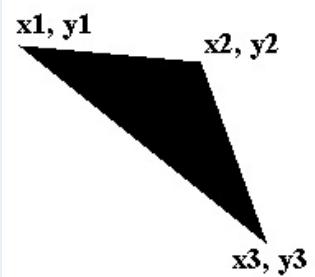
Je vais vous présenter la plupart de ces fonctions ci-dessous, et je vous montrerai ce que ça donne dans une image de taille 200x200, histoire d'avoir un aperçu 😊

Fonction	Description	Aperçu
<code>ImageSetPixel (\$image, \$x, \$y, \$couleur);</code>	Dessine un pixel aux coordonnées (x,y)	 <code>ImageSetPixel (\$image, 100, 100, \$noir);</code>
<code>ImageLine (\$image, \$x1, \$y1, \$x2, \$y2, \$couleur);</code>	Dessine une ligne entre 2 points de coordonnées (x1,y1) et (x2,y2)	 <code>ImageLine (\$image, 30, 30, 120, 120, \$noir);</code>
<code>ImageEllipse (\$image, \$x, \$y, \$largeur, \$hauteur, \$couleur);</code>	Dessine une ellipse, dont le centre est aux coordonnées (x,y), de largeur \$largeur et de hauteur \$hauteur.	 <code>ImageEllipse (\$image, 100, 100, 100, 50, \$noir);</code>

<pre>ImageFilledEllipse (\$image, \$x, \$y, \$largeur, \$hauteur, \$couleur);</pre>	<p>Pareil que ImageEllipse, sauf que l'ellipse est entièrement coloriée dans la couleur que vous avez demandée.</p>	 <pre>ImageFilledEllipse (\$image, 100, 100, 100, 50, \$noir);</pre>
<pre>ImageRectangle (\$image, \$x1, \$y1, \$x2, \$y2, \$couleur);</pre>	<p>Dessine un rectangle, dont le coin en haut à gauche est de coordonnées (x1, y1) et celui en bas à droite (x2, y2)</p>	 <pre>x1, y1</pre> <pre>x2, y2</pre> <pre>ImageRectangle (\$image, 30, 30, 160, 120, \$noir);</pre>
<pre>ImageFilledRectangle (\$image, \$x1, \$y1, \$x2, \$y2, \$couleur);</pre>	<p>Pareil que ImageRectangle, sauf que le rectangle est cette fois entièrement colorié.</p>	 <pre>x1, y1</pre> <pre>x2, y2</pre> <pre>ImageFilledRectangle (\$image, 30, 30, 160, 120, \$noir);</pre>
<pre>ImagePolygon (\$image, \$array_points, \$nombre_de_points, \$couleur);</pre>	<p>Dessine un polygône ayant un nombre de points égal à \$nombre_de_points (s'il y a 3 points, c'est donc un triangle). L'array \$array_points contient les coordonnées de tous les points du polygone dans l'ordre : x1, y1, x2, y2, x3, y3, x4, y4...</p>	 <pre>x1, y1</pre> <pre>x2, y2</pre> <pre>x3, y3</pre> <pre>\$points = array(10, 40, 120, 50, 160, 160); ImagePolygon (\$image, \$points, 3, \$noir);</pre>

```
ImageFilledPolygon  
($image,  
$array_points,  
$nombre_de_points,  
$couleur);
```

Pareil que ImagePolygon, mais cette fois le polygône est colorié à l'intérieur.



```
$points = array(10,  
40, 120, 50, 160,  
160);  
ImageFilledPolygon  
($image, $points, 3,  
$noir);
```

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) :

Code : PHP

```
<?php  
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 réaliser des opérations bien plus complexes !

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, ce 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 :

Code : PHP

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

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

Code : 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 Zéros !", $noir);  
imagecolortransparent($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. 😊

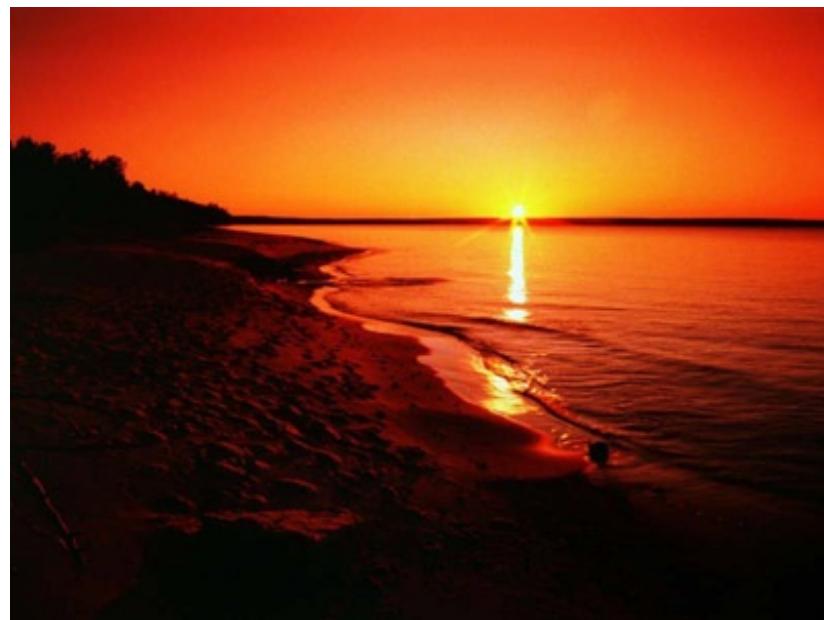
La fonction que je vais vous présenter permet de "fusionner" deux images en jouant sur un effet de transparence. Ca a l'air tordu comme ça, mais c'est en fait quelque chose de vraiment génial !

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

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

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

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

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

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

Voici le résultat que donne ce script :



imagecopymerge est une fonction vraiment sympa, parce que maintenant vous allez pouvoir "copyrighter" automatiquement toutes les images de votre site si vous le voulez.

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 les coordonnées sont (0, 0) ! 😊

- Vient ensuite la fonction `imagecopymerge`, la plus importante. Elle prend de nombreux 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 d'effet de "fusion"). Mettez un nombre autour de 60-70, en général c'est bien. 😊

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 :

Code : HTML

```

```

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 retirer 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
$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 :



Vous pouvez afficher ensuite l'image avec le code HTML :

```

```

On a créé 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.

Plusieurs paramètres sont à 0, et c'est pas vraiment la peine de chercher à comprendre pourquoi (même si ce n'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 quelque chose comme ceci :

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

Croyez-moi si vous voulez, mais ce charabia 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 que nous allons traiter ici un des aspects les plus difficiles du PHP, mais paradoxalement c'est très utile, intéressant (certains diront même "passionnant").

Il faut bien retenir que c'est un chapitre **difficile** (en fait, je suis obligé de l'étaler sur 2 chapitres), mais que ça vaut vraiment le coup de s'y intéresser. Pourquoi ? Parce que 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 cela 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 partiiii yiiiuhhhaaaa !!! 😊

Où utiliser une Regex ?

POSIX ou PCRE ?

Bonne nouvelle : vous n'aurez pas à activer quoi que ce soit pour faire des expressions régulières (contrairement à 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'autres une recherche et un 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, false (faux) 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  
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 n'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 placent 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 aimerez 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
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 pianolbanjo#	VRAI

Dans le dernier exemple, j'ai mis 2 fois la barre verticale. Cela signifie que l'on recherche guitare OU piano OU banjo.

Vous suivez toujours ? 😊

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'une chaîne.

Ainsi, si vous voulez qu'une chaîne commence par "Bonjour", il faudra utiliser la regex :

#^Bonjour#

Si vous placez 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 tester beaucoup de possibilités de recherche à la fois, tout en étant 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 une 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 perdu 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

C'est à partir de ce moment-là que les classes devraient 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, ça fonctionne aussi avec les chiffres : `[0-9]`. Si vous voulez plutôt un chiffre entre 1 et 8, tapez : `[1-8]`

Encore mieux ! Vous pouvez écrire 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. Ça donnerait : `[a-zA-Z0-9]`. C'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
<h1>Une balise de titre HTML</h1>	#<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, vous devrez placer le symbole `"^"` au début de la classe.



Mais ?! 😊

Je croyais que ce caractère servait à indiquer le début d'une chaîne ?

Oui, mais si vous le placez à l'intérieur d'une classe, il sert à dire que vous ne VOULEZ PAS de ce qui se trouve à l'intérieur de cette classe.

Ainsi, la regex suivante :

`# [^0-9] #`

... signifie que vous voulez que votre chaîne comporte au moins un caractère qui ne soit pas un chiffre.

Maintenant, je fais chauffer vos cervelles :

Chaîne	Regex	Résultat
Cette phrase contient autre chose que des chiffres	#[^0-9]#	VRAI
cette phrase contient autre chose que des majuscules et des chiffres	#[^A-Z0-9]#	VRAI

Cette phrase ne commence pas par une minuscule	#[^a-z]#	VRAI
Cette phrase ne se termine pas par une voyelle	#[^aeiou]\$#	FAUX
ScrrmmmblllGnngngnngnMmmmmffff	#[^aeiou]#	VRAI

Maintenant, faites une pause parce que ça va pas s'arranger par la suite ! 😴

Les quantificateurs

Les quantificateurs 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 suivie d'un @ (arobase), 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 de savoir 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 se trouvant directement devant. On peut ainsi autoriser le mot "chien" qu'il soit au singulier comme au pluriel, avec la regex `#chiens?#` (fonctionnera pour "chien" et "chiens").

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, comment je fais ?

Il faut utiliser des parenthèses. Par exemple, si on veut reconnaître "Ayayayayayay" (le cri de guerre de Speedy Gonzalez 😊), 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 "Ayayayoyayayoyoyoyayoy" ! 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 notre tête :

Chaîne	Regex	Résultat
eeeeee	<code>#e+#+</code>	VRAI
ooo	<code>#u?#+</code>	VRAI
magnifique	<code>#[0-9]+#+</code>	FAUX

Yahooooooo	#^Yaho+\$#	VRAI
Yahooooooo 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 n'est-ce pas ? 😊

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 répétitions.

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
eeeeee	#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, 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 connaissez 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 ont un rôle ou un sens particulier.

Alerte mon Général ! Les métacaractères s'échappent !

Dans le langage PCRE (des Regex), les métacaractères qu'il faut connaître sont les suivants :

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

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.



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 devant le point d'interrogation serait donc facultatif, mais ce n'est pas ce qu'on veut faire !

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 placer 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 montrés plus haut (# ! ^ \$ () [] { } ? + * . \) : il faut mettre un antislash devant si vous voulez les utiliser dans votre recherche. Vous remarquerez que pour utiliser un antislash il faut... un antislash devant ! Comme ceci : \\.

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 placer 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
Je suis (très) fatigué	#\(\très\)\ fatigué#	VRAI
J'ai sommeil...	#sommeil\\.\\.\\.#	VRAI

Le smiley :-\	#:-\\#	VRAI
---------------	--------	------

Le ~~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-zéèàâùüë] 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 ajouter le tiret dans la liste des caractères possibles ? Eh bien il suffit de le mettre soit au début de la classe, soit à la fin. Par exemple : [a-z0-9-] permet de chercher une lettre, un chiffre ou un tiret.

Les classes abrégées

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 place un antislash devant, on leur *donne* une signification spéciale. C'est l'inverse de ce qu'on faisait tout à l'heure : on utilisait 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" de PCRE. 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 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 (entrée 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). A noter que le 7 et le 9 commencent à être utilisés mais que nous ne les prendrons pas en compte dans nos exemples.
- 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).

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 a 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 le tout, et placer un {4} juste après pour indiquer que tout ça doit se répéter 4 fois. Ce qui nous donne 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>
<?php
if (isset($_POST['telephone']))
{
    $_POST['telephone'] = htmlspecialchars($_POST['telephone']); // On rend inoffensif le numéro
    if (preg_match("#^0[1-68]([-.]?[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" type="text" value="" />
    <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 gère tous les cas. 😊



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 arobase : @
3. Ensuite il y a le nom de domaine. Pour ce nom, même règle que pour le pseudonyme : que des minuscules, 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'arobase (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} $#
```

Vous sentez que vous commencez à parler chinois, non ? 😊

Allez, je suis en forme et de bonne humeur, je vous donne le script PHP pour tester cette Regex :

Code : PHP

```
<p>
<?php
if (isset($_POST['mail']))
{
    $_POST['mail'] = htmlspecialchars($_POST['mail']); // On rend inoffensives les lignes de code malicieuses

    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 reconnaiss 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, grrande nouvelle : MySQL comprend les Regex !

Et ça, bah c'est tout bénéf 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 ayez 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" :

```
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 et un 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 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 d'elles 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 utilisez des parenthèses, cela crée 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 effectivement inutiles. Mais pour faire un remplacement, cela 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 en fait entouré des balises !
3. Enfin, dernier paramètre, c'est le texte dans lequel on fait notre recherche / remplacement (ça vous connaissez déjà).

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

Si je schématisé le fonctionnement, ça donne ça :

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

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 *htmlspecialchars*) 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, grossièrement, ça ne marchera pas correctement si l'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

```
<?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).

Voici le résultat :

Code : PHP

```
<?php
$texte = preg_replace('#\[color=(red|green|blue|yellow|purple|olive)\](.+)\[/color#isU', '<span style="color:$1">$2', $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 la solution :

Code : PHP

```
<?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 dites-vous que moi j'ai mal à la tête ET mal aux doigts ! 😊

Mais je vais fournir quand même un dernier petit effort, voici l'heure du cadeau bonus !

Code : PHP

```
<?php
if (isset($_POST['texte']))
{
    $texte = stripslashes($_POST['texte']); // On enlève les slash qui se seraient ajoutés
    $texte = htmlspecialchars($texte); // On rend inoffensives les balises HTML
    $texte = nl2br($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('#\[color=(red|green|blue|yellow|purple|olive)\](.+)\]#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 Zéro ! <br />
    Nous avons écrit ce parser ensemble, j'espère que vous saurez apprécier de ce que nous avons fait.
</p>

<p>Amusez-vous à utiliser du bbCode. Tapez par exemple :</p>
```

```
<blockquote style="font-size:0.8em">
<p>
    Je suis un gros [b]Zéro[/b], et pourtant j'ai [i]tout appris[/i] sur http://
    Je vous [b][color=green]recommande[/color][/b] d'aller sur ce site, vous poi
</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, je ne 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 htmlspecialchars, 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.

Bon courage !

Vous en aurez besoin !

Après des chapitres comme ceux-ci, je pense que ce petit cadeau vous fera le plus grand bien. 😊



Je n'ai pas grand chose à ajouter, si ce n'est que je suis exténué mais heureux, parce que c'était vraiment un des chapitres les plus difficiles du cours et nous y sommes venus à bout ensemble. 😊



La programmation orientée objet

Il est possible de programmer en PHP de nombreuses façons différentes. C'est ce qui fait sa force : on peut en effet commencer à créer ses premières pages basiques en PHP avec très peu de connaissances au départ, mais il est aussi possible de programmer avec des outils avancés, plus complexes mais aussi plus solides et plus flexibles. 😊

La programmation orientée objet est une technique de programmation célèbre qui existe depuis des années maintenant. PHP ne l'a pas inventée : d'autres langages comme le C++, Java et Python l'utilisaient bien avant lui.

La programmation orientée objet, que nous abrègerons POO, n'est pas le Saint Graal : elle ne va pas améliorer subitement la qualité de votre site par magie. En revanche, elle va vous aider à mieux organiser votre code, à le préparer à de futures évolutions et à rendre certaines portions réutilisables pour gagner en temps et en clarté. C'est pour cela que les développeurs professionnels l'utilisent dans la plupart de leurs projets.

Qu'est-ce qu'un objet ?

Je l'ai dit et je le répète : la programmation orientée objet ne va pas instantanément révolutionner votre site web, contrairement à ce que beaucoup semblent penser. En fait, il est même probable que cela vous semble un peu inutile au début et que vous vous disiez "*Ok mais je n'en vois pas l'intérêt, ça marchait aussi bien avant*". Certaines choses vous paraîtront aussi un peu abstraites, vous ne verrez peut-être pas bien comment mettre tout cela en pratique sur votre site. C'est parfaitement normal, lorsqu'on apprend la programmation orientée objet il y a un petit temps d'adaptation pendant lequel on ne voit pas le rapport entre la théorie et la pratique.

Prenez quand même le temps de tout lire et de faire l'effort de comprendre ce qui est expliqué. Petit à petit, en pratiquant, vous découvrirez que cela vous permet de mieux maîtriser votre code PHP au fur et à mesure qu'il grossit, et vous verrez par la suite tout l'intérêt de ce que vous aurez appris. 😊

Ils sont beaux, ils sont frais mes objets

Nous allons commencer par définir ce qu'on appelle un **objet** en programmation. Alors, de quoi s'agit-il ?

 Encore un concept mystique ? Un délire de programmeurs après une soirée trop arrosée ?

Non parce que franchement, un objet c'est quoi ? Mon écran est un objet, ma voiture est un objet, mon téléphone portable... ce sont tous des objets !

Bien vu, c'est un premier point. 😊

En effet, nous sommes entourés d'objets. En fait, tout ce que nous connaissons (ou presque) peut être considéré comme un objet. L'idée de la programmation orientée objet, c'est de manipuler des éléments que l'on appelle des "objets" dans son code source.

 Mais concrètement, c'est quoi ? Une variable ? Une fonction ?

Ni l'un, ni l'autre. C'est un nouvel élément en programmation. Pour être plus précis, un objet c'est... un mélange de plusieurs variables et fonctions. 😊

Ne faites pas cette tête-là, vous allez découvrir tout cela par la suite. 😊

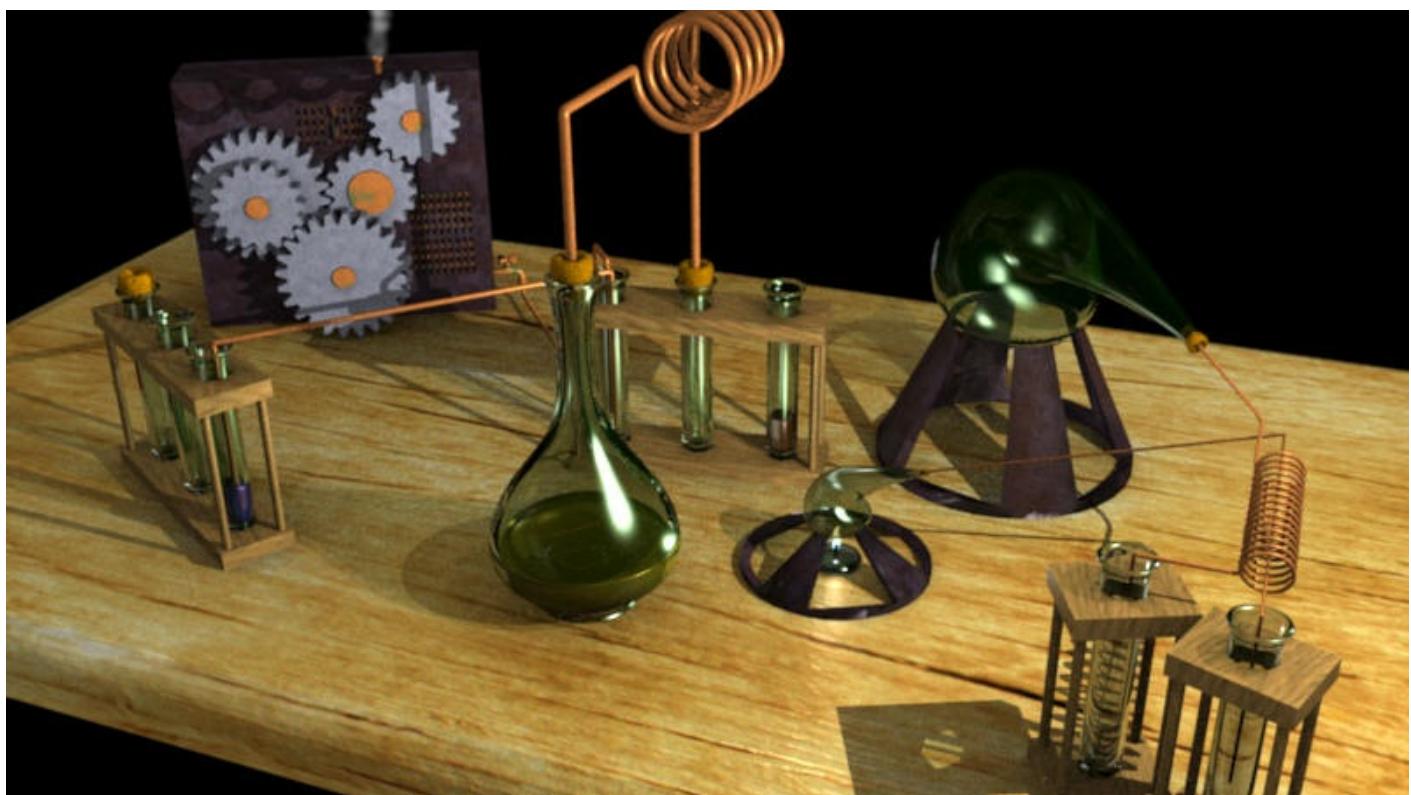
Imaginez... un objet

Pour éviter que ce que je vous raconte ressemble à un traité d'art moderne conceptuel, on va imaginer ensemble ce qu'est un objet à l'aide de plusieurs schémas concrets.

Les schémas 3D que vous allez voir par la suite ont été réalisés pour moi par Nab, que je remercie d'ailleurs vivement au passage.

Imaginez qu'un programmeur décide un jour de créer une section de son site web qui permet d'ajouter des membres, de les placer dans des groupes (administrateur, modérateur...), de les bannir, de les supprimer... Le code est complexe : il a besoin de créer plusieurs fonctions qui s'appellent entre elles, des variables pour mémoriser le nom du membre, sa date d'inscription, son groupe, etc.

Il met du temps à écrire ce code, c'est un peu compliqué, mais il y arrive. Au final, le code qu'il a écrit est composé de plusieurs fonctions et variables. Quand on regarde ça pour la première fois, ça ressemble à une expérience de savant fou à laquelle on ne comprend rien :



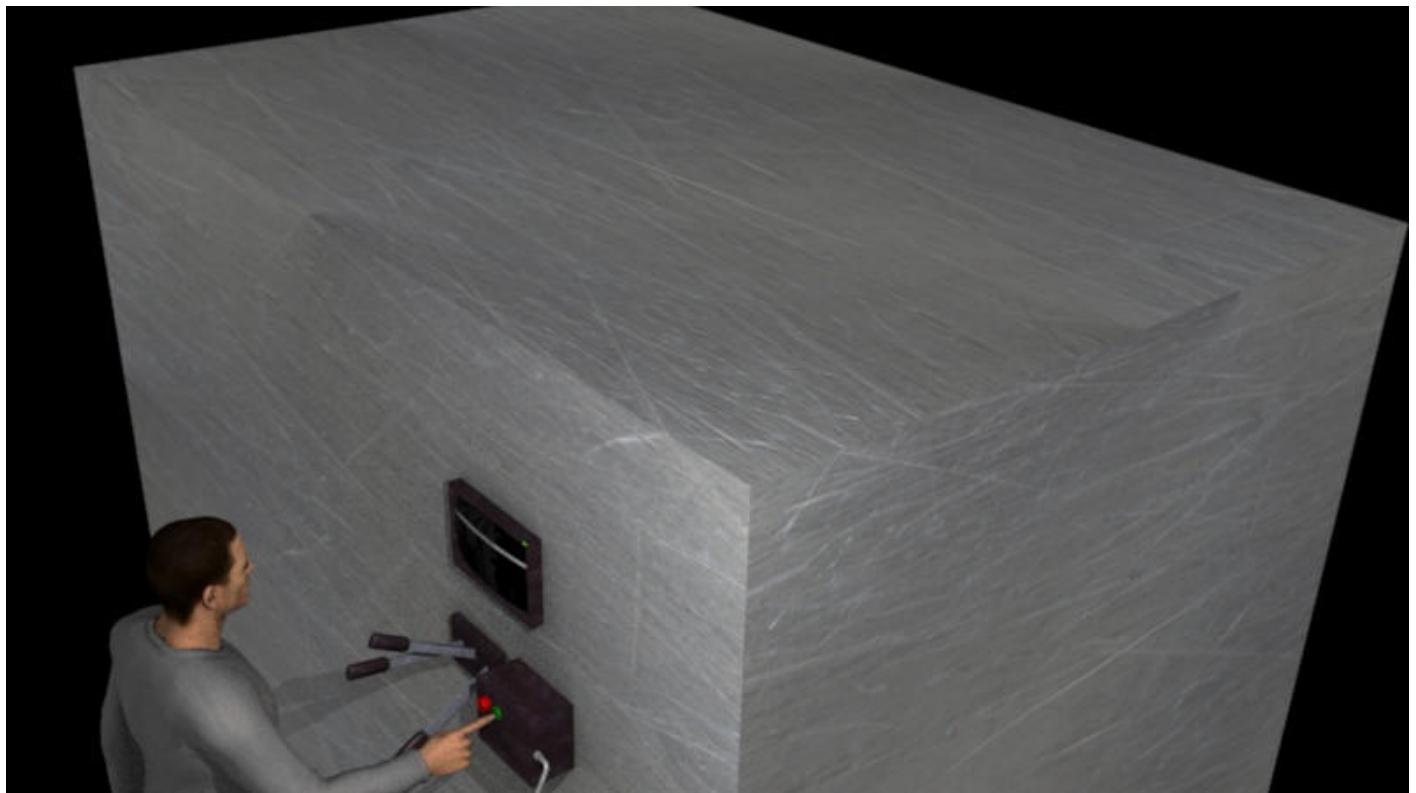
Ce programmeur est content de son code et veut le distribuer sur internet pour que tout le monde puisse créer sa section membres sur son site sans avoir à tout recoder. Seulement voilà, à moins d'être un expert en chimie certifié, vous allez mettre pas mal de temps avant de comprendre comment tout ce bazar fonctionne.

Quelle fonction appeler en premier ? Quelles valeurs envoyer à quelle fonction pour créer le membre ?

C'est là que notre ami programmeur pense à nous. Il conçoit son code de manière orientée objet. Cela signifie qu'il place tout son bazar chimique à l'intérieur d'un simple cube. Ce cube est ce qu'on appelle un objet :



Ici, une partie du cube a été volontairement mise en transparence pour vous montrer que nos fioles chimiques sont bien situées à l'intérieur du cube. Mais en réalité, le cube est complètement opaque, on ne voit rien de ce qu'il y a à l'intérieur :



Ce cube contient toutes les fonctions et les variables (nos fioles de chimie), mais il les masque à l'utilisateur.

Au lieu d'avoir des tonnes de tubes et fioles chimiques dont il faut comprendre le fonctionnement, on nous propose juste quelques boutons sur la face avant du cube : un bouton "créer un membre", un bouton "bannir", etc. L'utilisateur n'a plus qu'à se servir des boutons du cube et n'a plus besoin de se soucier de tout ce qui se passe à l'intérieur. Pour l'utilisateur, c'est donc complètement simplifié.

En clair : programmer de manière orientée objet, c'est *créer* du code source (peut-être complexe), mais que l'on masque en le plaçant à l'intérieur d'un cube (un objet) à travers lequel on ne voit rien. Pour le programmeur qui va l'utiliser, travailler avec un objet est donc beaucoup plus simple qu'avant : il a juste à appuyer sur des boutons et n'a pas besoin d'être diplômé en chimie pour s'en servir.

Bien sûr, c'est une image, mais c'est ce qu'il faut comprendre et retenir pour le moment. 😊

Vous avez déjà utilisé des objets !

Eh oui ! Vous vous souvenez de PDO ? Ne me dites pas que vous avez déjà oublié. 😊

PDO est une extension de PHP et elle est codée en orienté objet. C'est ce qui explique que son mode d'emploi était légèrement différent des fonctions auxquelles nous étions habitués.

Je vous avais promis de vous expliquer plus en détail comment fonctionnait PDO, c'est maintenant l'occasion idéale ! Souvenez-vous de cette ligne qui nous permettait de nous connecter à la base de données :

Code : PHP

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

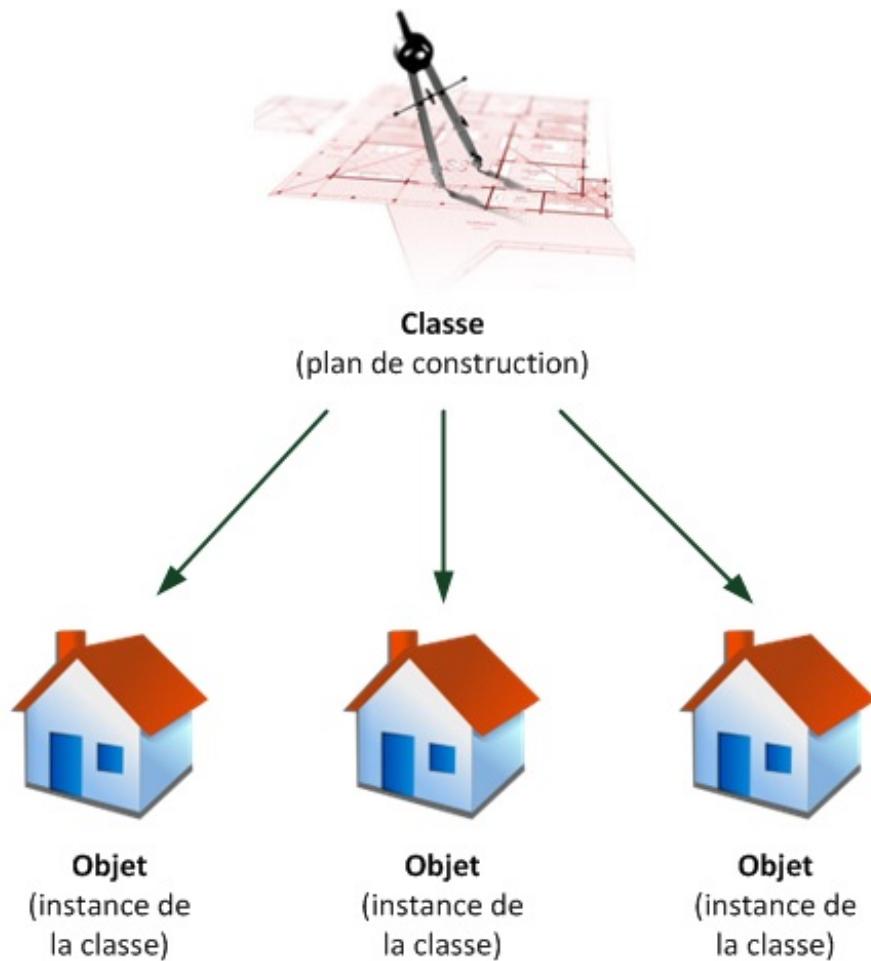
```
$pdo_options);  
?>
```

\$bdd n'est en fait pas une variable mais un objet. On crée un objet à l'aide de la commande `new` suivie du nom de la classe.



Classe ? Objet ? Quelle est la différence ?

- La **classe** est un plan, une description de l'objet. Imaginez qu'il s'agit par exemple des plans de construction d'une maison.
- L'**objet** est une *instance* de la classe, c'est-à-dire une application concrète du plan. Pour reprendre l'exemple précédent, l'objet est la maison. On peut créer plusieurs maisons basées sur un plan de construction. On peut donc créer plusieurs objets à partir d'une classe.



Par conséquent, si on reprend le code précédent, vous aurez deviné que :

- \$bdd est l'objet
- PDO est le nom de la classe sur laquelle est basé l'objet

Un objet est, je vous le disais plus tôt, un mélange de fonctions et de variables. Lorsqu'on l'utilise, on fait appel à ses fonctions :

Code : PHP

```
<?php  
$bdd->query();  
$bdd->prepare();  
$bdd->execute();  
?>
```

Cela signifie : exécuter la fonction `query()` de mon objet `$bdd`, puis la fonction `prepare()`, puis la fonction `execute()`, etc. La flèche `->` est propre aux objets. Il faut donc comprendre que l'on exécute la fonction `query()` de l'objet `$bdd` qui représente la connexion à la base de données.

Autre exemple, imaginaire cette fois, pour être sûr que vous comprenez bien :

Code : PHP

```
<?php  
$maison1 = new Maison();  
$maison2 = new Maison();  
$maison1->nettoyer();  
?>
```

Ici, nous avons plusieurs objets représentant des maisons (`$maison1` et `$maison2`), mais nous n'appelons que la fonction `nettoyer()` de la maison 1, donc c'est la seule qui sera propre. 😊

Un des avantages de la programmation orientée objet comme vous le voyez, c'est sa lisibilité. Ce code est facile à lire et à comprendre.

Nous allons maintenant apprendre dans la suite de ce chapitre à créer nos propres classes.

Créer une classe

Pour nos exemples, nous allons imaginer que nous créons une classe `Membre` qui représente un membre de notre site. Nous pourrons charger ce membre à partir des informations enregistrées en base de données, lui demander son pseudonyme, sa date d'inscription, mais aussi le bannir, le déconnecter du site, etc.

Le code d'une classe étant en général assez long, il est recommandé de créer un fichier PHP qui contiendra uniquement la définition de la classe et que l'on inclura à chaque fois qu'on en a besoin. Je vous recommande de créer un fichier nommé `Membre.class.php`.

 Les développeurs PHP ont l'habitude de donner l'extension `.class.php` à leurs fichiers contenant des classes pour bien les distinguer.

Quelques règles à ce sujet : ne définissez qu'une classe par fichier et donnez au fichier le même nom que votre classe. Le nom de votre classe devrait par ailleurs commencer par une majuscule.

Dans ce fichier `Membre.class.php`, commencez par inscrire le code suivant :

Code : PHP

```
<?php  
class Membre  
{  
}  
?>
```

 Etant donné que notre fichier ne contiendra que du code PHP, il est possible (et même recommandé par des développeurs expérimentés !) de retirer la balise de fermeture `?>` à la fin du fichier. Cela peut paraître surprenant, mais c'est en fait un moyen efficace d'éviter d'insérer des lignes blanches à la fin du code PHP, ce qui a tendance à produire des bugs du type "Headers already sent by".

A l'intérieur des accolades, nous allons définir des variables et des fonctions membres de la classe. Un point vocabulaire à ce sujet : certains développeurs utilisent d'autres mots pour désigner les variables et fonctions membres des classes. Les voici :

- **Variables membres** : aussi appelées attributs ou propriétés
- **Fonctions membres** : aussi appelées méthodes

Essayons maintenant de définir ensemble quelques variables et fonctions dans la classe `Membre`.

Les variables membres

Les variables permettent de définir l'objet, c'est ce qui fait qu'il sera unique. Alors, qu'est-ce qui représente un membre ? Essayons de définir quelques-unes de ses propriétés (si on en oublie ce n'est pas grave, on pourra toujours en rajouter par la suite). Un membre a :

- Un pseudonyme
- Un e-mail
- Une signature
- Un statut (actif ou non, selon que son compte a été validé ou banni)

Nous allons stocker toutes ces informations dans des variables sous forme de texte. Complétez votre classe en rajoutant ces variables :

Code : PHP

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

Nous indiquons que notre classe `Membre` est composée de 4 variables : `$pseudo`, `$email`, `$signature` et `$actif`. Pour l'instant, elles ne possèdent pas de valeur.

Ne vous préoccuez pas pour l'instant du mot-clé `private` devant ces noms de variables, je vous expliquerai un peu plus loin ce que cela signifie. 😊

Les fonctions membres

Maintenant que nous avons défini les variables, nous pouvons créer quelques fonctions. Leur rôle sera :

- Soit de lire ou mettre à jour les variables. On parle de fonctions *getters* et *setters*.
- Soit d'exécuter des actions plus complexes sur le membre (comme lui envoyer un email).

Les getters et setters

Ce sont des fonctions qui commencent par `get` ou par `set`, selon si on veut récupérer le contenu d'une variable ou la modifier.

Prenons par exemple le pseudonyme, on va créer une fonction `getPseudo` qui renvoie le pseudo et `setPseudo` qui modifie le pseudo.

Code : PHP

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

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

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

```
    }
}
```

Nous avons donc 2 fonctions qui permettent de manipuler le pseudonyme du visiteur. Elles sont vraiment très simples comme vous le voyez. Ainsi, `getPseudo` renvoie le pseudo :

Code : PHP

```
<?php
public function getPseudo()
{
    return $this->pseudo;
}
?>
```

La variable `$pseudo` est accessible dans les fonctions avec le préfixe `$this->`. Cela signifie "Le pseudo de cet objet". En effet, souvenez-vous, on peut créer plusieurs objets à partir d'une classe. Il peut y avoir plusieurs membres et chacun d'eux a un pseudo différent. Le préfixe `$this->` permet d'indiquer que c'est bien le pseudonyme du membre sur lequel on travaille que l'on veut récupérer.

On fait de même avec une fonction `setPseudo` qui prend en paramètre le nouveau pseudo du membre et qui le place dans `$this->pseudo`.



Ces fonctions sont très simples et un peu inutiles non ?

En fait, les getters et setters sont souvent des fonctions simples, mais l'intérêt est qu'on peut faire des calculs et des vérifications sur les données. Par exemple, on pourrait améliorer la fonction `setPseudo` comme ceci :

Code : PHP

```
<?php
public function setPseudo($nouveauPseudo)
{
    // Vérifier si le nouveau pseudo n'est pas vide ou trop long
    if (!empty($nouveauPseudo) AND strlen($nouveauPseudo) < 15)
    {
        // Ok on change son pseudo
        $this->pseudo = $nouveauPseudo;
    }
}
?>
```

Ainsi, on autorise le changement de pseudonyme uniquement s'il correspond à certains critères : pseudo non vide et longueur inférieure à 15 caractères. On pourrait profiter de cette fonction pour vérifier aussi la présence de caractères non autorisés.

L'intérêt de passer par une fonction pour modifier les variables est donc de pouvoir contrôler que l'on n'insère pas n'importe quoi. Pour l'e-mail, on pourrait ainsi vérifier que celui-ci a la forme d'un véritable e-mail.

Les autres fonctions

Bien entendu, nous pouvons introduire n'importe quel autre type de fonction dans la classe `Membre`, pas seulement des

fonctions qui se contentent de modifier les variables. A nous de décider quelles actions on veut pouvoir effectuer sur le membre : le bannir, lui envoyer un e-mail...

Code : PHP

```
<?php
class Membre
{
    public function envoyerEMail($titre, $message)
    {
        mail($this->email, $titre, $message);
    }

    public function bannir()
    {
        $this->actif = false;
        $this-
>envoyerEMail('Vous avez été banni', 'Ne revenez plus !');
    }

    ...
}

?>
```

La fonction `envoyerEMail` est toute simple : elle utilise la fonction `mail()` de PHP qui permet d'envoyer un e-mail ainsi que l'adresse e-mail stockée dans l'objet (`$this->email`).

D'autre part, la fonction `bannir()` change le statut `actif` du membre pour indiquer qu'il n'est plus actif et lui envoie un e-mail pour l'avertir de son bannissement. On en profite pour réutiliser la fonction `envoyerEMail()` de notre classe. Vous remarquerez qu'on doit placer là aussi le préfixe `$this->` devant le nom d'une fonction de la classe qu'on appelle.

C'est dans l'esprit de la programmation orientée objet : on réutilise du code déjà écrit pour éviter de réinventer la roue à chaque fois. 😊

Créer un objet à partir de la classe

Actuellement, vous devriez avoir un fichier `Membre.class.php` qui contient la définition de la classe, c'est-à-dire les plans de construction de vos futurs objets :

Code : PHP

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

    public function envoyerEMail($titre, $message)
    {
        mail($this->email, $titre, $message);
    }

    public function bannir()
    {
        $this->actif = false;
        $this->envoyerEMail('Vous avez été banni', 'Ne revenez plus
! ');
    }

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

    public function setPseudo($nouveauPseudo)
    {
        if (!empty($nouveauPseudo) AND strlen($nouveauPseudo) < 15)
        {
            $this->pseudo = $nouveauPseudo;
        }
    }
}
?>
```

Maintenant que la classe est prête (même si on peut encore l'améliorer), on peut commencer à l'utiliser et donc à créer des objets. Créez un nouveau fichier (que vous appellerez comme vous voulez, par exemple `index.php`), dans lequel vous allez créer et utiliser un objet de type `Membre`.

Code : PHP

```
<?php

include_once ('Membre.class.php');

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

?>
```

On commence par inclure la définition de la classe `Membre` située dans le fichier `Membre.class.php`. On utilise la

fonction `include_once()` qui nous assure que le fichier n'a pas déjà été inclus ailleurs dans la page, ce qui aurait provoqué une erreur car il est interdit de définir deux fois une même classe.

On crée ensuite un nouvel objet de type Membre avec la ligne :

Code : PHP

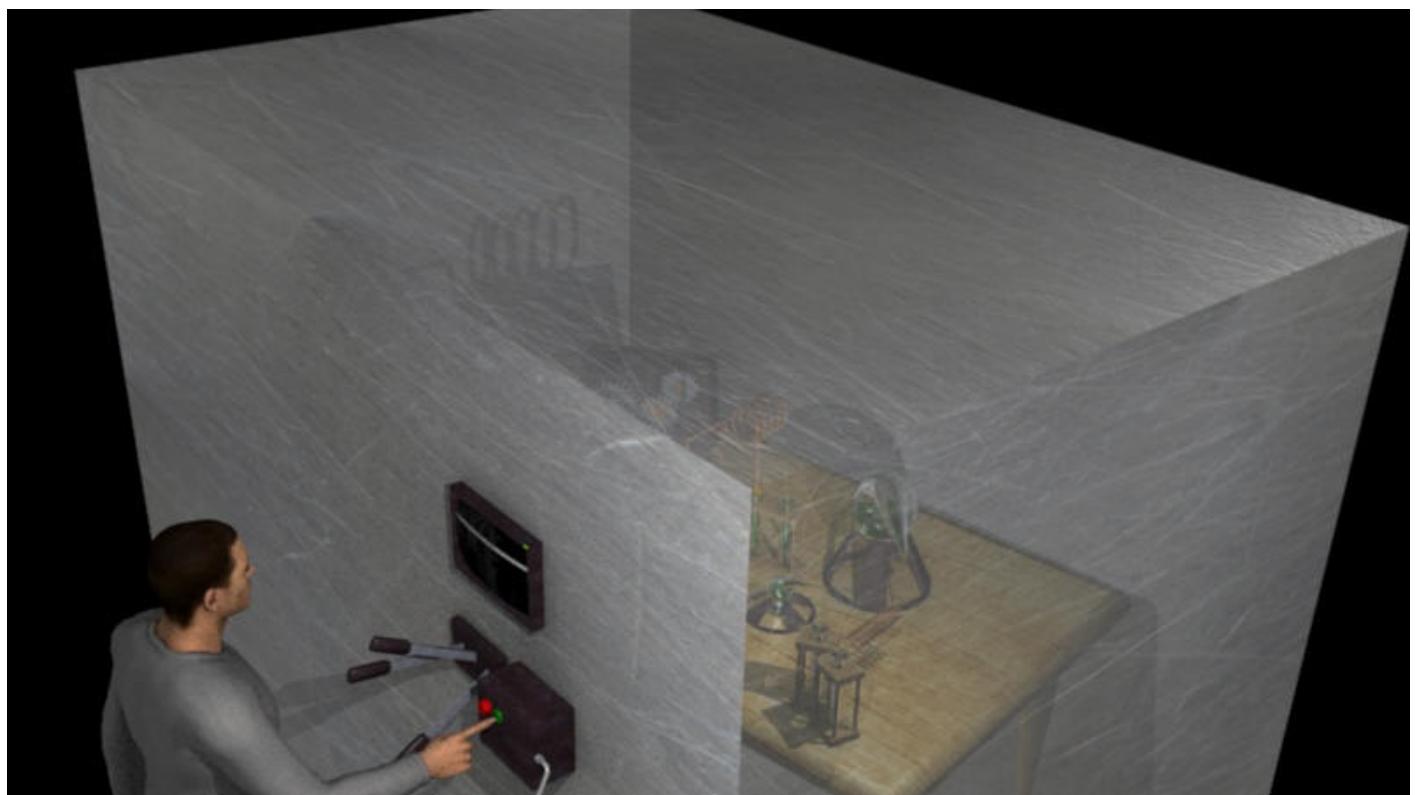
```
<?php  
$membre = new Membre();  
?>
```

Nous avons maintenant un objet `$membre` qui représente un membre vide. On lui définit ensuite un pseudo, on l'affiche, puis pour s'amuser on le bannit. 

 Ne confondez pas la classe et l'objet. La classe commence avec un "M" majuscule, tandis que l'objet avec un "m" minuscule. Nous aurions d'ailleurs pu donner n'importe quel nom à notre objet (`$mateo`, `$nouveauVenu...`), mais faute de trouver de meilleur nom j'ai utilisé le même nom que la classe, avec la première lettre en minuscule pour les différencier.

Vous avez donc là un exemple d'utilisation concrète de la classe. Bien que celle-ci soit basique, vous pourriez déjà donner ce fichier `Membre.class.php` à des amis programmeurs, accompagné de préférence d'une petite documentation qui explique comment l'utiliser, et vos amis pourront comme vous créer et manipuler des membres.

Une classe, c'est donc un peu un package prêt à l'emploi. Elle contient son lot de variables et de fonctions. Pour l'utiliser, nous faisons tout simplement appel à ses fonctions sans nous soucier de ce qu'elles font à l'intérieur. Cela rejoint donc le schéma que je vous avais présenté au début de ce chapitre :



Les variables et le code des fonctions n'intéressent pas le programmeur qui utilise la classe. Il se contente simplement d'appuyer sur des boutons, c'est-à-dire d'appeler les fonctions dont il a besoin pour manipuler son membre : "Donne-lui le pseudo M@teo21", "Bannis-le", etc.

Constructeur, destructeur et autres fonctions spéciales

En plus des fonctions que nous avons créées dans notre classe `Membre`, il existe un certain nombre de fonctions "spéciales" à connaître. On les appelle *fonctions magiques* ou encore *méthodes magiques*.

On les reconnaît facilement car leur nom commence par deux underscores (tiret bas, sous le chiffre 8 d'un clavier AZERTY français). Par exemple : `__construct`, `__destruct`, `__get`, etc.

Nous allons nous intéresser plus particulièrement ici aux deux plus importantes d'entre elles : le constructeur et le destructeur.

Le constructeur : `__construct`

Lorsque vous créez un objet, comme nous l'avons fait précédemment, celui-ci est vide au départ. Ses variables membres ne contiennent rien. Ainsi notre membre n'avait pas de pseudo, pas d'e-mail, rien.

Code : PHP

```
<?php  
$membre = new Membre(); // Le membre est vide  
?>
```

Or, quand vous créez un objet comme ceci avec `new`, il faut savoir que PHP recherche à l'intérieur de la classe une fonction nommée `__construct`. Jusqu'ici nous n'en avions pas créé, donc faute d'en trouver PHP créait un objet vide.

Le rôle d'une fonction constructeur est justement de construire l'objet (non, sans blague !), c'est-à-dire de le préparer à une première utilisation. Dans notre cas, on aimeraient par exemple charger en base de données les informations concernant le membre et insérer les bonnes valeurs dans les variables dès le départ.

Code : PHP

```
<?php  
class Membre  
{  
    public function __construct($idMembre)  
    {  
        // Récupérer en base de données les infos du membre  
        // SELECT pseudo, email, signature, actif FROM membres WHERE  
        id = ...  
  
        // Définir les variables avec les résultats de la base  
        $this->pseudo = $donnees['pseudo'];  
        $this->email = $donnees['email'];  
        // etc.  
    }  
  
    ...  
?>
```

Notre fonction constructeur prend un paramètre : l'id du membre. On peut à partir de là charger en base de données les informations concernant le membre et les insérer dans l'objet : `$this->pseudo`, `$this->email`...

Comme notre constructeur prend un paramètre, il faudra désormais créer nos objets en envoyant un id :

Code : PHP

```
<?php
```

```
$membre = new Membre(32); // Le membre n°32 est chargé !  
?>
```

Notre membre n°32 est maintenant prêt ! 😊

Il contient déjà le bon pseudonyme, le bon e-mail, etc. A vous de faire en sorte ensuite, lorsqu'on modifie son pseudo par exemple, que ce changement soit bien répercuté en base de données.

Le destructeur : `__destruct`

Moins couramment utilisé, le destructeur peut néanmoins se révéler utile. Cette fonction est appelée automatiquement par PHP lorsque l'objet est détruit.



Mais quand l'objet est-il détruit ?

Pour détruire un objet, ou toute autre variable, on peut le faire à la main avec la fonction `unset()` :

Code : PHP

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

Si vous ne le faites pas, l'objet sera détruit à la fin de l'environnement dans lequel il a été créé. Si l'objet a été créé dans la page (comme c'était le cas dans `index.php`), il sera supprimé à la fin de l'exécution de la page.

C'est alors que le destructeur est appelé. Son rôle est de réaliser toutes les opérations nécessaires pour mettre fin à la vie de l'objet.

La classe PDO, par exemple, a besoin d'utiliser le destructeur pour fermer la connexion à la base de données. Elle envoie alors un signal à la base de données "Fin de la connexion".

Dans le cas de notre classe `Membre`, il n'y aurait rien de spécial à faire a priori. Néanmoins, pour tester le fonctionnement du destructeur, vous pouvez en créer un qui affiche un message signalant que l'objet va être détruit :

Code : PHP

```
<?php  
public function __destruct()  
{  
    echo 'Cet objet va être détruit !';  
}  
?>
```

Vous verrez que ce message apparaît à la fin de la page dans notre cas. Si vous avez créé plusieurs objets, vous verrez autant de messages qu'il n'y a d'objets, car chacun d'eux va être détruit !

Les autres fonctions magiques

Il existe de nombreuses autres fonctions magiques dont l'utilité est cependant moins importante : `__get()`, `__set()`, `__call()`, `__sleep()`, `__wakeup()` ... Elles permettent de contrôler certaines autres étapes de la vie de votre objet.

Nous ne les détaillerons pas ici mais si vous voulez en savoir plus sur elles, vous pouvez lire la [page de la documentation qui présente les fonctions magiques](#).

L'héritage

L'héritage est probablement le concept le plus important de la programmation orientée objet. C'est ce qui lui donne toute sa puissance. Cela permet de réutiliser des classes pour construire d'autres classes. On se sert de certaines classes "de base" pour construire des classes plus complètes.

Comment reconnaître un héritage ?

C'est LA question à se poser. Certains ont tellement été traumatisés par l'héritage en cours de programmation qu'ils en voient partout, d'autres au contraire (surtout les débutants) se demandent à chaque fois s'il y a un héritage à faire ou pas. Pourtant, ce n'est pas "mystique", il est très facile de savoir s'il y a une relation d'héritage entre 2 classes.

Comment ? En suivant cette règle très simple :

Il y a héritage quand on peut dire :
"A est un B"

Pas de panique c'est pas des maths. 😊

Prenez un exemple très simple. On peut dire "Un administrateur est un membre", ou encore "Un modérateur est un membre". Donc on peut faire un héritage : "La classe Admin hérite de Membre", "La classe Moderateur hérite de Membre".

Pour vous imprégner, voici quelques autres bons exemples où un héritage peut être fait :

- Une voiture est un véhicule (Voiture hérite de Véhicule)
- Un bus est un véhicule (Bus hérite de véhicule)
- Un moineau est un oiseau (Moineau hérite d'Oiseau)
- Un corbeau est un oiseau (Corbeau hérite d'Oiseau)
- Un chirurgien est un docteur (Chirurgien hérite de Docteur)
- Un diplodocus est un dinosaure (Diplodocus hérite de Dinosaure)
- etc.

En revanche, vous ne pouvez pas dire "Un dinosaure est un diplodocus", ou encore "Un bus est un oiseau". Donc on ne peut pas faire d'héritage dans ces cas-là, du moins ça n'aurait aucun sens. 😊

Réaliser un héritage en PHP

Nous allons créer une nouvelle classe Admin qui sera basée sur la classe Membre. Elle aura toutes les variables et fonctions de la classe Membre, mais elle aura *en plus* de nouvelles variables et fonctions.

Créez un fichier Admin.class.php (souvenez-vous, on fait un fichier par classe !) et insérez-y le code suivant pour commencer :

Code : PHP

```
<?php
include_once ('Membre.class.php');

class Admin extends Membre
{
}

?>
```

Le nouveau mot-clé ici est `extends`, qui signifie "étend". Traduction : la classe Admin étend [les possibilités de] la classe Membre. C'est cela l'héritage : nous avons maintenant une classe Admin qui possède toutes les variables et fonctions de Membre, et nous allons pouvoir en définir de nouvelles qui seront propres aux admins.



Pour que PHP connaisse la classe Membre afin de permettre l'héritage, il est impératif d'inclure le fichier `Membre.class.php` au préalable.

Rajoutons maintenant des fonctionnalités qui seront propres aux admins. Par exemple, ceux-ci, grâce à leurs priviléges, peuvent choisir la couleur dans laquelle sera écrit leur pseudonyme. Ils ont donc une variable `$couleur` et des fonctions qui permettent de la lire et de la modifier :

Code : PHP

```
<?php
include_once ('Membre.class.php');

class Admin extends Membre
{
    private $couleur;

    public function setCouleur()
    {
        // ...
    }

    public function getCouleur()
    {
        // ...
    }
?>
```

Nous avons donc maintenant 2 classes : Membre et Admin.

- Avec Membre, on peut manipuler un pseudo, un e-mail, une signature et un état actif ou non.
- Avec Admin, on peut manipuler les mêmes choses : un pseudo, un e-mail, une signature et un état actif ou non... mais aussi de nouvelles propriétés, comme la couleur du pseudo.



Un peu de vocabulaire ici : on dit que Membre est la **classe mère**, tandis que Admin est la **classe fille**. C'est la fille qui hérite de la mère, imparable logique de programmeur. 😊

Dans notre fichier `index.php` on peut maintenant créer des membres mais aussi des admins :

Code : PHP

```
<?php
$membre = new Membre(31); // Contient un pseudo, un e-mail...
$maitreDesLieux = new Admin(2); // Contient les mêmes données qu'un
                                // membre + la couleur

$membre->setPseudo('Arckintox'); // OK
$maitreDesLieux->setPseudo('M@teo21'); // OK
```

```
$membre->setCouleur('Rouge'); // Impossible (un membre n'a pas de
couleur)
$maitreDesLieux->setCouleur('Rouge'); // OK
?>
```

Avec peu d'efforts, nous avons créé une nouvelle classe qui réutilise une classe existante. On peut donc appeler la fonction `setPseudo` comme pour les membres, et on peut *en plus* effectuer de nouvelles opérations comme définir une couleur de pseudo. 😊

Les droits d'accès et l'encapsulation

Pour terminer notre découverte de la programmation orientée objet, il me reste à vous présenter un concept très important : **l'encapsulation**. En effet, il y a beaucoup de règles en POO et personne ne vous oblige à les suivre. Elles paraissent parfois lourdes et un peu contraignantes. Cela fait que certains finissent par *mal* programmer en POO.

Pourtant, s'il y a une règle à suivre en POO c'est bien l'encapsulation. Avant de vous expliquer ce que cette règle raconte, je dois cependant vous présenter le principe des droits d'accès.

Les droits d'accès

Vous vous souvenez de ces petits mots que nous avons utilisés devant nos noms de variables et fonctions dans ce chapitre ? Oui, je fais référence à `public` et `private`. Ce sont ce qu'on appelle des **droits d'accès** : cela permet d'indiquer si l'utilisateur de la classe a le droit d'accéder directement à un élément de la classe ou non.

Il y a 3 droits d'accès à connaître :

- `public` : tout le monde peut accéder à l'élément.
- `private` : personne (à part la classe elle-même) n'a le droit d'accéder à l'élément.
- `protected` : identique à `private`, mais les classes qui hériteront de celles-ci (les classes filles) auront quand même accès à l'élément.

Essayons de traduire tout ça en mots français voulez-vous ! 😊

Reprendons une version simple de notre classe `Membre` :

Code : PHP

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

    public function getPseudo()
    {

    }

    public function setPseudo($nouveauPseudo)
    {

}
?>
```

Ici, vous voyez que les fonctions sont publiques et les variables sont privées. Rien ne nous empêche cependant de faire l'inverse, même si ce n'est vraiment pas conseillé pour les variables comme nous allons le voir.

Qu'est-ce que cela signifie concrètement ? Que la personne qui utilise la classe à travers des objets (dans `index.php` par exemple) peut uniquement accéder à ce qui est public et pas à ce qui est privé :

Code : PHP

```
<?php  
$membre = new Membre(4);  
$membre->setPseudo('M@teo21'); // OK car setPseudo est public  
$membre->pseudo = 'M@teo21'; // Interdit car $pseudo est private  
?>
```

C'est donc vous qui définissez avec ces `public`, `private` ou `protected` si l'utilisateur a le droit ou non d'appeler la fonction ou la variable.



Et `protected` justement, ça correspond à quoi ?

C'est identique à `private` mais il y a une différence lorsqu'on hérite de la classe. Par exemple, si on définit la variable `$email` comme `protected` dans la classe `Membre`, cela signifie que les classes qui en héritent (comme `Admin`) auront le droit d'y accéder. Sinon, en `private`, elles n'auraient pas pu les appeler directement.

L'encapsulation



Pourquoi est-ce qu'on s'embête avec ça ? Et si on mettait tout en `public` ? On se prendrait moins la tête non ?

Si tout était `public`, même les variables membres, n'importe qui pourrait faire :

Code : PHP

```
<?php  
$membre = new Membre(4);  
$membre->email = 'Portnawak';  
?>
```

Or, "Portnawak" n'est pas un véritable e-mail, je pense que vous serez d'accord avec moi. 🤦

Pour éviter qu'on puisse faire n'importe quoi avec ses objets, on a inventé une règle très simple qui s'appelle la règle d'encapsulation. On peut la résumer comme ceci :

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

Et si vous voulez modifier des variables vous devez passer par des fonctions `setVariable`, na !

Voilà pourquoi on prend l'habitude de créer des fonctions `get` et `set` : cela nous permet de vérifier qu'on ne fait pas n'importe quoi avec les variables, comme définir un e-mail invalide. 🤦

Il arrive en revanche que l'on définisse des fonctions membres privées. Ce sont des fonctions internes à la classe que l'utilisateur n'a pas à appeler directement.



Il est recommandé en général d'utiliser plutôt `protected` que `private`, surtout si vous pensez hériter de la classe. En effet, leur fonctionnement est le même mais `protected` est plus souple si vous avez un jour des classes filles.

Le but de l'encapsulation est de masquer à la personne qui utilise la classe son fonctionnement interne. Quand vous créez un objet, vous ne devez pas avoir à vous soucier des variables qu'il contient, de la façon dont celles-ci sont agencées, etc. Vous êtes juste une personne qui appuie sur des boutons pour effectuer des actions sur l'objet (revoir mes schémas du début du chapitre). Vous devez donc passer par des fonctions pour modifier vos objets.

Pour forcer l'utilisateur à "appuyer sur les boutons" plutôt que de "toucher aux fioles chimiques dangereuses qu'il ne comprend pas", on utilise les droits d'accès. Ainsi, les fonctions sont pour la plupart publiques (sauf celles qui servent au fonctionnement interne de la classe) et les variables sont toujours privées ou protégées pour que l'utilisateur ne puisse pas mettre n'importe quoi à l'intérieur.

Si vous retenez et comprenez ce principe, alors vous serez sur la bonne voie pour bien programmer en orienté objet. 😊

Notre présentation de la programmation orientée objet en PHP s'achève ici. Nous avons fait le tour de la plupart des fonctionnalités de base, mais la POO permet d'aller plus loin. Comme il faudrait un cours entier pour tout découvrir, je vous recommande vivement la lecture du tutoriel de vyk12 spécialement dédié à la [POO en PHP](#). Vous y découvrirez de nouvelles informations sur l'héritage, mais aussi de nouveaux concepts comme les design patterns, les interfaces, les exceptions, etc.

Comme je vous le disais au début, tout ceci doit vous paraître encore pas mal abstrait. Je sais ce que vous ressentez : un mélange de frustration et probablement de fascination pour quelque chose que vous ne comprenez pas encore vraiment. 😞

J'ai fait de mon mieux dans ce chapitre pour être le plus concret possible, mais c'est par la pratique que vous comprendrez le mieux. Un peu de patience. 😊

5) Organiser son code selon l'architecture MVC



[Qu'est-ce que l'architecture MVC ?](#)

[Le code du TP blog et ses défauts](#)

[Amélioration du TP blog en respectant l'architecture MVC](#)

[Aller plus loin : les frameworks MVC](#)

6) TP : créer un espace membres



[Conception de l'espace membres](#)

[Réalisation des pages principales de l'espace membres](#)

[Aller plus loin](#)

Retrouvez ces nouveaux chapitres dans le Livre du Zéro !



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 la lecture du cours, histoire de souffler entre 2 chapitres.

Envoyez votre site sur le web

Votre site est tout beau, tout propre, tout prêt... Mais comme il est sur votre disque dur, personne d'autre ne va pouvoir en profiter !

Vous aimerez donc l'envoyer sur le web, mais... bien sûr vous ne savez pas comment faire 😐

Nous allons découvrir dans cette annexe tout ce qu'il faut savoir pour envoyer son site sur le web. Dans l'ordre :

1. Nous découvrirons comment réserver **un nom de domaine**
 2. Puis nous verrons ce qu'est **un hébergeur** et comment cela fonctionne
 3. Enfin, une fois notre hébergeur choisi, nous verrons comment utiliser **un client FTP** pour enfin pouvoir transférer les fichiers sur le net 😊
-

Le nom de domaine

Savez-vous ce qu'est un **nom de domaine** ?

Il s'agit en fait d'une adresse sur le Web : siteduzero.com est par exemple un nom de domaine.

Un nom de domaine est constitué de 2 parties :

siteduzero.com

- **En rouge, le nom de domaine** proprement dit. Il s'agit d'un nom que l'on peut généralement choisir librement, du tant que personne ne l'a réservé avant nous. Il peut contenir des lettres et des chiffres, mais pas de symboles particuliers (comme le ç français, le é, le è, les espaces, etc).
- **En bleu, l'extension (aussi appelée tld)**. Il existe grossièrement une extension par pays (.fr pour la France, .be pour la Belgique, .ca pour le Canada). Toutefois, il y a aussi des extensions utilisées au niveau international comme .com, .net, .org. Elles étaient au départ réservées aux sites commerciaux, aux organisations, etc... mais cela fait longtemps que tout le monde peut les réserver. D'ailleurs, .com est très probablement l'extension la plus utilisée sur le Web.

 En général, un site web voit son adresse précédée par "www", comme par exemple "www.siteduzero.com". Cela ne fait pas partie du nom de domaine : en fait, "www" est ce qu'on appelle un sous-domaine, et on peut en théorie en créer autant qu'on veut une fois qu'on est propriétaire du nom de domaine 😊

Le "www" a été adopté par tous les webmasters, c'est une sorte de convention, mais elle n'est absolument pas obligatoire.

Réserver un nom de domaine



Moi aussi je veux un nom de domaine pour mon site ! Comment dois-je faire ?

Alors j'ai une bonne et une mauvaise nouvelle 😱

Comme d'hab, on va commencer par la mauvaise :

- **La mauvaise** : ce n'est pas gratuit...
- **La bonne** : ... ce n'est vraiment pas cher du tout 😊

En effet, un nom de domaine coûte entre 7 et 12 € pour un an.

Le prix peut varier en fonction de l'extension. Ainsi, l'extension .info est généralement proposée à plus bas prix et peut s'avérer être une alternative intéressante. Mais si vous voulez une adresse plus "courante", il faudra plutôt viser une extension de type ".com", ou encore ".fr".

Pour réserver un nom de domaine, 2 solutions :

- Passer par un **registrar** spécialisé. C'est un organisme qui sert d'intermédiaire entre l'ICANN (l'organisation qui gère les noms de domaine au niveau international, tel les .com) et vous. **1&1**, **OVH** et **Gandi** sont de célèbres registrars français.
- Encore mieux : vous pouvez commander le nom de domaine en même temps que l'hébergement (c'est ce que je vous conseille). Comme ça vous faites d'une pierre deux coups, vu que vous aurez de toute façon besoin de l'hébergement *et* du nom de domaine.



Pour plus d'info sur l'ICANN, je vous invite à lire [cette page](#) en français de leur site qui détaille un peu plus leur rôle sur le Web. C'est que c'est un sacré boulot de gérer la plupart des noms de domaine du Web !

Dans ce chapitre, nous allons voir comment commander un nom de domaine en même temps que l'hébergement, c'est de loin la solution la plus simple et la moins coûteuse pour vous.

L'hébergeur

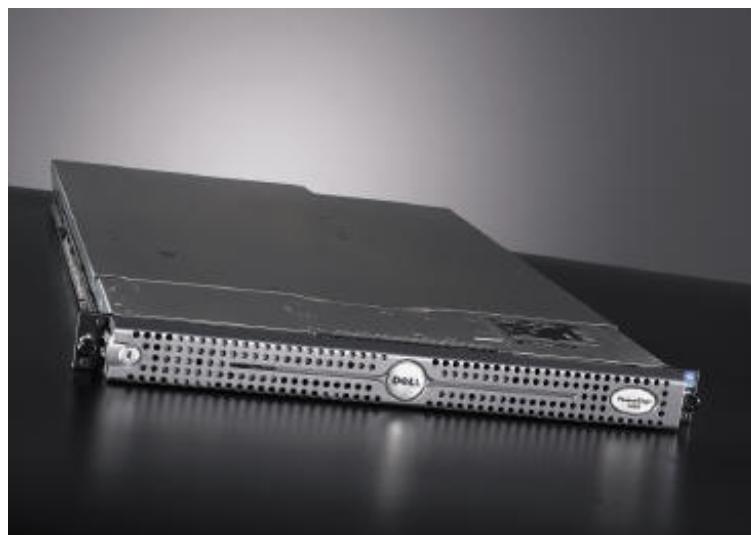
Intéressons-nous maintenant à l'hébergeur.



Qu'est-ce qu'un hébergeur et pourquoi aurais-je besoin de lui ?

Sur Internet, tous les sites web sont stockés sur des ordinateurs particuliers appelés "**Serveurs**". Ce sont des ordinateurs généralement très puissants qui restent tout le temps allumés. Ils contiennent les pages des sites web et les délivrent aux internautes qui les demandent, à toute heure du jour et de la nuit.

Voici à quoi ressemble par exemple un serveur :



Un serveur

Un serveur ne possède pas d'écran car, la plupart du temps, il tourne tout seul sans avoir besoin qu'on fasse quoi que ce soit dessus. Comme vous le voyez, les serveurs sont très plats : c'est un format spécial de serveur (appelé "1U"). Cela permet de les empiler dans des **baies** (une sorte d'armoire climatisée pour serveurs 😊).

Voici à quoi ressemble une baie :



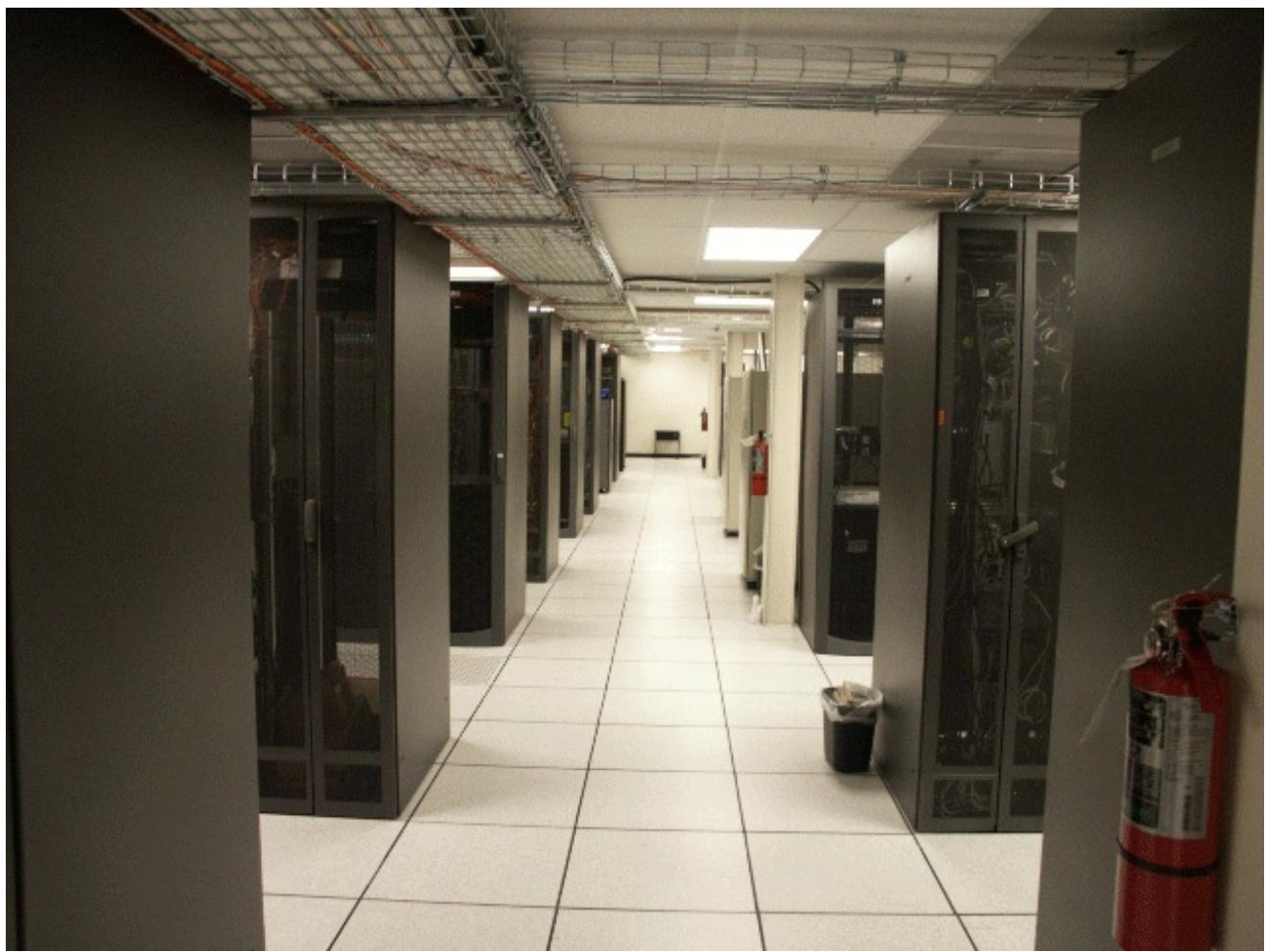
Une baie de serveurs

Comme vous le voyez, il y a un écran pour toute la baie. C'est suffisant car on ne branche l'écran sur un serveur que si celui-ci rencontre un problème. La plupart du temps, heureusement, le serveur travaille sans broncher 😊

Le rôle de l'hébergeur

L'hébergeur est une entreprise qui se charge de gérer des baies de serveurs. Elle s'assure du bon fonctionnement des serveurs 24h/24 7j/7. En effet, si l'un d'eux tombe en panne, tous les sites présents sur la machine deviennent inaccessibles (et ça fait des clients mécontents 😞).

Ces baies se situent dans des lieux particuliers appelés **datacenters**. Les datacenters sont donc des "entrepôts à serveurs" en quelque sorte, et leur accès est très protégé.



Un datacenter. On voit ici plusieurs baies de serveurs.

Il est aussi possible en théorie d'héberger un site sur son propre ordinateur. Toutefois, c'est complexe, il vaut mieux avoir des connaissances en Linux, l'ordinateur doit être assez puissant, tourner jour et nuit et... surtout... la connexion doit être à très très haut débit (surtout en *upload*, la vitesse d'envoi des fichiers compte énormément). Les particuliers n'ont en règle générale pas une connexion suffisamment puissante pour héberger des sites, tandis que les datacenters oui : ils sont câblés en fibre optique (ça peut aller à une vitesse de plusieurs Gbps ! 😊)

Bref, gérer un serveur soi-même est complexe, et la plupart du temps les particuliers et les entreprises font appel à un hébergeur dont c'est le métier.

Trouver un hébergeur

Les hébergeurs, contrairement aux registrars, sont très très nombreux. Il y en a de tous types, à tous les prix. Il y a un vocabulaire à connaître pour vous repérer dans leurs offres :

- **Hébergement mutualisé** : si vous optez pour une offre d'hébergement mutualisée, votre site sera placé sur un serveur gérant plusieurs sites à la fois (peut-être une centaine, peut-être plus). C'est l'offre la moins chère et c'est celle que je vous recommande de viser si vous démarrez votre site web.
- **Hébergement dédié virtuel** : cette fois, le serveur ne gère que très peu de sites (généralement moins d'une dizaine). Cette offre est généralement adaptée aux sites qui ne peuvent plus tenir sur un hébergement mutualisé car ils ont trop de trafic (trop de visiteurs), mais qui ne peuvent pas se payer un hébergement dédié (voir ci-dessous).
- **Hébergement dédié** (on parle aussi de "serveur dédié") : c'est le nec plus ultra. Le serveur gère uniquement votre site et aucun autre. Attention, cela coûte assez cher et il vaut mieux avoir des connaissances en Linux pour administrer le serveur à distance.
Par exemple, le Site du Zéro est lui-même sur un hébergement dédié, car son trafic est très important.

Dans le cas d'un hébergement mutualisé, il faut veiller à ce que l'hébergeur supporte PHP et MySQL. C'est presque toujours le cas, mais il vaut quand même mieux vérifier que celui-ci sera capable de faire tourner votre site en PHP !



Mais où puis-je trouver un hébergeur ?

Oh ça c'est très simple 😊

Une recherche dans Google de "hébergeur web" vous donnera plusieurs millions de résultats. Vous n'aurez que l'embarras du choix.



Si je puis me permettre un conseil, je vous recommande de jeter un œil à l'hébergeur **PlanetHoster** qui propose des services d'hébergement de qualité. Ils font d'ailleurs **des réductions pour tous les visiteurs du Site du Zéro !** 😊

Si les offres de PlanetHoster ne vous conviennent pas, vous pouvez regarder chez **l'hébergeur 1&1** (un concurrent 😊) ou encore **MavenHosting**, qui proposent d'autres offres pour les particuliers et entreprises.



La suite de ce chapitre vous détaille la procédure pour héberger votre site chez PlanetHoster, mais sachez que cela fonctionne quasiment de la même manière avec 1&1 et MavenHosting.

Revenons à **PlanetHoster**. L'hébergeur propose plusieurs offres d'**hébergement mutualisé** :

Plan Essentiel	Plan Performance	Plan Illimité
€2.99 /mois	€5.99 /mois	€10.99 /mois
<ul style="list-style-type: none"> ✓ 10GB Espace disque ✓ 250GB Trafic ✓ Aucun Frais D'installation ✓ Activation Instantanée ✓ Nom de domaine GRATUIT COMMANDER PLUS DE DÉTAILS	<ul style="list-style-type: none"> ✓ 50GB Espace disque ✓ Illimité Trafic ✓ Aucun Frais D'installation ✓ Activation Instantanée ✓ Nom de domaine GRATUIT COMMANDER PLUS DE DÉTAILS	<ul style="list-style-type: none"> ✓ Illimité Espace disque ✓ Illimité Trafic ✓ Aucun Frais D'installation ✓ Activation Instantanée ✓ Nom de domaine GRATUIT COMMANDER PLUS DE DÉTAILS

PlanetHoster fait des **réductions** spéciales pour les visiteurs du Site du Zéro sur tous ces plans via un code

promotionnel :



- 5% de remise pour le plan essentiel
- 15% de remise pour les plans performance et illimité

Ces remises sont valables si vous rentrez un code promotionnel (j'en reparle un peu plus bas) pour une commande annuelle de l'un de ces plans.

- **Plan essentiel** : 10 Go d'espace disque et 250 Go de trafic.
- **Plan performance** : 50 Go d'espace disque et trafic illimité.
- **Plan illimité** : espace disque et trafic illimités.

Ces offres sont en fait très similaires, elles diffèrent seulement au niveau de l'espace de stockage et du trafic.



Mais qu'est-ce qu'ils appellent le "trafic" ? 😊

Le trafic, c'est la quantité de données envoyées par mois aux visiteurs de votre site. Par exemple, si vous avez une image de 1 Mo sur votre site et qu'elle est chargée 500 fois dans le mois par vos visiteurs, alors vous créerez un trafic de 500 Mo. En pratique, il faut savoir que les navigateurs des visiteurs mettent en cache les images, ce qui leur évite d'avoir à recharger plusieurs fois une même image. Cela diminue d'autant plus le trafic nécessaire.

Si vous avez beaucoup de visiteurs (donc beaucoup de trafic), il faudra choisir un plan qui autorise plus de trafic.

Commander un hébergement pour votre site web

Après avoir cliqué sur n'importe quel bouton "Commander", nous arrivons sur la page suivante :

The screenshot shows the "Formulaire de commande PlanetHoster".

Choisissez un produit:

- Mutualisé
- Revendeur
- Dédié
- VPS
- E-Commerce
- Plan Essentiel
- Plan Performance
- Plan Illimité

RÉSUMÉ:

Mutualisé Plan Essentiel	€36.00 EUR
Sous-total: €36.00 EUR	
Total à payer aujourd'hui: €36.00 EUR	
Entrez le code de la prom	GO
Montant récupéré: €36.00 EUR Annuel	

Nom de domaine:

Merci de saisir le nom de domaine que vous souhaitez utiliser ci-dessous.

Nous avons ici trois informations :

- **Choisissez un produit** : indique quel plan vous avez choisi. Vous pouvez en changer directement via cet encart ;
- **Résumé** : ce cadre, comme son nom l'indique, résume votre commande avec le plan choisi ainsi que le prix à payer ;
- **Nom de domaine** : cette partie vous permet de choisir le nom de domaine de votre site web. Nous allons y venir.

Le champ de texte se trouvant dans le cadre **Résumé** vous permet de rentrer un code promotionnel :

- **SiteDuZero-Perso** : si vous commandez un plan essentiel (à 2,99€ / mois)
- **SiteDuZero** : si vous commandez n'importe quel autre plan

Par exemple pour le code SiteDuZero-Perso :

The screenshot shows a payment summary page with a green header bar labeled "RÉSUMÉ". Below it, a section for a "Mutualisé Plan Essentiel" is shown at a price of "€36.00 EUR". A yellow box highlights a discount: "Sous-total: €36.00 EUR" and "Total à payer aujourd'hui: €36.00 EUR". At the bottom, there's a text input field containing "SiteDuZero-Perso" and a blue "GO" button. A note below says "Montant récursif: €36.00 EUR Annuel".

Validez, et vous aurez alors une réduction sur le montant total du pack d'hébergement ! 😊

Merci qui ?

Le champ de texte se trouvant sous **Nom de domaine** vous invite à saisir... votre nom de domaine. Le site de [PlanetHoster](#) va alors se charger de vérifier si le domaine est disponible. Si c'est bon, vous pouvez passer à la suite. 😊
Sinon, il faudra choisir un autre nom de domaine car quand le domaine est déjà pris vous ne pouvez pas faire grand chose. 😞

Ensuite, le site vous demande si vous désirez qu'il enregistre ce domaine ou si vous désirez l'enregistrer séparément.

The screenshot shows a search interface for a domain name. A blue bar at the top has a globe icon and the text "Nom de domaine". Below it, a message says "Merci de saisir le nom de domaine que vous souhaitez utiliser ci-dessous.". A search bar contains "www. adresse demonsite.fr". A green success message "Félicitations, ce domaine est disponible !" is displayed. Two radio buttons are shown: one selected for "Merci d'enregistrer ce domaine pour 1 An/s @ €8.99 EUR" and another for "Je vais enregistrer ce domaine moi-même séparément".



Obtenir un "vrai" nom de domaine (.fr, .com, .net, .org...) est habituellement payant chez les hébergeurs. Néanmoins, si vous achetez un hébergement d'un an chez PlanetHoster, le nom de domaine est offert (il est compris dans l'hébergement).

Il ne vous reste plus qu'à renseigner vos coordonnées et finaliser l'achat par carte bancaire ou Paypal. Une fois les formalités et le paiement effectués, vous êtes redirigé vers PlanetHoster, qui vous confirme la bonne prise en compte de votre commande.

Vous devriez recevoir un peu plus tard un e-mail vous indiquant toutes les informations nécessaires pour mettre en place votre site. Conservez-les précieusement, vous en aurez besoin.

Lorsque vous aurez reçu par email vos identifiants pour vous connecter au serveur de votre hébergeur, vous pouvez passer à l'étape suivante : **envoyer votre site web sur le serveur de votre hébergeur !**

Utiliser un client FTP

Installer un client FTP

FTP signifie *File Transfer Protocol* et, pour faire court et simple, c'est le moyen que l'on utilise pour envoyer nos fichiers. Il existe des logiciels permettant d'utiliser le FTP pour transférer vos fichiers sur Internet.

Bien entendu, des logiciels FTP il en existe des centaines, gratuits, payants, français, anglais etc...

Pour que nous soyons sur la même longueur d'ondes, je vais vous proposer celui que j'utilise qui est gratuit et en français : **FileZilla**.



Ce logiciel n'a rien à avoir avec Mozilla, si ce n'est qu'il se termine lui aussi par "zilla". N'allez donc pas croire que je vous force à utiliser des logiciels d'un même éditeur, c'est tout à fait faux
D'ailleurs, vous pouvez utiliser n'importe quel autre logiciel FTP si ça vous chante, ça ne me dérange absolument pas.

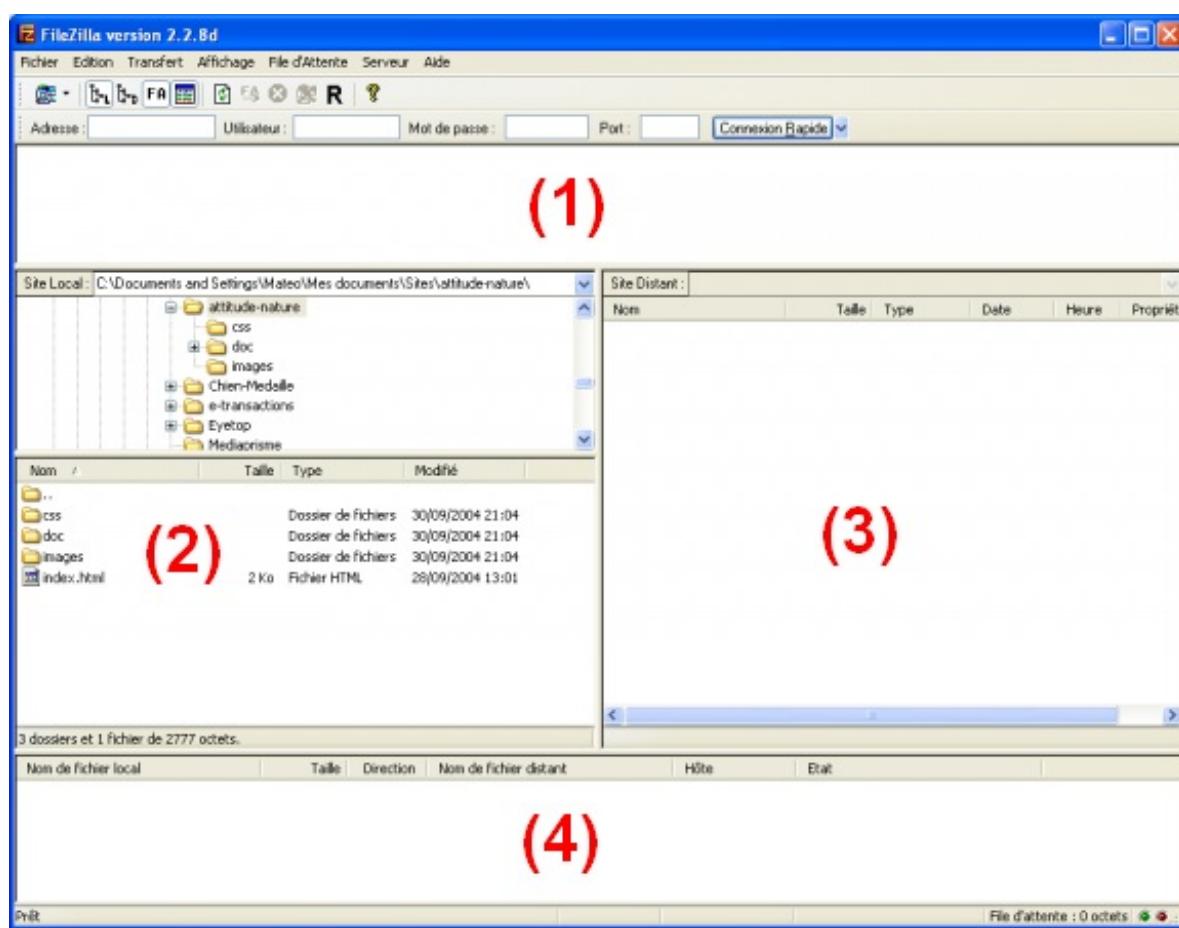
Quoiqu'il en soit, je vais vous montrer quelle est la marche à suivre avec FileZilla.

Première étape : ... le télécharger bien entendu 😊

Télécharger FileZilla (3,9 Mo)

Je vous fais confiance pour l'installation, elle est toute simple et vous ne devriez pas avoir de problème 😊

Lancez le logiciel, et vous devriez voir ceci :



A première vue, ça a l'air un peu compliqué (à première vue seulement). En fait, le principe est très simple. Il y a 4 grandes zones dans la fenêtre à connaître :

1. En haut, vous verrez apparaître les messages qu'envoie et reçoit le logiciel. Si vous avez un peu de chance, vous verrez même la machine vous dire Bonjour" (si si je vous jure ;)) En général, cette zone ne nous intéresse pas vraiment, sauf s'il y a des messages d'erreur... et comme ils sont écrits en rouge, vous devriez pas les louper 😊

2. A gauche, c'est votre disque dur. Dans la partie du haut vous avez les dossiers, et dans la partie du bas la liste des fichiers du dossier actuel.
3. A droite, c'est la liste des fichiers envoyés sur Internet. Pour le moment il n'y a rien, car on ne s'est pas "connecté", mais ça va venir ne vous en faites pas.
4. Enfin, en bas, vous verrez apparaître les fichiers en cours d'envoi (et le pourcentage d'envoi).

La première étape va être de se "connecter" au serveur de votre hébergeur.

Configurer le client FTP

Quel que soit l'hébergeur que vous avez choisi, cela fonctionne toujours de la même manière. On va vous fournir **3 informations** qui sont indispensables pour que FileZilla puisse se connecter au serveur :

- **L'IP** : c'est "l'adresse" du serveur. Le plus souvent, on vous donnera quelque chose du genre *ftp.mon-site.com*, mais il peut aussi s'agir d'une suite de nombres comme *122.65.203.27*
- **Le login** : c'est votre identifiant, on vous l'a probablement demandé. Vous avez peut-être mis votre pseudo, ou le nom de votre site. Mon login pourrait par exemple être *mateo21*.
- **Le mot de passe** : soit on vous a demandé un mot de passe, soit (c'est plus probable) on vous en a attribué un d'office (un truc imprononçable du genre *crf45u7h*)

Si vous avez ces 3 informations, vous êtes le roi du monde 😊 (du moins, vous allez pouvoir continuer ce tutoriel 😊)

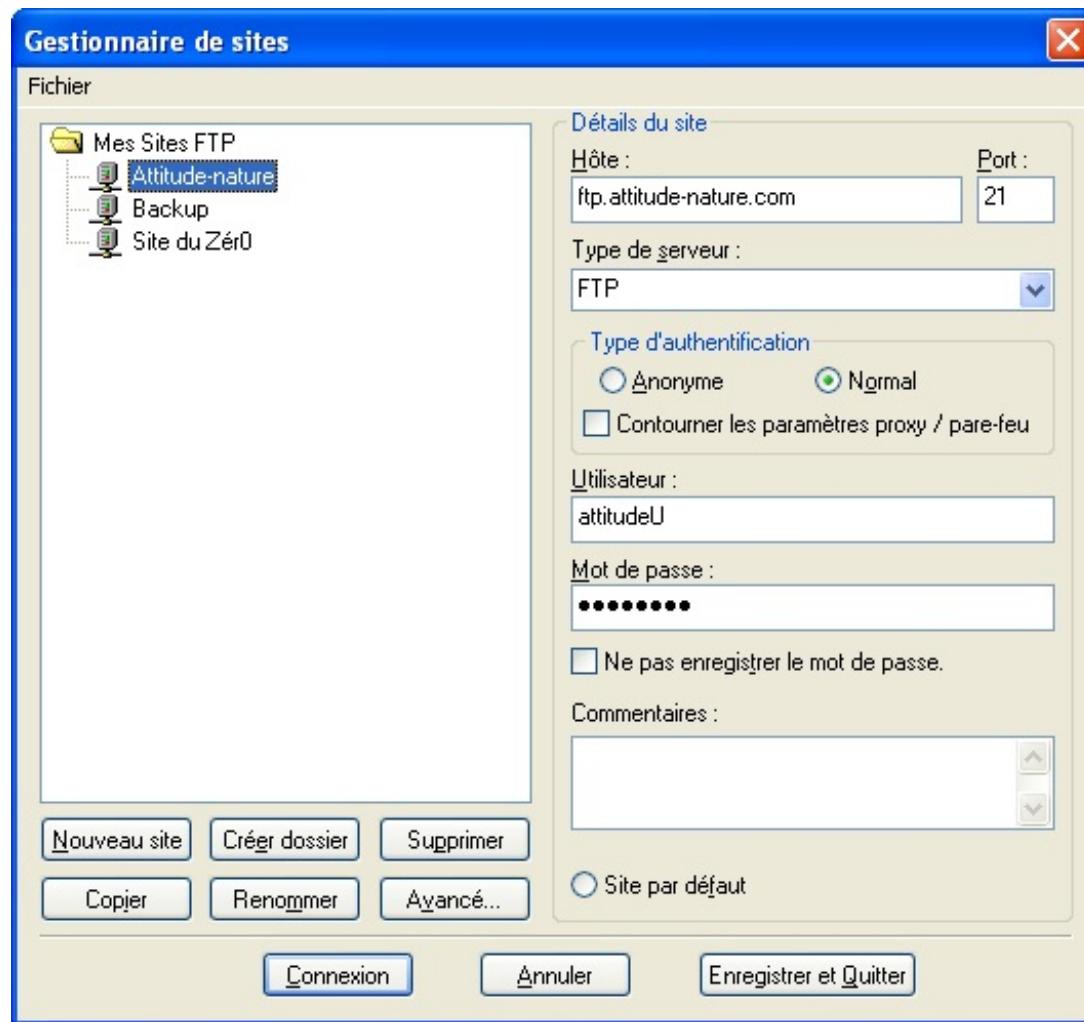
Si vous ne les avez pas, il faut que vous les cherchiez c'est indispensable. On vous les a probablement envoyées par mail. Sinon, n'hésitez pas à les demander à votre hébergeur (IP / Login / Mot de passe).

Maintenant que nous sommes en possession de ces informations, nous allons les donner à FileZilla qui en a besoin pour se connecter au serveur.

Cliquez sur la petite icône en haut à gauche (pas sur la petite flèche à droite, mais sur l'image) :



Une fenêtre s'ouvre. Cliquez sur "Nouveau site" et donnez-lui le nom que vous voulez (par exemple *Site du Zér0*). A droite, vous allez devoir indiquer les 3 informations dont je viens de vous parler, comme ceci :



Vous pouvez distinguer en haut l'hôte (c'est là qu'il faut indiquer l'adresse ip, du genre [ftp.attitude-nature.com](ftp://ftp.attitude-nature.com)).

Pour pouvoir entrer le login / mot de passe, cochez **Type d'authentification : Normal**

Ici le login est *attitudeU* et le mot de passe est... caché (vous croyiez quand même pas que j'allais vous le donner ?! :p)

Cliquez sur "Connexion" et le tour est (presque) joué.

Transférer les fichiers

A ce stade, 2 possibilités :

- Soit la connexion a réussi, et vous voyez apparaître en haut des messages en vert comme "Connecté". Dans ce cas, la zone de droite de la fenêtre de FileZilla devrait s'activer et vous verrez les fichiers qui se trouvent déjà sur le serveur (il se peut qu'il y en ait quelques-uns déjà présents).
- Soit ça a planté, vous avez plein de messages écrits en rouge et là bah... Il n'y a pas 36 solutions : vous vous êtes plantés en tapant l'IP ou le login ou le mot de passe. Un de ces éléments est incorrect, veillez à les redemander à votre hébergeur car s'ils sont bons ça doit marcher.

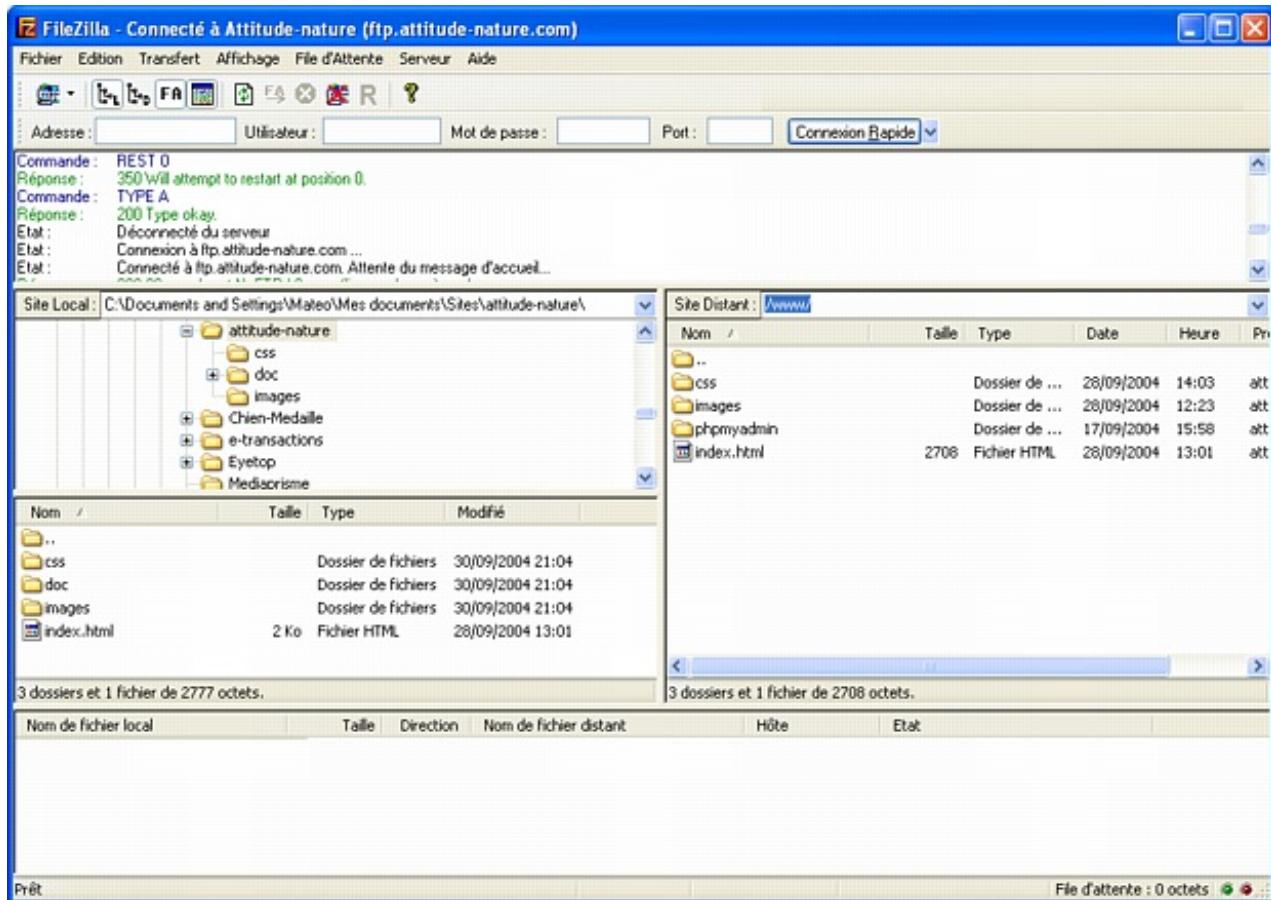
Si la connexion a réussi, alors ce que vous avez à faire est très simple : dans la partie de gauche, cherchez où se trouvent sur votre disque dur vos fichier .html et .css (mais aussi vos images .jpg, .png, .gif etc....).

Double-cliquez sur le fichier que vous voulez transférer à gauche. Au bout de quelques secondes, il apparaîtra à droite, ce qui voudra dire qu'il a été correctement envoyé sur le serveur, donc qu'il est accessible sur Internet !!! 😊



Vous pouvez envoyer n'importe quel type de fichier. Bien entendu, généralement on envoie des fichiers .html, .css et des images, mais vous pouvez aussi très bien envoyer des .pdf, des programmes, des zip etc etc...

Voici par exemple ce que ça donne quand on a transféré un fichier "index.html" :



Il apparaît à droite, ce qui veut dire qu'il est maintenant disponible sur le serveur 😊

Veuillez noter qu'il faut que votre page d'accueil s'appelle "index.php". C'est la page qui sera chargée lorsqu'un nouveau visiteur arrivera sur votre site.

Vous pouvez aussi transférer des dossiers entiers d'un seul coup : il suffit de faire glisser-déplacer le dossier depuis la partie de gauche jusqu'à la partie de droite de la fenêtre.

Allez, avouez qu'une fois configuré, c'est pas bien compliqué 😊

Accéder à la base de données de l'hébergeur

L'envoi des fichiers .php se fait via un logiciel FTP comme nous venons de le voir. Ceux-ci fonctionneront sans aucun problème si vous avez choisi un hébergeur qui supporte PHP (et je vous rassure, c'est le cas de la très grande majorité d'entre eux 😊).

Il vous suffit, après envoi du fichier .php, de vous rendre par exemple sur : www.monsite.com/mapage.php (la page de votre site en ligne) et celle-ci s'exécutera comme sur votre PC. Les serveurs de l'hébergeur font eux aussi tourner Apache et PHP, comme vous sur votre machine avec WAMP.



Mais comment ça fonctionne pour la base de données ? Comment je peux envoyer ma base de données sur Internet ?

Bonne question. En fait, votre hébergeur vous propose déjà une base de données MySQL. Celle-ci est le plus souvent déjà créée.

Normalement, votre hébergeur doit vous donner le moyen d'y accéder (par e-mail ou sur leur panel web).

Vous avez besoin de ces informations :

- L'adresse IP du serveur MySQL
- Votre login MySQL
- Votre mot de passe MySQL
- Le nom de la base de données, si elle a déjà été créée
- L'adresse du phpMyAdmin qui vous permet de gérer votre base en ligne



Généralement, les hébergeurs ne vous proposent qu'une ou deux bases de données. Ce n'est pas vraiment une limitation car une base de données suffit à stocker toutes les informations dont on a besoin. Avoir plusieurs bases permet juste de mieux découper les informations, un peu comme les dossiers sur votre ordinateur.

Servez-vous des 4 premières informations (IP, login, mot de passe, nom de la base) pour adapter votre code PHP afin qu'il puisse se connecter à la base de données de l'hébergeur :

Code : PHP

```
<?php  
$bdd = new PDO('mysql:host=mysql.hebergeur.com;dbname=mabase',  
'pierre.durand', 's3cr3t');  
?>
```

Maintenant que c'est fait, vos scripts ont accès à la base de données de l'hébergeur... Mais celle-ci est encore vide ! Il faut utiliser le phpMyAdmin qu'ils mettent à votre disposition pour y recréer les tables.

Pour cela, vous devez suivre ces étapes :

1. Sur votre machine, via WAMP, accédez à votre phpMyAdmin local. Utilisez-le pour exporter toutes vos tables, [comme nous avons appris à le faire](#). Cela va créer un fichier .sql sur votre disque dur qui contiendra vos tables.

Afficher le schéma de la table

Exporter <ul style="list-style-type: none"> <input type="radio"/> CodeGen <input type="radio"/> CSV <input type="radio"/> CSV pour MS Excel <input type="radio"/> Microsoft Excel 2000 <input type="radio"/> Microsoft Word 2000 <input type="radio"/> LaTeX <input type="radio"/> Tableur "Open Document" <input type="radio"/> Texte "Open Document" <input type="radio"/> PDF <input checked="" type="radio"/> SQL <input type="radio"/> Texte Texy! <input type="radio"/> XML <input type="radio"/> YAML 	Options <ul style="list-style-type: none"> Commentaires mis en en-tête (\n sépare les lignes) <input type="text"/> <input checked="" type="checkbox"/> Commentaires <input type="checkbox"/> Utiliser le mode transactionnel <input type="checkbox"/> Désactiver la vérification des clés étrangères Mode de compatibilité SQL <input type="button" value="NONE"/> <input type="checkbox"/> <input checked="" type="checkbox"/> Structure <ul style="list-style-type: none"> <input type="checkbox"/> Ajouter DROP TABLE <input checked="" type="checkbox"/> Ajouter IF NOT EXISTS <input checked="" type="checkbox"/> Inclure la valeur courante de l'AUTO_INCREMENT <input checked="" type="checkbox"/> Protéger les noms des tables et des champs par des ` <input type="checkbox"/> Ajouter CREATE PROCEDURE / FUNCTION / EVENT Inclure sous forme de commentaires <ul style="list-style-type: none"> <input type="checkbox"/> Dates de création/modification/vérification <input checked="" type="checkbox"/> Données <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Insertions complètes <input type="checkbox"/> Insertions étendues <input type="checkbox"/> Taille maximum de la requête générée <input type="text" value="50000"/> <input type="checkbox"/> Insertions avec délais (DELAYED) <input type="checkbox"/> Ignorer les erreurs de doublons (INSERT IGNORE) <input checked="" type="checkbox"/> Utiliser l'hexadecimal pour les BLOB Type d'exportation <input type="button" value="INSERT"/>
Exporte <input type="text" value="3"/> enregistrement(s) à partir du rang n° <input type="text" value="0"/>	
<input type="checkbox"/> Transmettre	
Modèle de nom de fichier ¹ : <input type="text" value="TABLE_"/> (<input checked="" type="checkbox"/> se souvenir du modèle)	
Compression: <input checked="" type="radio"/> aucune <input type="radio"/> "zippé" <input type="radio"/> "gzippé"	
<input type="button" value="Exécuter"/>	

2. Rendez-vous ensuite sur l'adresse du phpMyAdmin *de votre hébergeur*. Une fois là-bas, utilisez la fonctionnalité d'importation pour y importer le fichier .sql qui se trouve sur votre disque dur.

Fichier à importer

Emplacement du fichier texte (Taille maximum: 2 048 Kio)

Jeu de caractères du fichier: utf8

Ces modes de compression seront détectés automatiquement : aucune, gzip, zip

Importation partielle

Permettre l'interruption de l'importation si la limite de temps est sur le point d'être atteinte. Ceci pourrait aider à importer des fichiers volumineux, au détriment du respect des transactions.

Nombre d'enregistrements (requêtes) à ignorer à partir du début

Format du fichier d'importation

CSV

CSV via LOAD DATA

SQL

Options

Mode de compatibilité SQL

3. Patinez un peu... Et vos tables sont maintenant chargées sur le serveur MySQL de l'hébergeur ! 😊

C'est à peu près tout ce que vous avez besoin de savoir.

Bien entendu, vous ne pouvez pas deviner tout seul tout ceci. C'est d'ailleurs pour cela que j'ai rédigé cette annexe, car bon nombre de débutants sont perdus et ne comprennent pas l'intérêt des registrars, hébergeurs, serveurs et compagnie 😊

Allez, au boulot, vous avez des fichiers à transférer je crois 😊

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 font l'effort 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 sait qu'un code qui marche n'est pas forcément bon.

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 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 tutoriel PHP, et cette fois j'y reviens avec un peu plus d'explications. Quand vous créez un script PHP, vous devez *inventer* des noms. Vous allez devoir donner des noms à différents types d'éléments :

- Les variables
- Les fonctions
- Les classes

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
- \$val
- \$val2

Je n'ai pas inventé ces noms de variables, en fait pour tout vous dire ce sont des noms que j'ai vraiment vus dans de nombreux codes source.

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 😊). 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 des forums sur internet, vous verrez que si vous utilisez des noms peu 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 de variables peu clairs

Code : PHP

```
<?php
$mess_page = 20;

$ret = $bdd->query('SELECT COUNT(*) AS nb FROM livre');

$data = $ret->fetch();
$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 de variables beaucoup plus clairs

Code : PHP

```
<?php
$nombreDeMessagesParPage = 20;

$retour = $bdd->query('SELECT COUNT(*) AS nb_messages FROM livre');
$donnees = $retour->fetch();
$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. 😊

Indentez votre code

Une des premières choses qui saute aux yeux quand on regarde un code source, c'est son **indentation**.

Le principe de l'indentation, 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 utilisez du code indenté, et vous aident donc beaucoup 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 ferez bien d'essayer un vrai logiciel fait pour ça, comme Notepad++ dont je vous ai parlé dans le deuxième 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 !

Il y a plusieurs "styles" d'indentation de code, cela varie un peu selon les goûts des développeurs. Celui que je vous propose est simple à retenir :

- A chaque fois que vous ouvrez des accolades {, par exemple pour `if`, `un while` ou `un for`, 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.

Avec un code non indenté

C'est plus clair avec un exemple, alors voyez-vous même. Voici ce que ça donne avec un code non indenté :

Code : 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 />';
}
?>
```

Avec un code indenté

Et voici maintenant le même code correctement indenté si on respecte la règle des tabulations :

Code : 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 !).



Certains développeurs ont tendance à remplacer les tabulations par 2 ou 4 espaces, car la largeur d'une tabulation peut varier d'un logiciel à un autre, alors que celle d'un espace est fixe. En général, c'est l'éditeur de texte lui-même qui convertit nos tabulations par des espaces, de façon transparente.

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 :

Code : PHP

```
<?php 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 quand cela n'est pas évident.

C'est le **sens général** de votre code que vous devez expliquer dans les commentaires, et non pas le rôle 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
/*
Script "Questionnaire de satisfaction"
Par M@teo21

Dernière modification : 20 Août XXXX
 */

// 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
{
    $bdd->prepare('INSERT INTO enquete VALUES (\'\', ?, ?)');
    $bdd->execute(array($_POST['description'], $_POST['mail']));

    // Puis on envoie 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 sa documentation très complète, gratuite, disponible sur Internet, et traduite dans de très nombreuses langues (dont le français 😊).

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 n'est pas ce qui se fait de plus accueillant. Mais lorsque vous commencez à réellement pratiquer le PHP, vous allez vite avoir besoin d'un support plus complet que ce cours (eh oui, la doc restera toujours plus complète que ce tutoriel 😊).

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 :

- [Affecte le comportement de PHP](#)
 - [APC](#) — Cache PHP alternatif
 - [APD](#) — Débogueur PHP avancé
 - [bcompiler](#) — Compilateur bytecode PHP
 - [Gestion des erreurs](#) — Gestion des erreurs
 - [htscanner](#) — Support du format .htaccess
 - [includ](#) — Arbre d'inclusions
 - [Options PHP et informations PHP](#)
 - [Memtrack](#)
 - [Surcharge d'objets](#)
 - [Contrôle de l'affichage](#) — Bufferisation du contenu
 - [runkit](#)
 - [scream](#) — Casse l'opérateur de silence
 - [WinCache](#) — Windows Cache pour PHP
- [Manipulation audio](#)
 - [ID3](#) — Balises ID3
 - [KTaglib](#)
 - [oggvorbis](#) — OGG/Vorbis
 - [OpenAL](#) — Gestion Audio OpenAL

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 (je suis comme vous si ça peut vous rassurer), 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, vous pouvez y voir les thèmes "Mail" et "Mathématiques". Supposons que je sois intéressé par les fonctions mathématiques de PHP. Je clique sur "[Mathématiques](#)".

Là, une nouvelle page s'ouvre. On vous propose une petite introduction que je vous recommande de lire à chaque fois, ainsi que la liste des fonctions

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.

D'autres fonctions appartiennent à des extensions qu'il faut installer manuellement.

En ce qui concerne les fonctions mathématiques, elles 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" :

- [is_finite](#) — Indique si un nombre est fini
- [is_infinite](#) — Indique si un nombre est infini
- [is_nan](#) — Indique si une valeur n'est pas un nombre
- [lcg_value](#) — Générateur de congruence combinée linéaire
- [log10](#) — Logarithme en base 10
- [log1p](#) — Calcule précisément $\log(1 + \text{nombre})$
- [log](#) — Logarithme naturel (népérien)
- [max](#) — La plus grande valeur
- [min](#) — La plus petite valeur

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 souhaite obtenir directement la description d'une fonction.

Par exemple, supposons que vous souhaitiez générer un nombre aléatoire entre 0 et 100. Vous savez que la fonction s'appelle `mt_rand` parce que quelqu'un en a parlé sur des forums.

Cette information est normalement suffisante, vous avez le nom de la fonction, vous n'avez plus qu'à vous documenter.

Pour accéder directement à la présentation d'une fonction, tapez l'adresse suivante dans votre navigateur :

`php.net/nom_de_la_fonction`

 Il est inutile d'écrire `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 :



Lorsque vous validez cette adresse, vous arrivez directement 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 :

mt_rand

(PHP 4, PHP 5)

mt_rand — Génère une meilleure valeur aléatoire

■ Description

int mt_rand (void)

int mt_rand (int \$min , int \$max)

De nombreux générateurs de nombres aléatoires provenant de vieilles bibliothèques libcs [rand\(\)](#). **mt_rand()** est une fonction de remplacement, pour cette dernière. Elle utilise un

Appelée sans les arguments optionnels *min* et *max*, **mt_rand()** retourne un nombre pse

■ Liste de paramètres

min

Valeur la plus basse qui peut être retournée (par défaut : 0)

max

Valeur la plus haute qui peut être retournée (par défaut : [mt_getrandmax\(\)](#)).

■ Valeurs de retour

Un entier aléatoire compris entre *min* (ou 0) et *max* (ou [mt_getrandmax\(\)](#), inclusif).

Ce qui nous intéresse le plus là-dedans, c'est le "Mode d'emploi de la fonction". Il correspond à ces lignes :

Code : Autre

```
int mt_rand ( void )
```

```
int mt_rand ( int $min, int $max )
```

Ces lignes décrivent le mode d'emploi de `mt_rand`. 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 qu'il y a deux façons d'utiliser la fonction : avec ou sans paramètres. Prenons le cas avec paramètres, plus complexe :

Code : PHP

```
int mt_rand ( int min, int max )
```

Examinons toutes les infos que cet extrait de code renferme :

- `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 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 compris entre 5 et 15 par exemple. La signification des paramètres est expliquée dans la section "Liste des paramètres" de la page.

Il est aussi possible d'appeler la fonction sans aucun paramètre, c'est ce que signifie la ligne suivante :

Code : Autre

```
int mt_rand ( void )
```



Mais alors... Qu'est-ce que ça signifie si on n'envoie aucun paramètre ? Que va faire la fonction ?

C'est écrit sur la page :

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. 😊

Un autre exemple : 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 :

Code : PHP

```
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 ?)

On notera qu'il y a un second paramètre entre crochets, ce qui signifie qu'il est facultatif. Il s'agit d'un int dénommé timestamp. Pour savoir ce qu'il signifie, lisez la description des paramètres.

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 qui 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 faire découvrir 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 fait (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 tutoriels du Site du Zéro. 

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, moi jamais !).

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

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 passer en revue les erreurs suivantes :

- Parse error
- Undefined function
- Wrong parameter count

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 :

Code : Autre

```
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 😊). Parfois c'est la ligne précédente qui a un problème, pensez donc à regarder autour de la ligne indiquée.

Avec un éditeur de texte spécialisé comme Notepad++, vous avez les numéros de ligne sur votre gauche comme ceci :

```
17 if (isse
18 {
19     // C
20     $ret
21     $dor
```

Bon, concrètement qu'est-ce qu'un parse error ? Un "parse error" est en fait une instruction PHP mal formée. Il peut y avoir plusieurs causes :

- 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 :

Code : PHP

```
$id_news = 5
```

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

Code : PHP

```
$id_news = 5;
```

- Vous avez oublié de **fermer un guillemet** (ou une apostrophe, ou une parenthèse). Par exemple :

Code : PHP

```
echo "Bonjour !;
```

... il suffit de fermer correctement les guillemets et vous n'aurez plus de problème 😊

Code : PHP

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

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

Code : PHP

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

Une fois corrigé, ça donne :

Code : PHP

```
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 :

Code : Autre

```
Fatal Error: Call to undefined function: fonction_inconnue() in fichier.php on li
```

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.
- 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 bibliothèque GD pour les images en PHP, on vous dira que la fonction n'existe pas. Activez la bibliothèque qui utilise la fonction et tout sera réglé. 😊

Une dernière chose : il se peut aussi que vous essayiez d'utiliser une fonction qui n'est pas disponible dans la version de PHP que vous avez.

Vérifiez dans le manuel dans quelles versions de PHP cette fonction est disponible.

Wrong parameter count

Si vous utilisez mal une fonction, vous aurez cette erreur :

Code : Autre

```
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 :

Code : PHP

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

... vous aurez l'erreur "Wrong parameter count". Pensez donc à rajouter le paramètre qui manque, par exemple comme ceci :

Code : PHP

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



Dans les versions actuelles de PHP, on vous dit même le nombre de paramètres que vous avez oublié dans le message d'erreur !

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. 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 !)

Repérer l'erreur SQL en PHP

Lorsqu'il s'est produit une erreur SQL, la page affiche le plus souvent l'erreur suivante :

```
Fatal error: Call to a member function fetch() on a non-object
```

Cette erreur survient lorsque vous voulez afficher les résultats de votre requête, généralement dans la boucle while
(\$donnees = \$reponse->fetch())

Quand vous avez cette erreur, il ne faut pas chercher plus loin, c'est la requête SQL qui précède qui n'a pas fonctionné. Il vous manque cependant des détails sur ce qui a posé problème dans la requête. 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. Je vous avais d'ailleurs présenté cette méthode dans un des premiers chapitres sur MySQL.

Vous devez utiliser un bloc try/catch autour de votre code qui fait appel à la base de données, sur ce modèle-ci :

Code : PHP

```
<?php
try
{
    $pdo_options [ PDO::ATTR_ERRMODE ] = PDO::ERRMODE_EXCEPTION;
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '',
$pdo_options);

    // Insérez ici toutes vos requêtes SQL

}
catch (Exception $e)
{
    die('Erreur : '.$e->getMessage()); // Affichage des erreurs
éventuelles
}
?>
```

Si vous respectez bien ce canevas de code, toute erreur qui se produira à l'intérieur du bloc try provoquera l'appel du bloc catch... qui se charge d'afficher l'erreur SQL.

Si la requête marche, aucune erreur ne sera affichée et le contenu du bloc catch ne sera jamais appelé. Si la requête plante en

revanche, 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 'XXX'". 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 😄).

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 :

Code : Autre

```
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 effectuées avant que le moindre code HTML ne soit envoyé. En PHP, la fonction qui permet d'envoyer des informations de headers s'appelle `header()`. 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 de ces fonctions doit être **utilisée au tout début de votre code PHP**. Il ne faut RIEN mettre avant, sinon ça provoquera l'erreur "Headers already sent by...".

Un exemple de code qui génère l'erreur :

Code : PHP

```
<html>
<?php 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 :

Code : PHP

```
<?php 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 bibliothèque 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 suivante dans votre code :

Code : PHP

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

L'erreur apparaîtra à la place du message "L'image contient des erreurs".

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

Dans les deux cas, vous verrez le message d'erreur apparaître. 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 :

Code : Autre

```
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 inaccessible !

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

Code : PHP

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

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, car c'est elle qui provoque ce problème.

Rassurez-vous : la limite est fixée à 30s, mais vous n'y serez jamais confronté. En général, un serveur met moins de 50 millisecondes à charger une page PHP (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 Zéro 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 ayez 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`. Oui, c'est un fichier qui n'a pas de nom et qui a seulement une extension, à savoir `.htaccess`. Ne soyez donc pas étonnés s'il commence par un point.

Ouvrez un nouveau fichier avec votre éditeur de texte favori. Nous allons écrire des codes qui n'ont rien à voir avec du HTML ou du PHP : ce sont des instructions pour le serveur. Elles vont lui expliquer 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/site/www/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 car cela dépend du serveur. 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. Inscrivez juste cette ligne de code à l'intérieur :

Code : PHP

```
<?php 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/site/www/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/site/www/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.

Si vous êtes hébergés chez Free, il y a une petite subtilité dans la gestion de la localisation du `.htpasswd` : vous ne devez pas renseigner la ligne `AuthUserFile` par le chemin absolu du fichier, mais par son chemin *relatif* à partir de la racine de votre espace perso. Exemple : si vous utilisez un espace Free nommé `monftpfree`, et que vous placez votre fichier `.htpasswd` dans un répertoire `admin`, le fichier `chemin.php` vous renverra un chemin sous la forme `/mnt/XXX/sda/X/X/monftpfree/admin/.htpasswd`. Vous devez alors simplement écrire :
`/monftpfree/admin/.htpasswd`. D'autre part, il ne faut pas écrire `AuthUserFile` mais `PerlSetVar`

AuthFile.

Enregistrez le fichier en inscrivant le nom entre guillemets, comme ceci : ".htaccess". Cela permet de forcer l'éditeur à enregistrer un fichier qui commence par un point.

Voilà, on a fini de créer le .htaccess, on peut maintenant passer au .htpasswd ! 😊

Créer le .htpasswd

Créez maintenant un nouveau fichier avec votre éditeur de texte.

Le .htpasswd va contenir la liste des personnes autorisées à accéder aux pages du dossier. On y inscrit une personne par ligne, sous cette forme :

Code : Autre

```
login:mot_de_passe_crypté
```

Au final, votre fichier .htpasswd devrait ressembler à ceci :

Code : Autre

```
mateo21:$1$MEqT//cb$hAVid.qmmSGFW/wD1IfQ81  
ptipilou:$1$/gP8dYa$sQNXCCP47KhP1sneRIZ0O0  
djfox:$1$1T7nqnsq$cVtoPfe0IgrjES7Ushmoy.  
vincent:$1$h4oVHp3O$X7Ejpn.uuOhJRkT3qnw3i0
```

Dans cet exemple, il y a 4 personnes autorisées à accéder au dossier : mateo21, ptipilou, djfox et vincent.



Comment peut-on crypter les mots de passe ?

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, elle 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 :

Code : PHP

```
<?php 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**. Il est à sens unique.

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 (non non, ne me remerciez pas, c'est tout naturel !). 😊

Code : PHP

```
<p>  
<?php  
if (isset($_POST['login']) AND isset($_POST['pass']))  
{  
    $login = $_POST['login'];  
    $pass_crypté = crypt($_POST['pass']); // On crypte le mot de passe  
  
    echo 'Ligne à copier dans le .htpasswd :<br />' . $login . ':' .  
    $pass_crypté;
```

```
}

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 :

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 chargement 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 flemme de le créer, pas de souci, vous n'avez qu'à venir sur cette page et cliquer sur le bouton "Essayer !". 😊



Il y a certains cas où vous ne devrez pas crypter les mots de passe. Sous WAMP ou sur les serveurs de Free.fr par exemple, vous ne DEVEZ PAS crypter vos mots de passe pour que cela fonctionne. Vous devrez donc les écrire directement. Par exemple :

`mateo21:kangourou`

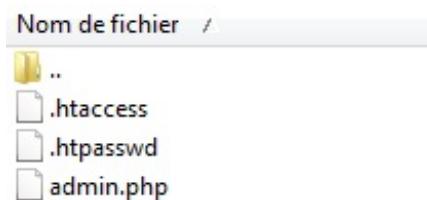
Envoyer les fichiers sur le serveur

Vous avez maintenant 2 fichiers sur votre disque dur : .htaccess et .htpasswd.

Lancez votre logiciel FTP.

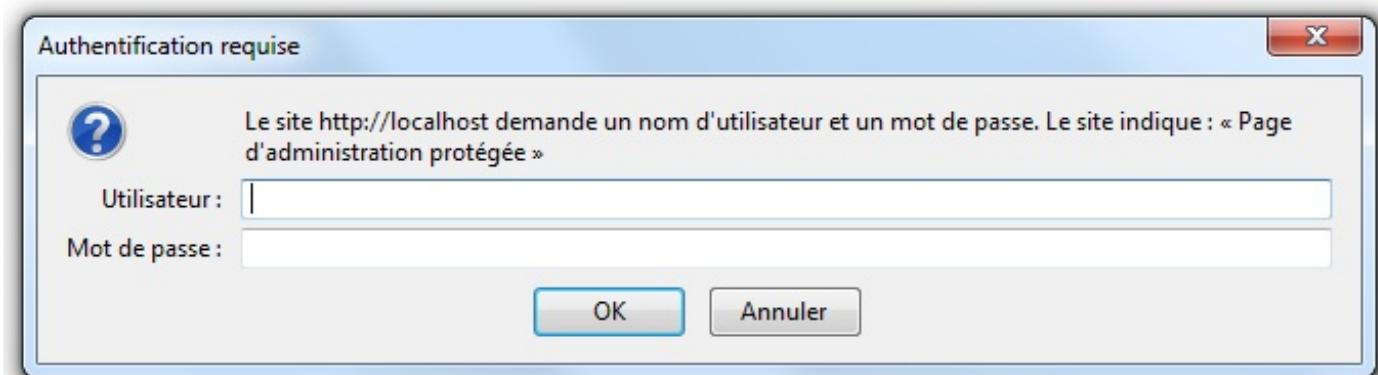
Transférez les fichiers .htaccess et .htpasswd dans le dossier que vous voulez protéger par mot de passe.

Vous devriez voir ceci dans votre logiciel FTP :



Voilà, désormais le dossier est protégé 😊

Si quelqu'un essaie d'accéder à une des pages du dossier (en l'occurrence admin.php), 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 !

La protection par .htaccess que je viens de vous présenter a l'avantage d'être rapide et simple à mettre en place. Elle vous permet donc de créer une zone Admin en moins de temps qu'il n'en faut pour le dire. 😊

Cependant, sur un site un peu plus évolué, vous aurez probablement besoin de limiter les accès à certaines sections du site en fonction du nom du membre ou du groupe auquel il appartient. Là, tout dépend de l'architecture de votre site. En règle générale, un if suffit à déterminer si la page peut être affichée ou non.

Memento des expressions régulières

Cette annexe va être utile à ceux qui ont lu les 2 chapitres sur les expressions régulières. Il s'agit d'un memento, c'est-à-dire d'un résumé qui vous sera utile lorsque vous écrirez 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 succinctes 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 :

Code : PHP

```
<?php 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" doit apparaître au moins 1 fois
#a*#	"a" peut apparaître 0, 1 ou plusieurs fois
#bor?is#	"bois" ou "boris"
#Ay(ayloy)*#	Fonctionne pour Ay, Ayay, Ayoy, Ayayayoyayoyoyoy 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 :

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

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 placer 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 remplacements à l'aide de Regex.

Code : 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
- `(?:texte)` 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)(tutionnelle)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 occurrence 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. 😊
