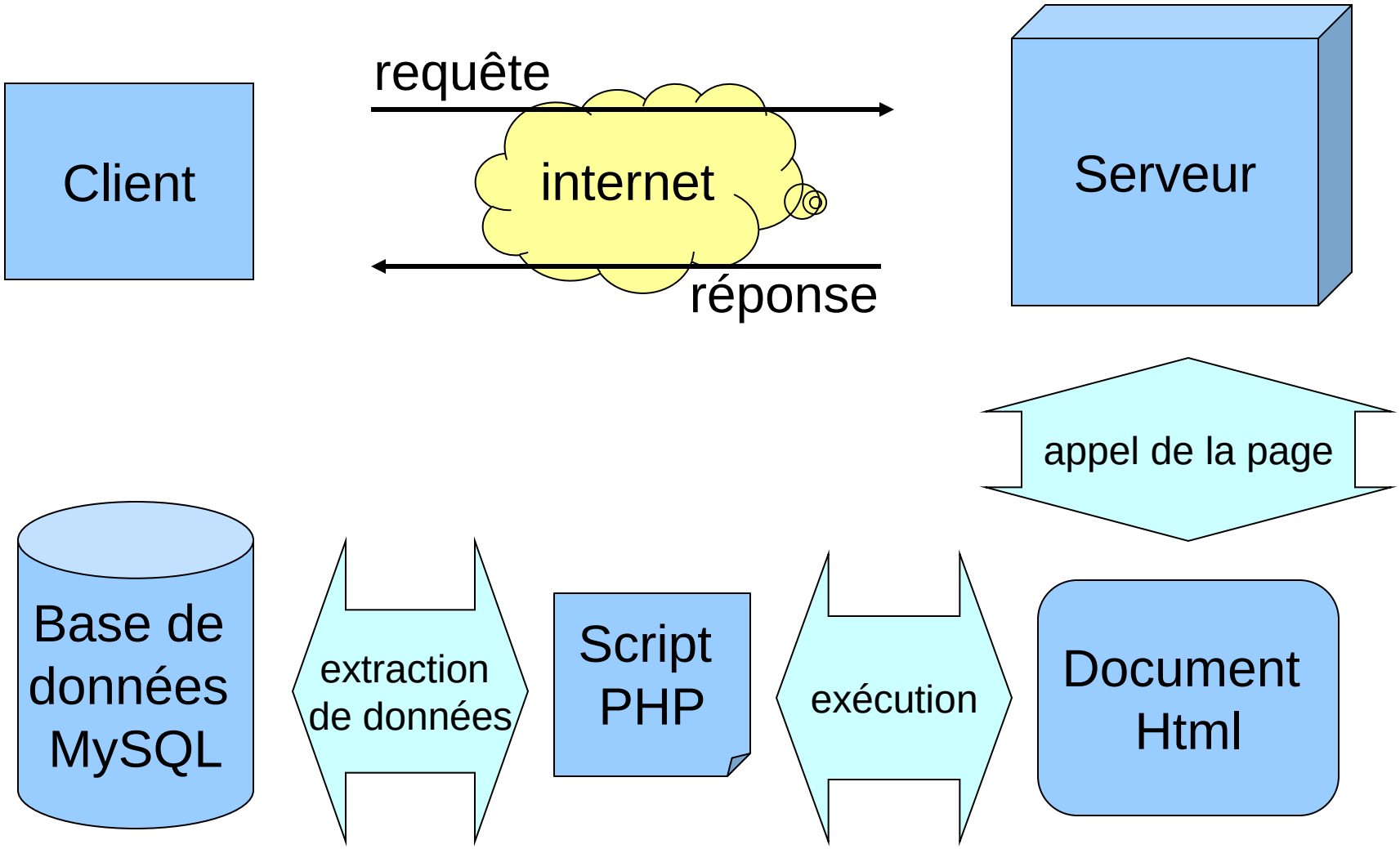
Le langage PHP

Modèle


Client
requête
internet
réponse
Serveur
appel de la page
Base de données MySQL
extraction de données
Script PHP
exécution
Document Html

Documentation en ligne

Pour obtenir en ligne toute la documentation officielle (en français) sur une commande, tapez l'URL suivante dans la barre d'adresse de votre navigateur Internet :

http://fr.php.net/

Et rajouter en fin d'URL le nom de la commande.

Exemple :

http://fr.php.net/echo

La petite histoire du PHP

Il a été créé en 1994 par Rasmus Lerdorf pour les besoins des pages web personnelles (livre d'or, compteurs, etc.). A l'époque, PHP signifiait *Personnal Home Page*.

C'est un langage incrusté au HTML et interprété côté serveur. Il dérive du C et du Perl dont il reprend la syntaxe. Il est extensible grâce à de nombreux modules et son code source est ouvert. Comme il supporte tous les standards du web et qu'il est gratuit, il s'est rapidement répandu sur la toile.

En 1997, PHP devient un projet collectif et son interpréteur est réécrit par Zeev Suraski et Andi Gutmans pour donner la version 3 qui s'appelle désormais *PHP : Hypertext Preprocessor*.

Il existe par ailleurs des applications web prêtes à l'emploi (PHPNuke, PHP SPIP, PHPSlash…) permettant de monter facilement et gratuitement son portail. En juillet 2000 plus de 300.000 sites tournaient déjà sous PHP !

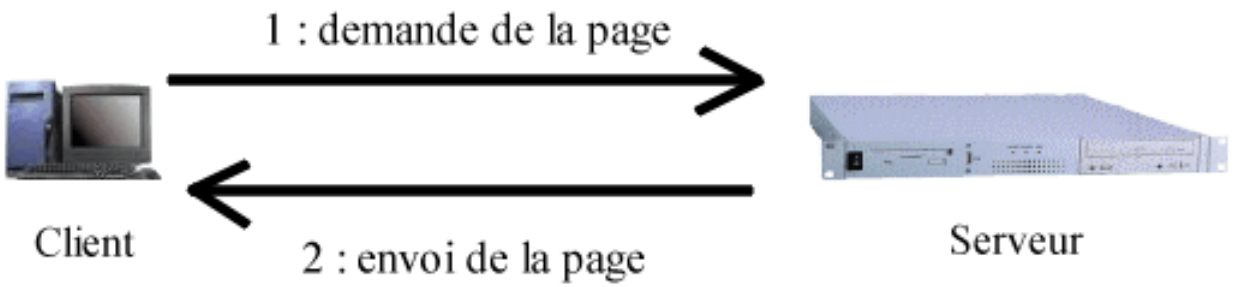
PHP ou HTML ?

Les 2! Les scripts PHP sont encapsulés dans le code HTML.
Néanmoins, la page contenant des instructions PHP aura une extension .php

Le HTML est pratique mais limité. Le PHP permet de réaliser, entre autre :

Un forum de discussions, un livre d'or, un chat, un compteur de visite ou des application plus spécifiques.

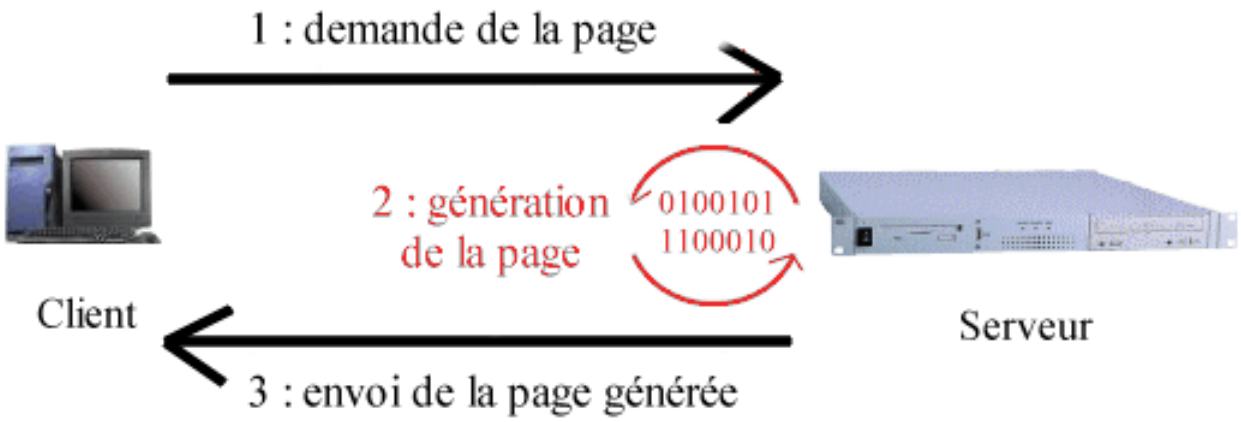
<u>Rappel sur le HTML :</u>



Le client (vous) demande à voir une page web, le serveur répond à la demande et envoi la page demandée.

PHP ou HTML ?

En PHP :



Le client souhaite voir une page PHP. Il fait la demande au serveur.

Le serveur n'a pas la page en « stock », il la génère spécialement pour le client, ce dernier n'étant pas capable de lire (et surtout comprendre) directement le PHP. Le serveur transforme donc la page PHP en HTML ayant au préalable exécuté les commandes (et script) de la page PHP.

Le serveur peut alors envoyé la page (html) au client.

La page générée est donc UNIQUE !

Intégration d'un script dans une page

Les pages web sont au format html. Les pages web dynamiques générées avec PHP4 sont au format php. Le code source php est directement inséré dans le fichier html à l'aide de la balise :

<?php … ?>

Exemple:

```
<html>
<body>
<?php
        echo ''Bonjour'';
?>
</body>
</html>
```

Autres syntaxes d'intégration :

- **<? … ?>**
- **<script language=''php''> … </script>**
- **<% … %>**

Exemple de script

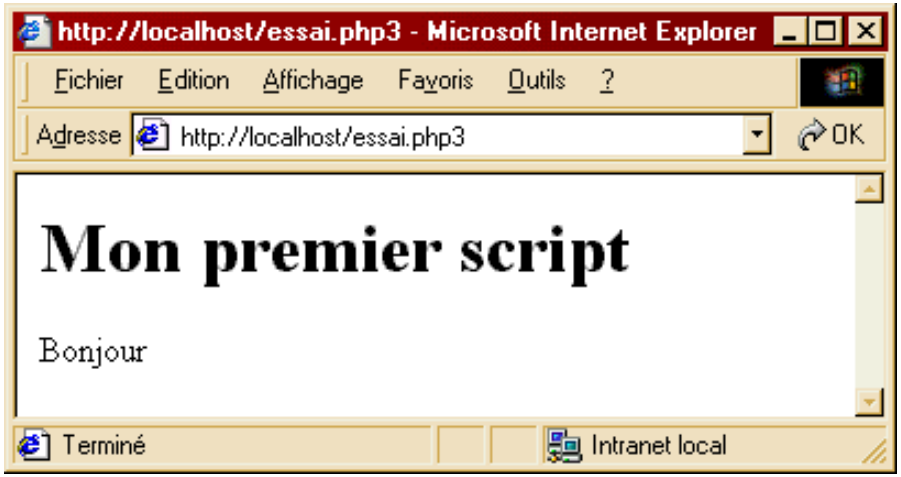
Exemple de script, code source (côté serveur) :

```
<html>
<body>
<h1>Mon premier script</h1>
<?php echo ‘’Bonjour”; ?>
</body>
</html>
```

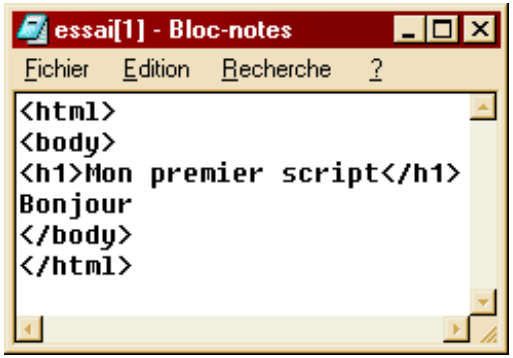
Autre écriture du même script :

```
<?php
echo ‘’<html><body>”;
echo ‘’<h1>Mon premier script</h1>”;
echo ‘’Bonjour”;
echo ‘’</body></html>”;
?>
```

Résultat affiché par le navigateur :



Code source (côté client) de la page essai.php résultant du script



De quoi avons-nous besoin ?

Pour développer en PHP, il est nécessaire d'installer plusieurs outils :

- un éditeur de texte, Bloc Note ou Notepad++ (qui reconnaît le PHP) pour taper les codes du langage,
- un serveur PHP, qui va interpréter le langage,
- un navigateur pour afficher les pages.

Comme serveur PHP, nous utiliserons EasyPHP (www.easyphp.org). Ce dernier va installer :

- Apache : c'est le programme qu'utilisent les serveurs. Il permet au serveur de distribuer des pages web... mais il ne connaît que le HTML !
- PHP : PHP est comme un "plugin" de Apache. Il a besoin d'Apache pour fonctionner, et grâce à lui Apache saura travailler sur des pages PHP. En clair, Apache + PHP = un serveur PHP
- MySQL : c'est un programme qui permet d'utiliser des bases de données.
- PHPmyAdmin : c'est une interface d'administration des BDD.

Installons EasyPHP

Télécharger et installer EasyPHP v1.7

Lancer EasyPHP. L'icône apparaît dans la barre des tâches.

Cliquer avec le bouton droit de la souris sur l'icône.

C'est le menu « Administration » qui nous intéresse.

(1) Définition des sites web.

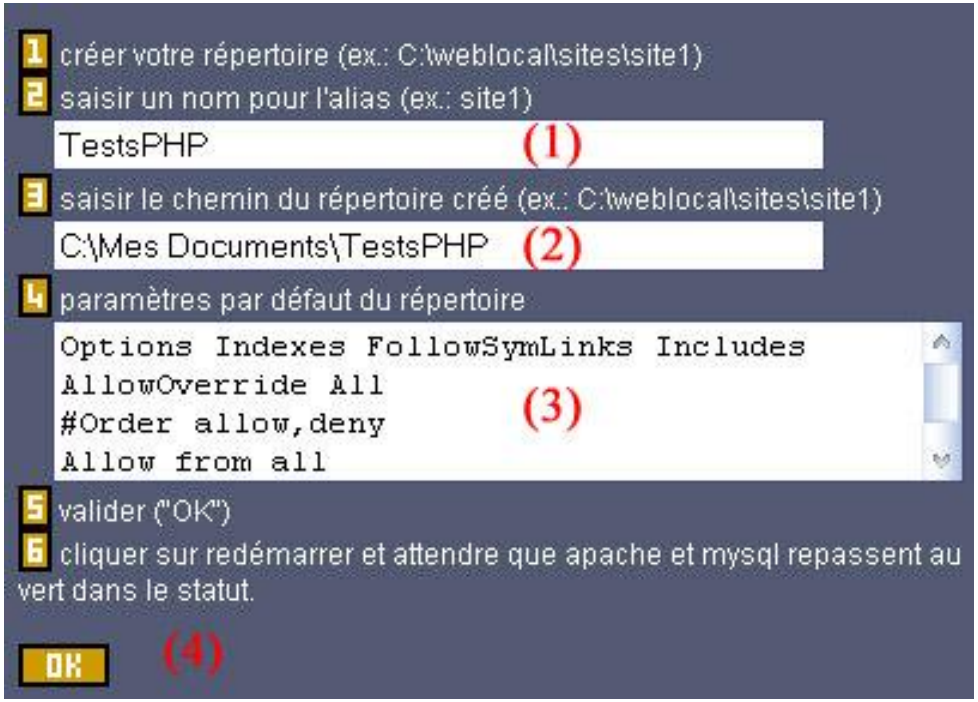
(2) Gestion de la base de données.

Configurons EasyPHP

Dans le menu « Apache », cliquer sur « Ajouter… »

Indiquer les informations :
Nom (1) et emplacement (2) du site à créer.

(3) NE RIEN TOUCHER!!!



Si le serveur est en fonctionnement, l'adresse locale devient :

http://localhost ou http://128.0.0.1

En tapant une des ces deux adresses dans le navigateur, on atteint la racine de site. Il est alors possible d'exécuter les page PHP.

Mais passons aux choses sérieuses…

Commentaires

Un script php se commente :

Exemple :

<?php

// commentaire de fin de ligne

/* commentaire
sur plusieurs
lignes */

 # commentaire de fin de ligne comme en Shell
?>

Tout ce qui se trouve dans un commentaire est ignoré. Il est conseillé de commenter largement ses scripts.

Afficher du texte

Taper le code suivant :

```
<html>
   <head>
      <title>Notre première instruction : echo</title>
   </head>
   <body>
      <h2>Affichage de texte avec PHP</h2>
<p>
   Cette ligne a été écrite entièrement en HTML.<br>
   <? echo "Celle-ci a été écrite entièrement en PHP."; ?>
</p>
</body>
</html>
```

ECHO est l'instruction qui donne l'ordre d'afficher du texte.

Attention à ne pas oublier le « ; » à la fin de la ligne!

*Note : pour afficher des " il convient de les faire précéder de *

Variables, types et opérateurs (I)

Une variable est une 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.

C'est donc au « codeur » à créer des variables, autant que nécessaire.

Une variable est toujours constituée de 2 éléments :

- Son nom : pour pouvoir la reconnaître, par exemple "jour".
- Sa valeur : c'est l'information qu'elle contient, qui peut changer. Par exemple "jeudi".

Affectation d'une valeur à une variable :

```
<?
$votrenom = "Jean-Claude";
?>
```

IMPORTANT : Le signe $ précède toujours la variable.

Variables, types et opérateurs (II)

Il existe plusieurs type de données :

- ❑ le texte : suite de caractères alphanumérique entre guillemets.
- ❑ le numéraire : chiffres et nombres sans guillemet.
- ❑ les booléens : expression logique, vrai (true) ou faux (false).

Exemples :

```
<?
$nombre_de_stagiaires = 12;
?>
```

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

Variables, types et opérateurs (III)

Afficher la valeur d'une variable :

```
<?
$pseudo_du_stagiaire = "Jean Brasse";
echo "Bonjour $pseudo_du_stagiaire !";
?>
```

Les opérateurs :

Opérateurs arithmétiques :
+ (addition), **-** (soustraction), ***** (multiplié), ***/*** (divisé),
++ (incrément), **--**(décrément).

Opérateurs logiques :
and, **&&** (et), **or**, **||** (ou), **xor** (ou exclusif), **!** (non)

Opérateurs de comparaison :
== (égalité), **<** (inférieur strict), **<=** (inférieur large), **>**, **>=**, **!=** (différence)

Variables, types et opérateurs (IV)

Faire des calculs simples :

```
<?
$nombre1 = 2 + 4; // $nombre prend la valeur 6
$nombre2 = 5 - 1; // $nombre prend la valeur 4
$nombre3 = 3 * 5; // $nombre prend la valeur 15
$nombre4 = 10 / 2; // $nombre prend la valeur 5

// Allez on rajoute un peu de difficulté
$nombre5 = 3 * 5 + 1; // $nombre prend la valeur 16
$nombre6 = (1 + 2) * 2; // $nombre prend la valeur 6
?>
```

OU :

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

Variables, types et opérateurs (V)

Transmettre des variables :

1) Il est possible de transmettre des variables vers une autre page par l'intermédiaire de l'URL :

http://www.monsite.com/infos.php?jour=01&mois=01&annee=2005&titre=Informations

Les variables sont :
jour, mois, annee, titre.

L'écriture de cette URL va créer les variables suivantes :

- $_GET['jour'] = 01;
- $_GET['mois'] = 01;
- $_GET['annee'] = 2005;
- $_GET['titre'] = "Informations";

qu'il suffira de rappeler dans le script PHP.

Note : plusieurs variables dans l'URL seront séparer par & (code de « & »)

Variables, types et opérateurs (VI)

Exercice :

PAGE « appel.php » :

```
<html>
        Cette page ne contient que du HTML.<br>
        Voici 3 liens vers la page cible.php, avec des variables aux valeurs différentes :<br>
        <a href="cible.php?nom=Brasse&prenom=Jean">Lien de Jean Brasse</a><br>
        <a href="cible.php?nom=Proviste&prenom=Alain">Lien vers Alain Proviste</a><br>
        <a href="cible.php?nom=Huaire&prenom=Anne">Lien vers Anne Huaire</a>
</html>
```

PAGE « cible.php »

```
<html>
        Bonjour !
Votre nom est <? echo $_GET['nom']; ?> , et votre prénom est <? echo $_GET['prenom']; ?>.<br>
Faites un autre essai, <a href="appel.php">cliquez ici</a> pour revenir à appel.php.
<html>
```

Variables, types et opérateurs (VII)

2) Il est possible de transmettre des variables par formulaire.

On crée dans le fichier « appel.php » une zone de texte dans lequel sera inscrit du texte :

```
<html>
        Cette page, elle aussi, ne contient que du HTML.<br>
        Veuillez taper votre prénom :
        <form action="cible.php" method="post">
        <input type="text" name="prenom"><input type="submit"
value="Valider">
        </form>
</html>
```

On récupère la variable dans le fichier « cible.php » à l'aide de la variable $_POST['variable'] :

```
<html>
        Bonjour ! Tu t'appelles <? echo $_POST['prenom']; ?> !<br>
        Si tu veux changer de prénom, <a href="appel.php">clique ici</a> pour
        revenir à appel.php.
</html>
```

Les fonctions (I)

Une **fonction** est une série d'instructions qui retourne une valeur.

Il est intéressant d'utiliser les fonctions lors de l'utilisation complexe, longue ou répétitive d'instructions PHP.

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

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

$nom = "Claude";
echo "Bonjour, $nom !";

Etc…
?>
```

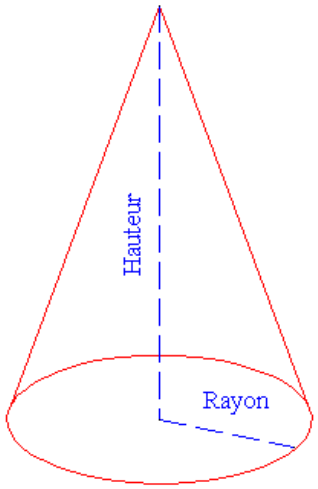
```php
<?
function DireBonjour($nom)
{
echo "Bonjour $nom !<br>";
}


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

Note : Les { } définissent les limites de la fonction.
La ligne comportant « function » ne se termine pas par « ; »!

Les fonctions (II)

On souhaite définir une fonction qui calcul le volume d'un cône.
Les variables seront le rayon et la hauteur.
La fonction retournera le résultat du calcul.

La formule du volume d'un cône est la suivante :
V = 2 * PI * R * H * (1/3)




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

Note : Utilisation du « . » à la place de la « , »!

Les fonctions (III)

Création de la fonction :

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

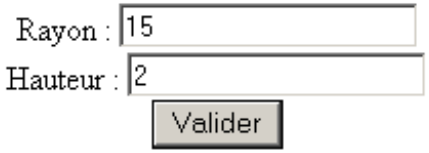
L'instruction RETURN indique que la fonction doit renvoyer le résultat du calcul dans la variable $volume.

Les fonctions (IV)

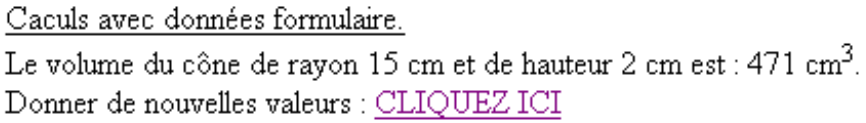
Exercice :

Sur la base des exercices précédents, on souhaite réaliser la fonction à partir de données envoyées par formulaire.

Le formulaire permettra d'inscrire les données nécessaire au calcul :



Une page de résultat sera affichée :



Les fonctions (V)

PHP intègre d'innombrables fonctions. Il est alors utile de s'en servir pour réaliser des calculs ou commandes particulières.

Il y a deux type de fonctions :

- celles qui effectuent des actions et ne renvoient aucune valeur,
- celles,qui après calculs, renvoient une valeur.

Les fonctions le plus courantes permettent :

- de rechercher et de remplacer des mots dans une variable
- d'envoyer un fichier sur un serveur
- de créer des images miniatures (aussi appelées thumbnails)
- d'envoyer un mail avec PHP (très pratique pour faire une newsletter !)
- de modifier des images, y écrire du texte, tracer des lignes, des rectangles etc...
- de crypter des mots de passe.
- de renvoyer l'heure, la date...
- etc…

Les fonctions (V)

Application d'une fonction : date

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

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

Exemple :

```
<?
// 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.";
?>
```

Les conditions (I)

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

La structure IF …. ELSE

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

IF introduit la condition. Elle s'exprime entre ().
Si la condition est remplie, les instructions entre accolades seront exécutées.
On ajoute ELSE (si la condition n'est pas remplie) :

```
<?
$age = 8;
if ($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";
?>
```

Les conditions (II)

On peut alors utiliser la variable booléenne pour d'autres conditions :

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

« Else if » signifie « Sinon si ».

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.

Note : au départ une variable est vide! Pour vérifier : if ($variable == NULL)

Les conditions multiples

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

```
<?
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 !";
}
?>
```

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

Les conditions booléennes

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

Ce sont les variables qui valent vrai (true) ou faux (false).

En omettant le résultat " == true " , la condition est tout de même réalisée.

<u>Remarque :</u>
if ($variable == false)
équivaut à
if (NOT $variable)

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

L'astuce sur les conditions

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

EQUIVAUT A :

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

Si la condition remplie nécessite l'exécution d'un (long) code HTML, il s'avère pratique d'ouvrir l'accolade juste après la condition puis de fermer la balise PHP (?>) et d'insérer le code HTML souhaité.

Penser enfin à rouvrir la balise PHP, à refermer l'accolade puis à refermer PHP.

TP n°1 : protection par mot de passe (I)

Ce premier TP va permettre de mettre en pratique les quelques instructions PHP vue jusqu'à présent et de vérifier l'efficacité du langage.

Notions requises :

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

Problème posé : vous avez créé une page web qui contient des informations confidentielles et vous voulez la protéger par mot de passe pour que seuls ceux possédant le mot de passe puissent y accéder. Sans ce mot de passe, on ne doit pas pouvoir afficher la page.

Schéma du code :

On ne doit travailler que sur une seule page.

Cette page affiche au départ une zone de texte pour rentrer le mot de passe.

Si le mot de passe est bon, on affiche les informations confidentielles. Sinon, on propose à nouveau de rentrer le mot de passe.

TP n°1 : protection par mot de passe (II)

Quelques indices :

- une seule variable est nécessaire pour le code, celle correspondant au mot de passe : $mot_de_passe
- Récupérer le mot de passe entré par formulaire : $_POST['mot_de_passe']
- Puisqu'au premier affichage, aucun mot de passe n'a été entrée, la variable $_POST['mot_de_passe'] est donc vide : inutile de vérifier si le mot de passe est bon! C'est seulement lorsque le visiteur clique sur « Envoyer » pour valider le mot de passe que la variable est créée.
  La fonction PHP « isset » sera vraie si la variable existe : autant l'utiliser!
  Cela donne :

```
(isset($_POST['mot_de_passe']))
{
// on peut vérifier le mot de passe
}
```

A VOUS DE JOUER !

Les boucles (I)

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

Sans instruction supplémentaire, cette boucle se réalise indéfiniement. Il faut donc ajouter une condition : « WHILE » (tant que)

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

Les boucles (II)

Essayer ce code :

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

La boucle pose la condition : tant que le nombre de ligne est inférieur à 100

Si la condition esr remplie on affiche un message puis on incrémente (valeur = valeur +1) la variable $nombre_de_lignes.

Important : Il faut TOUJOURS s'assurer que la condition soit au moins remplie une fois. Si elle ne l'est jamais, alors la boucle s'exécutera à l'infini !

Note : L'astuce concernant les conditions est aussi valable pour les boucles!

Les boucles (III)

L'instruction **FOR**.

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

On définit en une seule ligne trois éléments, séparés par des « ; » :

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

Les tableaux (array) (I)

Un « array » est une variable « tableau » un peu spéciale. Elle peut prendre plusieurs valeurs. Prenons un exemple :

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

Nom	Valeur
$prenom	Nicole

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

Il est pourtant possible de donner à la variable $prenom plusieurs valeurs. $prenom est une **array.** Les valeurs sont rangées dans des cases différentes, et sont numérotées dans ce cas.

Pour afficher Benoît, il faut faudra pas appeler la variable $prenom mais lui indiquer d'afficher le contenu de la case n°4 de la varialbe $prenom :

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

Les tableaux (array) (II)

Créer un tableau : la fonction « ARRAY » :

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

Attention, l'ordre a beaucoup d'importance. Il va déterminer le n° des « cases ».

Exercice :

Créer un script qui crée un tableau avec 5 prénoms et qui affiche sur des lignes séparée :

Le prénom n° $numero est : $prenom[$numero]

La variable $numero pendra les valeurs 0 à 4 (incrément).

Les tableaux (array) (III)

Solution :

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

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

Les tableaux (array) (IV)

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

Solution : Les tableaux associatifs : plutôt que d'utiliser des numéros pour les cases du tableau, on va utiliser des étiquettes :

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

Note : une seule instruction, un seul « ; », => pour faire l'association.

Pour afficher le contenu de cet array : echo $coordonnees['Ville']

Partie 2 : MySQL

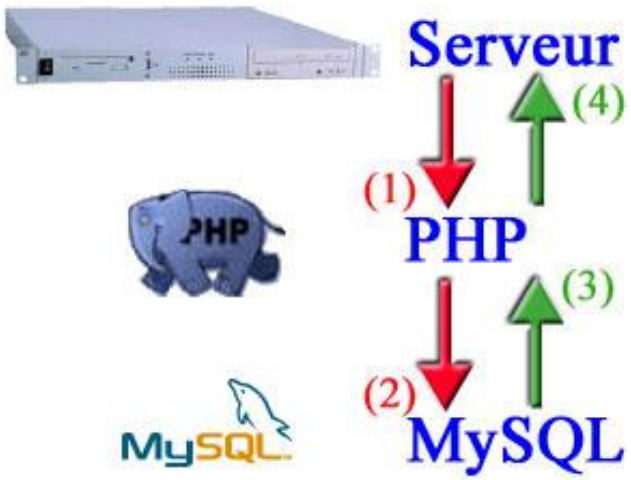
Présentation

MySQL est une base de données implémentant le langage de requête SQL un langage très connu.

Il existe un outil libre et gratuit développé par la communauté des programmeurs libres : phpMyAdmin qui permet l'administration aisée des bases de données MySQL avec PHP.

phpMyAdmin est inclus dans EasyPHP.

Avec MySQL on peut créer plusieurs bases de données sur un serveur. Une base est composée de tables contenant des enregistrements.



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

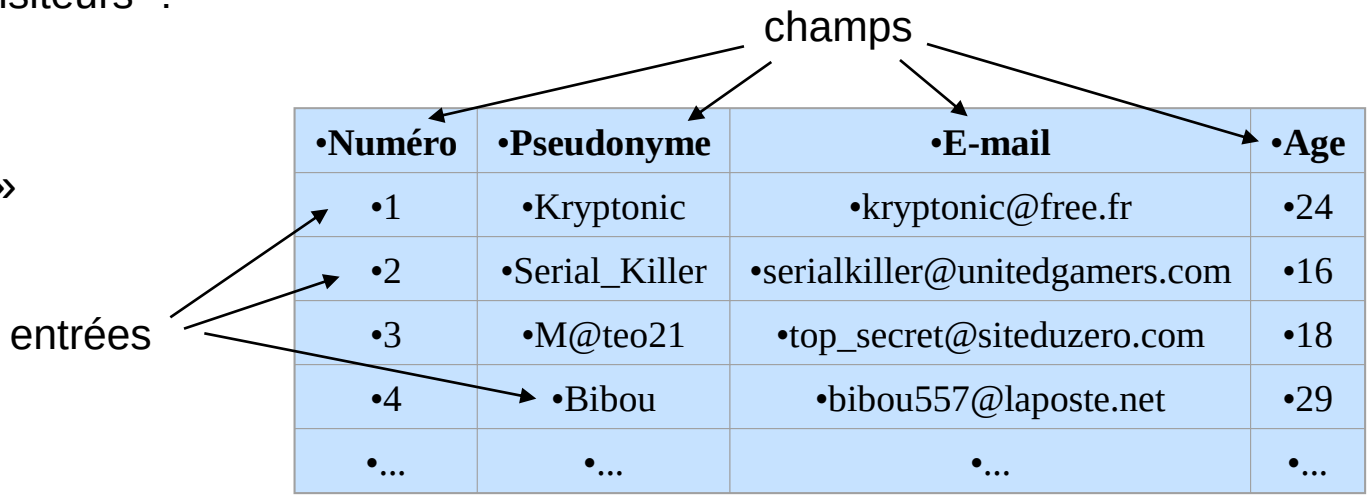
Structure d'une base de données

La base de donnée utilise un vocabulaire précis :

•**La base**, c'est l'armoire. C'est le gros meuble dans lequel les secrétaires ont l'habitude de classer les informations.

•Dans une armoire, il y a plusieurs tiroirs. Un tiroir, dans le langage MySQL, c'est ce qu'on appelle **une table**. Chaque tiroir contient des données différentes. Par exemple, on peut imaginer un tiroir qui contient les pseudonymes et infos sur vos visiteurs, un autre qui contient les messages postés sur un forum...

•Mais que contient une table ? C'est là que sont enregistrées les données, sous la forme d'un tableau. Dans ce tableau, les colonnes sont appelées **des champs**, et les lignes sont appelées **des entrées**. Par exemple, voici à quoi peut ressembler le contenu d'une table appelée "visiteurs" :



•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
•...	•...	•...	•...

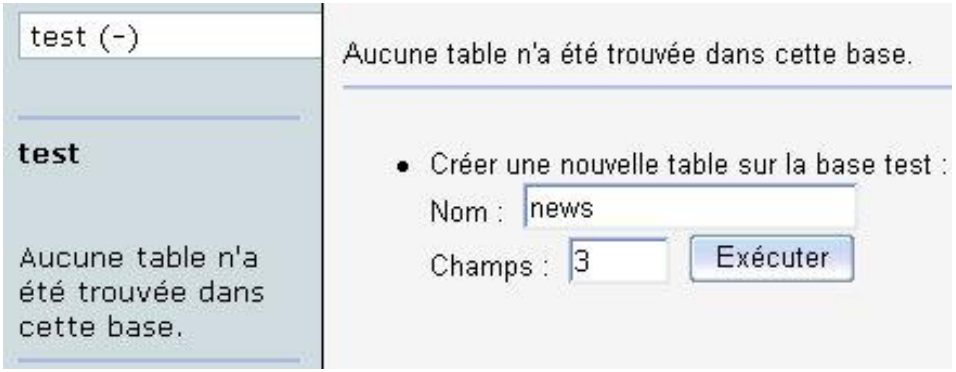
phpMyAdmin

Pour effectuer des manipulations sur les bases de données, nous utiliserons un ensemble de pages PHP qui dialogue avec la base de données : phpMyAdmin.

1. Liste des bases : dans ce menu déroulant sont listées vos bases de données. Le nombre entre parenthèses, c'est le nombre de tables qu'il y a dans la base.
2. Créer une base : tapez un nom pour votre base de données, cliquez sur "Créer" et hop ! C'est fait.

Création d'une base

Nous allons travailler sur la base « test », et créer une table « news » :



Et créer trois champs :

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

- titre : ce champ contiendra le titre de la news.
- contenu : enfin, ce champ contiendra la news en elle-même.

Champ	Type [Documentation]	Taille/Valeurs*
id	MEDIUMINT	
titre	TEXT	
contenu	TEXT	

Les champs

Un champ peut contenir du texte, des nombres, des dates etc...
Il faut donc définir quel type de données contiendra le champ.
Voici les principaux types de données :

•INT : nombre entier. Il y a plusieurs variantes, selon la grandeur des nombres que ça peut comporter. Dans l'ordre, il y a TINYINT (très petit, c'est-à-dire 255 maximum), SMALLINT (jusqu'à 30 000), MEDIUMINT (8 000 000), INT (2 000 000 000), BIGINT (vraiment beaucoup !).

•TEXT : du texte. Là encore il y a plusieurs variantes, ça fonctionne de la même manière. A vous de choisir celui qui vous paraît le plus adapté.

•DATE : date de la forme "YYYY-MM-DD", "YY-MM-DD" ou "YYMMDD" (c'est le format américain, eh oui !)

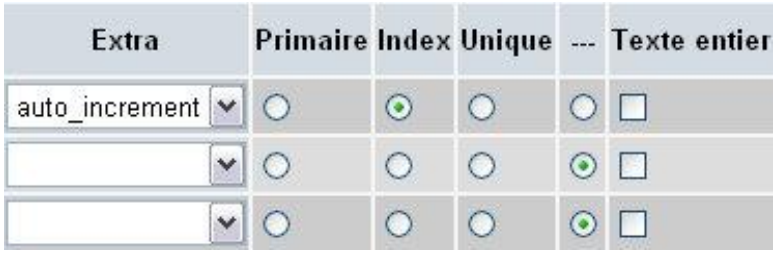
•TIME : l'heure, de la forme "HH:MM:SS" ou "HHMMSS" ou "HHMM" ou "HH".

•DATETIME : mélange la date et l'heure, de la forme "YYYY-MM-DD HH:MM:SS"

•BLOB : plus particulier, ce type est rarement utilisé. Il permet de stocker des fichiers dans la base de données.

Options du champ « id » :

- *auto_increment* pour incrémenter automatiquement la valeur de id
- *index* pour accélerer les recherches.



Modification d'une base (I)

La table « news » est visible sous la base « test ».

Affichage de la structure de la base :

Base de données *test* - Table *news* sur le serveur *localhost*

Structure | Afficher | SQL | Sélectionner | Insérer

	Champ	Type	Attributs	Null	Défaut	Extra	Action
☐	id	mediumint(9)		Non		auto_increment	
☐	titre	text		Non			
☐	contenu	text		Non			

Tout cocher / Tout décocher *Pour la sélection :*

Index : [Documentation]

Nom de la clé	Type	Cardinalité	Action	Champ
id	INDEX	aucune	Supprimer Modifier	id

Créer une clef sur 1 colonne(s) Exécuter

Espace utilisé :

Type	Espace
Données	0 Octets
Index	1 024 Octets
Total	1 024 Octets

St

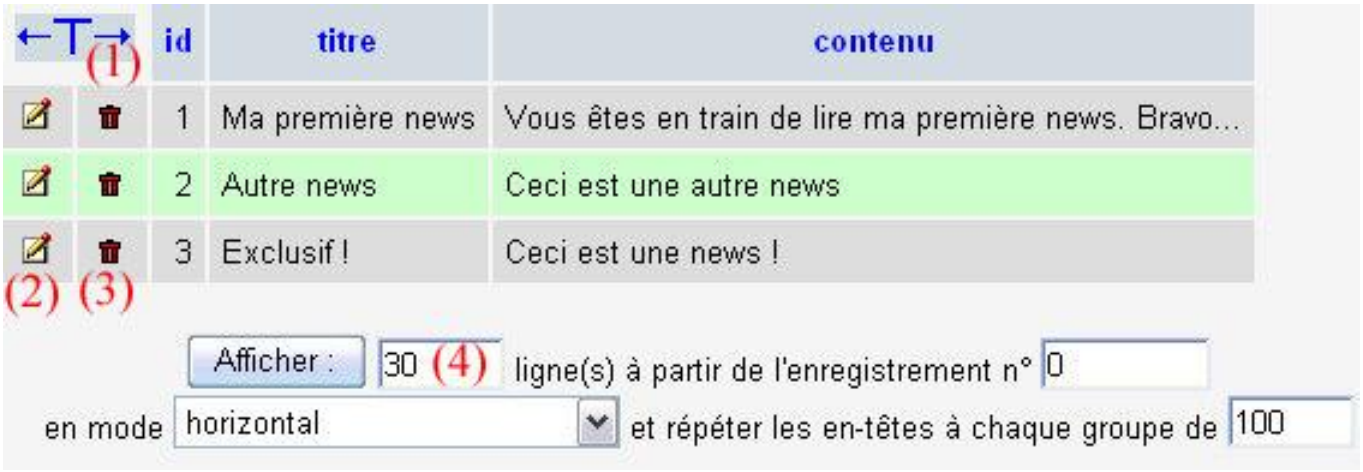
Modification d'une base (II)

On clique sur « Insérer »

Remplir « titre » et « contenu » puis « Exécuter », ceci plusieurs fois.

Modification d'une base (III)

On affiche le contenu de la base : « Afficher »



1. Afficher tout le texte : en cliquant sur le T majuscule, cela affiche la totalité du texte.
2. Modifier l'entrée : cette petite image permet de modifier l'entrée sélectionnée (pour apporter des modifications à une news par exemple).
3. Supprimer l'entrée : ce bouton supprime l'entrée sélectionnée.
4. Afficher X lignes à partir de l'enregistrement n° X : s'il y a beaucoup d'entrées dans la table, PhpMyAdmin n'en affichera qu'une sélection (les 30 premières lignes normalement). Pour en afficher plus, il suffit de modifier ces valeurs puis de cliquer sur "Afficher".

Autres opérations

•**SQL :** Pour effectuer des requêtes dans la base.

•**Exporter :** Pour récupérer le contenu et/ou la structure de la base de données sur le disque dur (sous forme de fichier).

Utile pour la sauvegarde!

•**Opérations :** changer le nom de la table, déplacer, copier, réparer et optimiser la table.

•**Vider :** Pour supprimer tout le contenu (les champs) de la table. La structure est maintenue.

•**Supprimer :** Pour supprimer la totalité de la base : structure + données.

Connexion

Pour lire des données dans une base de données, il faut s'y connecter.

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 s'y trouvant (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, il sera possible d'effectuer n'importe quelle opération sur la base de données.

Utilisation de la fonction : **mysql_connect**

Cette fonction a besoin de trois arguments :

Le nom de l'hôte : c'est l'IP de l'ordinateur où MySQL est installé. Le plus souvent, MySQL est installé sur le même ordinateur que PHP. Dans ce cas, mettez la valeur "localhost".

- Le login : c'est l'identifiant de connexion. Avec EasyPHP : root
- Le mot de passe. Avec EasyPHP : aucun mot de passe.

```
<?
mysql_connect("localhost", "monlogin", "monmotdepasse");
?>
```

Choix de la base

Une fois connecté, il faut sélectionner la base de données sur laquelle on veut travailler. C'est la fonction : **mysql_select_db**

```
<?
mysql_connect("localhost", "monlogin", "monmotdepasse"); // Connexion à MySQL
mysql_select_db("mabase"); // Sélection de la base mabase
?>
```

Une fois le travail terminé, il convient de se déconnecter de la base : **mysql_close**

```
<?
mysql_connect("localhost", "monlogin", "monmotdepasse"); // Connexion à MySQL
mysql_select_db("mabase"); // Sélection de la base mabase

// connecté, on peut travailler sur la BDD
// ...
// ...

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

Lire des données

L'objectif est de créer un page PHP pour afficher le contenu d'une table dans une base de données.

Faire une requête : **mysql_query**

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

$reponse contiendra la réponse de MySQL.

La requête doit être faite avec un langage reconnu par MySQL : le SQL!

La première requête : **SELECT * FROM matable**

- **SELECT :** en langage SQL, le premier mot indique quel type d'opération doit faire MySQL. SELECT demande à MySQL d'afficher ce que contient une table.
- *** :** indique quels champs MySQL doit récupérer dans la table. * = tous les champs ; pour des champs particuliers : *SELECT champ1, champ2 FROM matable*
- **FROM :** c'est un mot de liaison. Il se traduit par "dans". FROM fait la liaison entre le nom des champs et le nom de la table.
- **matable :** c'est le nom de la table dans laquelle il faut aller chercher.

Afficher les données

Le problème, c'est que $reponse contient quelque chose d'inexploitable. La table pouvant contenir d'innombrables champs avec une multitude d'entrées!

Il faudrait pouvoir ranger les résultats dans une tableau…. **Le ARRAY !**

PHP dispose d'une fonction qui permet d'effectuer cette tâche : elle crée un array à partir de la variable $reponse : **mysql_fetch_array**

Pour créér cet array il suffit d'utiliser :

```
$donnees = mysql_fetch_array($reponse)
```

Le fait d'utiliser la fonction mysql_fetch_array fait passer la table à l'entrée suivante.

Si la table contient trois champs : champ1, champ2 et champ3, on pourrait les afficher :

- le contenu du champ n°1 : `<? echo $donnees['champ1']; ?>`
- le contenu du champ n°2 : `<? echo $donnees['champ2']; ?>`
- le contenu du champ n°3 : `<? echo $donnees['champ2']; ?>`

En utilisant un WHILE, la boucle d'affichage serait exécutée tant qu'il y a des données dans la table.

Afficher les données : Exemple

```
<?
mysql_connect("localhost", "monlogin", "monmotdepasse"); // Connexion à MySQL
mysql_select_db("mabase"); // Sélection de la base mabase

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


// Avec cette boucle, on liste uniquement le champ1 de toutes les entrées :

while ($donnees = mysql_fetch_array($reponse) )
{
echo $donnees[‘champ1'];
echo "<br />";
}

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

Les critères de sélection

Heureusement, il est possible de trier et d'ordonner les données renvoyées par MySQL à l'aide des éléments : WHERE, ORDER BY et LIMIT.

- **SELECT * FROM matable WHERE nomduchamp='jackpot'**

Traduction : "*Sélectionner tous les champs de la table matable lorsque le champ nomduchamp est égal à jackpot.*

- **SELECT * FROM matable ORDER BY nomduchamp**

Traduction : "*Sélectionner tous les champs de matable, et afficher les résultats par ordre croissant du champ nomduchamp.*"

Ajouter DESC pour un tri décroissant.

- **SELECT * FROM matable LIMIT 0, 20**

Traduction : On indique tout d'abord à partir de quelle entrée on commence à lire la table.Ensuite, le deuxième nombre indique combien d'entrées on doit sélectionner. Ici : 20 entrées.

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

Les critères de sélection : Exemple

```
<?
mysql_connect("localhost", "monlogin", "monmotdepasse");
mysql_select_db("mabase");

// Sélectionner les 10 premières entrées de la table matable
// pour lesquel le champ1=« texte » et le champ2 > 10, triées selon le champ3

$reponse = mysql_query("SELECT * FROM matable WHERE champ1='texte' AND
champ2>10 ORDER BY champ3 LIMIT 0, 10");

echo "Voici les 10 entrées de la table avec les conditions souhaitées :<p>";

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

<? echo $donnees['champ1']; ?><br>

<?
}

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

Le comptage

Demandons à MySQL de compter le nombre d'entrée d'une table :

```php
<?
mysql_connect("localhost", "monlogin", "monmotdepasse");
mysql_select_db("mabase");

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

?>
```

Il y a <? echo $donnees['nbre_entrees']; ?> entrées dans la table « matable »!

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

Exercice

Vous disposez du fichier « table.sql » qui va créer une table « personnel ».

Cette table contient le carnet d'adresse du personnel d'une entreprise. Les champs sont le suivants :

id	nom	prenom	naissance	sexe	salaire
INT	TEXT	TEXT	DATE	TEXT	INT

1. Créer la table à partir de phpMyAdmin : outil SQL.
2. Créer une page qui permette d'afficher :

- le nombre de personnes travaillant dans cette entreprise,
- tous le personnel, du salaire le plus élevé au moins élevé,
- le pourcentage d'hommes et de femmes,
- le nombre et le nom des femmes gagnant moins de 1250€,
- le nombre et le nom des hommes ayant plus de 30 ans
- la moyenne d'âge de l'entreprise.

Ecrire des données

On souhaite ajouter un nouvel employé à l'entreprise, en PHP.

id	nom	prenom	naissance	sexe	salaire
1	Brasse	Jean	1976-01-27	M	1524
2	Proviste	Alain	1971-10-02	M	1253
…	…	…	…	…	…

La requête SQL est la suivante :

INSERT INTO personnel(ID, nom, prenom, naissance, sexe, salaire, commentaires) VALUES('', 'Impique', 'Jean-Paul', '1959-08-08', 'M', '963')

- garder dans VALUES l'ordre défini dans INSERT INTO,
- le premier champ ID est vide puisqu'en auto_increment,
- le nom des champs après matable est facultatif.

Modifier et supprimer des données

Pour modifier le champ d'une entrée, la requête est la suivante :

UPDATE personnel SET nom='Brasse-Huaire', salaire='1831' WHERE ID='3'

Mise à jour

Nom de la table

mot-clé qui détermine les champs à modifier

Indique quelle entrée subit la modification. Ici, c'est l'entrée où le champ ID vaut 3

Pour supprimer des données :

DELETE FROM personnel WHERE prenom='Jean'

Note : Ne pas préciser de champ parès WHERE équivaut à vider la table!

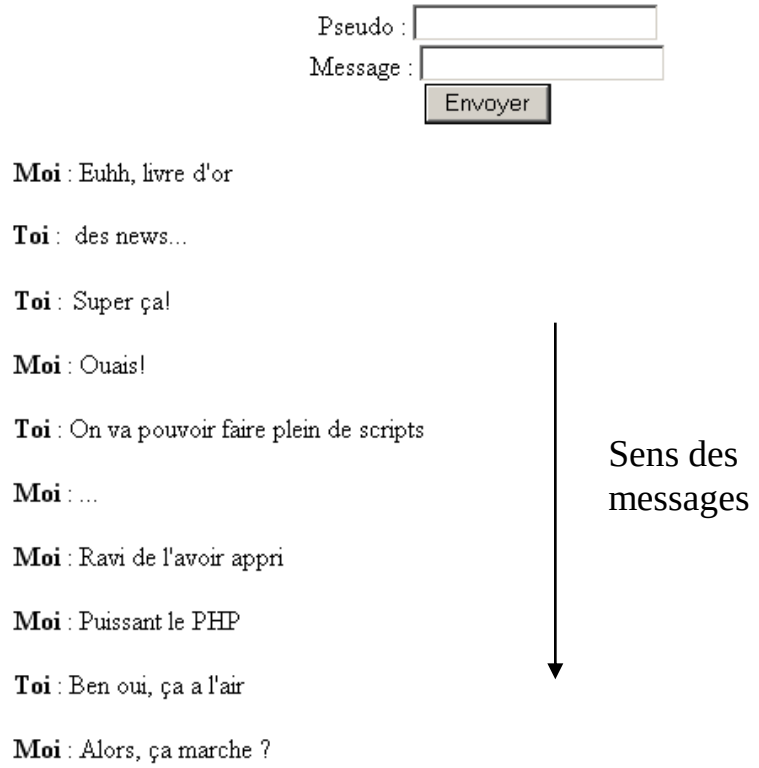
TP n°2 : Un T'chat (I)

Notions requises :

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

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

En-dessous, le script devra afficher les 10 derniers messages qui ont été enregistrés



TP n°2 : Un T'chat (II)

Ce qu'il faut :

- une table appelée « minichat »
- un champ ID de type INT en auto_increment,
- deux champs (PSEUDO et MESSAGE) de type VARCHAR (limité à 255 caractères)

Quelques indices :

- si un message a été entré, alors on enregistre dans la table et on affiche les 2 zones de texte et les 10 derniers messages. Champs inexistants (isset) ou vide (NULL) : on ne fait rien!
- le formulaire doit renvoyer sur la même page. Si votre page s'appelle "minichat.php", alors votre balise de formulaire sera :
<form action="minichat.php" method="post">

Facultatif : pour éviter d'écrire du code HTML dans la BDD, utiliser :

$message = htmlentities ($_POST['message']); (de même pour $pseudo)

A VOUS DE JOUER !