Table des matières

Ressources

Il existe de très nombreux sites dédiés au PHP ! Il faut au moins le manuel (notamment pour les fonctions) :
http://fr.php.net/manual/fr/

Les fichiers

Introduction

La manipulation de fichier se fait grâce à un **identifiant de fichier**. Les fonctions de base sont :

```
fopen($file [,$mode]) : ouverture du fichier identifié par son nom $file
et dans un mode $mode particulier, retourne un identificateur $fp de
fichier ou FALSE si échec

fopen($fp) : ferme le fichier identifié par le $fp

fgets($fp, $length) : lit une ligne de $length caractères au maximum

fputs($fp, $str) : écrit la chaîne $str dans le fichier identifié par $fp

fgetc($fp) : lit un caractère

feof($fp) : teste la fin du fichier
```

Pour obtenir des informations sur un fichier spécifique, on utilise généralement les fonctions suivantes :

```
file_exists($file) : indique si le fichier $file existe

filesize($file) : retourne la taille du fichier $file

filetype($file) : retourne le type du fichier $file

unlink($file) : détruit le fichier $file

copy($source, $dest) : copie le fichier $source vers $dest

readfile($file) : affiche le fichier $file

rename($old, $new) : renomme le fichier $old en $new
```

Pour ouvrir les fichiers, afin d'effectuer des opérations sur ceux-ci, on utilise la fonction `fopen()` :

```
fopen(string nom_du_fichier, string mode);
```

L'argument nom_du_fichier désigne le nom du fichier à ouvrir et l'argument mode désigne le mode d'ouverture (lecture, ajout, écriture..) :

Valeur	Opérations permises
a	Ouverture du fichier pour : écrire et créer le fichier. L'écriture commence a la fin du fichier
a+	Mêmes fonctions que ci-dessus sauf que la lecture est permise.
r	Ouverture d'un fichier en lecture seule.
r+	Mêmes fonctions que ci-dessus sauf qu'il est possible d'écrire dans le fichier. L'écriture commence au début du fichier.
w	Ouverture du fichier en écriture seulement. Création du fichier si celui-ci n'existe pas sauf que les données contenues précédemment sont effacées.
w+	Mêmes fonctions sauf qu'il est possible de lire dans le fichier.

Exemple d'ouverture d'un fichier

```
< ?php
if($ouverture = @fopen("fichier.txt", "r"))
{
        echo "L'ouverture du fichier est possible";
}
else
{
        echo "Ouverture du fichier impossible";
}
?>
```

La fonction `fopen()` renvoie true si l'ouverture est possible et false si elle ne l'est pas. Elle permet d'ouvrir des fichiers dont le chemin est relatif ou absolu. Elle permet aussi d'ouvrir des ressources avec les protocoles HTTP ou FTP.

Exemples

```
$fp = fopen("../docs/faq.txt", "r");
$fp = fopen("http://www.php.net/", "r");
$fp = fopen("ftp://user:password@cia.gov/", "w");
```

Affichage de fichiers

Pour afficher tout le contenu d'un fichier dans le navigateur, on utilise la fonction `fpassthru()` :

```
fpassthru(string pointeur);
```

Exemple d'envoi d'un fichier essai.txt au navigateur

```
<?php
$fichier = fopen("essai.txt","r");
fpassthru($fichier);
?>
```

Lecture de fichiers

Pour n'afficher qu'une partie d'un fichier, il existe plusieurs fonctions différentes.

La fonction `fgetc()` qui, elle, permet d'extraire le premier caractère du fichier :

```
fgetc(string pointeur);
```

Exemple

```
 < ?php $fichier = fopen("essai.txt","r");
$premier = fgetc($fichier);
echo "Premier Caractère : " . $premier;
fclose($fichier);
?>
```

La fonction `fgets()` permet, elle, d'extraire une chaîne d'une certaine longueur. Il faut aussi savoir que la fonction extrait la chaîne de la longueur précise définie en argument, plus un caractère et que la fonction s'arrête aux sauts de lignes :

```
fgets(string pointeur, string longueur);
```

Exemple

```
<?php
$fichier = fopen("essai.txt","r");
$premier = fgets($fichier, 10);
echo "Dix Premier Caractères : " . $premier;
fclose($fichier);
?>
```

La fonction `fread()` permet de lire une chaîne de caractère dans un fichier ouvert, jusqu'a la longueur indiquée en argument :

```
fread(string pointeur, string longueur);
```

Exemple

```
<?php
$fichier = fopen("essai.txt","r");
$premier = fread($fichier, 10);
echo "Dix Premiers Caractères : " . $premier;
fclose($fichier);
?>
```

Pour terminer, la fonction `file()` permet de mettre le contenu entier d'un fichier ouvert dans un tableau :

```
file(string fichier);
```

Exemple

```
 <?php
$premier = file("essai.txt");
echo "Première Ligne du fichier : " . $premier[0];
?>
```

Affichage du contenu d'un fichier :

```
<?php
$file = "fichier.txt" ;
if($fd = fopen($file, "r")) // ouverture du fichier en lecture
{       // tant que la fin de fichier n'est pas atteinte
        while(!feof($fd))
        {          $str .= fgets($fd, 1024); /* lecture jusqu'à fin de ligne ou
des 1024 premiers caractères */
        }
        fclose ($fd); // fermeture du fichier
        echo $str;    // affichage
}
else    die("Ouverture du fichier <b>$file</b> impossible.");
?>
```

Ecriture de fichiers

Pour écrire dans un fichier, on peut utiliser au choix la fonction `fwrite() ou fputs()` :

```
fwrite(string pointeur, string chaîne, (string longueur));
fputs(string pointeur, string chaîne, (string longueur));
```

Le paramètre longueur permet de limiter le nombre de caractères qui pourra être écrit dans le fichier. Arrivé a cette longueur, la fonction s'arrête.

Exemple

```
<?php
$fichier = fopen("essai.txt","w");
if(fwrite($fichier, "TEXTE A ECRIRE"))
{
echo "OK !";
}
else echo "Erreur";
fclose($fichier);
?>
```

Attention toutefois : le fichier doit bien être ouvert en mode écriture (w, a ou r+).

Opérations complémentaires

Pour vérifier si un fichier existe, on utilise la fonction `file_exists()` :

```
file_exists(string fichier);
```

L'argument fichier est le chemin permettant d'y accéder.

Exemple

```
<?php
if(file_exists("essai.txt"))
{
        echo "Fichier existant";
}
else    echo "Introuvable !";
?>
```

Déplacement : `copy(string fichier_depart, string fichier_destination);`

Renommer : `rename(string nom_depart, string nom_nouveau);`

Suppression : `unlink(string fichier);`

Liste des fonctions sur les systèmes de fichiers

Lien : http://fr.php.net/manual/fr/book.filesystem.php

basename — Sépare le nom du fichier et le nom du dossier
chgrp — Change le groupe d'un fichier
chmod — Change le mode du fichier
chown — Change le propriétaire du fichier
clearstatcache — Efface le cache de stat
copy — Copie un fichier
delete — Voir unlink ou unset
dirname — Renvoie le nom du dossier
disk_free_space — Renvoie l'espace disque disponible dans le répertoire
disk_total_space — Retourne la taille d'un dossier
diskfreespace — Alias de disk_free_space
fclose — Ferme un fichier
feof — Teste la fin du fichier
fflush — Envoie tout le contenu généré dans un fichier
fgetc — Lit un caractère dans un fichier
fgetcsv — Renvoie la ligne courante et cherche les champs CSV
fgets — Récupère la ligne courante sur laquelle se trouve le pointeur du fichier
fgetss — Renvoie la ligne courante du fichier et élimine les balises HTML
file_exists — Vérifie si un fichier ou un dossier existe
file_get_contents — Lit tout un fichier dans une chaîne
file_put_contents — Écrit un contenu dans un fichier
file — Lit le fichier et renvoie le résultat dans un tableau
fileatime — Renvoie la date à laquelle le fichier a été accédé pour la dernière fois
filectime — Renvoie la date de dernier accès à un inode
filegroup — Lire le nom du groupe
fileinode — Lit le numéro d'inode du fichier
filemtime — Lit la date de dernière modification du fichier
fileowner — Lit l'identifiant du propriétaire d'un fichier
fileperms — Lit les droits d'un fichier
filesize — Lit la taille d'un fichier
filetype — Retourne le type de fichier
flock — Verrouille le fichier
fnmatch — Repère un fichier à partir d'un masque de recherche
fopen — Ouvre un fichier ou une URL
fpassthru — Affiche le reste du fichier
fputcsv — Formate une ligne en CSV et l'écrit dans un fichier
fputs — Alias de fwrite
fread — Lecture du fichier en mode binaire
fscanf — Analyse un fichier en fonction d'un format
fseek — Modifie la position du pointeur de fichier
fstat — Lit les informations sur un fichier à partir d'un pointeur de fichier
ftell — Renvoie la position courant du pointeur de fichier
ftruncate — Tronque un fichier
fwrite — Écrit un fichier en mode binaire
glob — Recherche des chemins qui vérifient un masque

is_dir — Indique si le fichier est un dossier
is_executable — Indique si le fichier est exécutable
is_file — Indique si le fichier est un véritable fichier
is_link — Indique si le fichier est un lien symbolique
is_readable — Indique si un fichier est accessible en lecture
is_uploaded_file — Indique si le fichier a été téléchargé par HTTP POST
is_writable — Indique si un fichier est accessible en écriture
is_writeable — Alias de is_writable
lchgrp — Change l'appartenance du groupe d'un lien symbolique
lchown — Change le propriétaire d'un lien symbolique
link — Crée un lien
linkinfo — Renvoie les informations d'un lien
lstat — Retourne les informations sur un fichier ou un lien symbolique
mkdir — Crée un dossier
move_uploaded_file — Déplace un fichier téléchargé
parse_ini_file — Analyse un fichier de configuration
parse_ini_string — Analyse une chaîne de configuration
pathinfo — Retourne des informations sur un chemin système
pclose — Ferme un processus de pointeur de fichier
popen — Crée un processus de pointeur de fichier
readfile — Affiche un fichier
readlink — Renvoie le contenu d'un lien symbolique
realpath — Retourne le chemin canonique absolu
rename — Renomme un fichier ou un dossier
rewind — Replace le pointeur de fichier au début
rmdir — Efface un dossier
set_file_buffer — Alias de stream_set_write_buffer
stat — Renvoie les informations à propos d'un fichier
symlink — Crée un lien symbolique
tempnam — Crée un fichier avec un nom unique
tmpfile — Crée un fichier temporaire
touch — Modifie la date de modification et de dernier accès d'un fichier
umask — Change le "umask" courant
unlink — Efface un fichier

Accès aux répertoires

Il est possible de parcourir les répertoires grâce à ces quelques fonctions :

chdir($str) : Change le dossier courant en **$str**. Retourne TRUE si succès, sinon FALSE.
getcwd() : Retourne le nom du dossier courant (en format chaîne de caractères).
opendir($str) : Ouvre le dossier **$str**, et récupère un pointeur **$d** dessus si succès, FALSE sinon et génère alors une erreur.
closedir($d) : Ferme le pointeur de dossier **$d**.
readdir($d) : Lit une entrée du dossier identifié par **$d**. C'est-à-dire retourne un nom de fichier de la liste des fichiers du dossier pointé. Les fichiers ne sont pas triés. Ou bien retourne FALSE s'il n'y a plus de fichier.
rewinddir($d) : Retourne à la première entrée du dossier identifié par **$d**.

Affichage des noms de fichiers contenus dans le répertoire courant :

```php
<?php
// ouverture du dossier
if ($dir = opendir('.'))
{
          // lecture d'une entrée
         while($file = readdir($dir))
         {
                    echo " $file<br /> ";// affichage du nom de fichier
         }
         closedir($dir);                    // fermeture du dossier
}
?>
```

Remarques :

- **$dir** est un pointeur vers la ressource dossier
- **$file** est une chaîne de caractères qui prend pour valeur chacun des noms de fichiers retournés par **readdir()**

La classe dir

Il existe un autre moyen d'accéder aux dossiers : l'utilisation de la pseudo-classe **dir**.

En voici les attributs :
handle : valeur du pointeur
path : nom du dossier

En voici les méthodes :
read() : équivalent à **readdir($d)**
close() : équivalent à **closedir($d)**

Constructeur :
dir($str) : retourne un objet **dir** et ouvre le dossier **$str**

Exemple équivalent au précédent :

```php
<?php
$d = dir('.');     // ouverture du dossier courant
echo "Pointeur: ".$d->handle."<br />";
echo "Chemin: ".$d->path."<br />";
while($entry = $d->read())   // lecture d'une entrée
{
        echo $entry."<br />";
}
$d->close();                              // fermeture du dossier
?>
```

Séquence 1 : les fichiers

Objectifs

S'initier à la manipulation des fichiers à partir du langage PHP.

Moyens disponibles

Le logiciel ***Qanta+*** ou l'éditeur de texte ***vim ou kwrite*** pour la réalisation de la page HTML/PHP ou tout autre éditeur
Un navigateur (***Konqueror***, ***Mozilla***, ***Firefox*** ou ***Opéra***) et un serveur Web (***Apache***)
Le manuel sur les fonctions PHP
Les annexes 1 et 2

Présentation de la problématique

Dans le cadre d'un développement d'un site web spécialisé dans la mesure industrielle, vous participez à la réalisation d'un script en langage PHP.

L'acquisition de mesures de température (capteur pt100) va mettre dans un fichier texte une valeur toutes les minutes. Ces séries de mesures peuvent comporter des mesures incohérentes (lire l'annexe 1). Après traitement, on ne conservera que la médiane (et non la moyenne) de ces séries de mesures. La valeur médiane (lire l'annexe 2) est la valeur qui se trouve au milieu d'un ensemble de nombres triés. Si cet ensemble contient un nombre pair de nombres, la médiane sera alors la moyenne des deux nombres du milieu.

Manipulations

1 . Ecrire le script `f_mediane.php` qui permet de calculer et d'écrire dans un fichier `mediane_<fichier de mesures>.txt` la mesure médiane d'une série de mesures lues dans un fichier passé en argument dans l'URL du script. L'appel du script sera du type : `f_mediane.php?mesure=<fichier de mesures>`

Contraintes :

- Deux fichiers de mesures sont disponibles sur le serveur. Le script doit fonctionner correctement pour ces deux fichiers (nombre pair et impair de mesures).
- Le calcul de la mediane se fera dans une fonction CalculerMediane(). Les mesures doivent être préalablement triées (la fonction de tri est fournie sur le serveur).
- Les fichiers `mediane_<fichier de mesures>.txt` seront créés dans un répertoire `fichiers_mediane` à la racine du script. Ce répertoire doit avoir les droits d'écriture pour les autres (*other*).

Base de données MySQL

Introduction

Parmi les nombreux atouts du langage PHP, un des plus connus est son interfaçage avec la majorité des bases de données du marché.

Parmi les plus connues, on peut citer : **MySQL**, **PostgreSQL**, Oracle, Ingres, Interbase, Informix, Microsoft SQL Server, mSQL, Sybase, FrontBase, dBase, etc …

La base de donnée la plus utilisée avec PHP est sans aucun doute : MySQL, un SGDBR GPL.

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

SGBDR = Système de Gestion de Base de Données Relationnelle

Remarque : cette partie suppose connue les principes des bases de données relationnelles.

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. Il est disponible sur : http://sourceforge.net/projects/phpmyadmin/ et http://www.phpmyadmin.net.

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

Plus d'informations sont disponibles à http://www.mysql.com/.

La documentation de MySQL est disponibles à http://www.mysql.com/documentation/, ainsi qu'en français chez nexen : http://dev.nexen.net/docs/mysql/.

PHP fournit un grand choix de fonctions permettant de manipuler les bases de données. Toutefois, parmi celles-ci quatre fonctions sont essentielles :

- La fonction de connexion au serveur
- La fonction de choix de la base de données
- La fonction de requête
- La fonction de déconnexion

Avec le SGBD *MySQL*, ces fonctions sont les suivantes :

- `mysql_connect`
- `mysql_select_db`
- `mysql_query`
- `mysql_close`

Evidemment, il faudra traiter le résultat de la requête effectuée et donc transformer le résultat d'un ligne soit sous forme de variables, de tableau, de tableau associatif, d'objets.

Connexion

Pour se connecter à une base de donnée en php, il faut spécifier un nom de serveur, un nom d'utilisateur, un mot de passe et un nom de base.

Les fonctions de connexion :

- **mysql_connect($server,$user,$password)** : permet de se connecter au serveur **$server** en tant qu'utilisateur **$user** avec le mot de passe **$password**, retourne l'identifiant de connexion si succès, FALSE sinon
- **mysql_select_db($base[,$id])** : permet de choisir la base **$base**, retourne TRUE en cas de succès, sinon FALSE
- **mysql_close([$id])** : permet de fermer la connexion
- **mysql_pconnect()** : idem que **mysql_connect()** sauf que la connexion est persistante, il n'y a donc pas besoin de rouvrir la connexion à chaque script qui travaille sur la même base.

Remarque : les identifiants de connexion ne sont pas nécessaires si on ne se connecte qu'à une seule base à la fois, ils permettent seulement de lever toute ambiguïté en cas de connexions multiples.

Exemple simple de connexion

```
if($db = mysql_connect("localhost", "root", "password"))
{
        $id_db = mysql_select_db("test");
        if(!$id_db) die("Echec de connexion à la base !");
}
else die("Echec de connexion au serveur de base de données");
// code du script
// ...
mysql_close($db);
```

En pratique, on constate les comportements suivants :

- Utilisation de variables globales de connexion **($user, $passwd, $host et $base** ou dans un tableau) qui seront placées dans un fichier du style config.inc.php et inclus dans chaque script qui en a besoin par un require() ou un include().

- Intégration du code de connexion ou plus largement d'interfaçage à la base de données dans un fichier à inclure (par exemple mysql.inc.php)

- Utilisation d'un API de haut niveau pour notamment le portage vers d'autres bases de données (pilote ODBC, couche DAO ou des bibliothèques comme PEAR::DB). Remarque : le plus souvent sous forme de classes.

- Utilisation des connexions persistantes : évite d'avoir à rouvrir une connexion dans chaque script. Les connexions sont automatiquement fermées au bout d'un certain temps en cas d'absence de toute activité…

Interrogation

Pour envoyer une requête à une base de donnée, on utilise la fonction : **mysql_query($str)** qui prend pour paramètre une chaîne de caractères qui contient la requête écrite en SQL et retourne un identificateur de résultat ou FALSE en cas d'échec.

Les requêtes les plus couramment utilisées sont : **CREATE** (création d'une table), **SELECT** (sélection), **INSERT** (insertion), **UPDATE** (mise à jour des données), **DELETE** (suppression), **ALTER** (modification d'une table), etc ...

Exemple : une table revues avec deux champs revueid et nom

```
$result = mysql_query("SELECT * FROM revues");
// ou
$result = mysql_query("SELECT * FROM revues WHERE revueid='".$id."'");
```

L'identificateur de résultat **$result** permettra à d'autres fonctions d'extraire ligne par ligne les données retournées par le serveur.

Une fois la requête effectuée et l'identificateur de résultat obtenu, il ne reste plus qu'à extraire les données retournées par le serveur.

Extraire les données

Sous MySQL (comme pour beaucoup de SGBD), le traitement des résultats d'une requête se fait ligne par ligne. Une boucle permettra de recueillir chacune des lignes à partir de l'identifiant de résultat.

Une ligne contient une ou plusieurs valeurs correspondants aux différents attributs retournés par la requête. Ainsi, une ligne de résultat pourra être sous la forme de variables, d'un tableau, d'un tableau associatif, ou d'un objet.

mysql_fetch_row($result) : retourne une ligne de résultat sous la forme d'un tableau. Les éléments du tableau étant les valeurs des attributs de la ligne. Retourne FALSE s'il n'y a plus aucune ligne.

Exemple

```
$requete = "SELECT * FROM revues";
if($result = mysql_query($requete))
{       while($ligne = mysql_fetch_row($result))
        {         $revueid = $ligne[0];
                  $nom = $ligne[1];
                  echo "$revueid -&gt; $nom<br />";
        }
}
```

Ici, on accède aux valeurs de la ligne par leur **indice** dans le tableau.

Une pratique courante est d'utiliser la fonction **list()** pour avoir automatiquement les champs sous forme de variables :

```
while(list($revueid, $nom) = mysql_fetch_row($result))
{
        echo "$revueid -&gt; $nom<br />";
}
```

mysql_fetch_array($result) : retourne un tableau associatif. Les clés étant les noms des attributs et leurs valeurs associées leurs valeurs respectives. Retourne FALSE s'il n'y a plus aucune ligne.

Exemple

```
$requete = "SELECT * FROM revues";
if($result = mysql_query($requete))
{
        while($ligne = mysql_fetch_array($result))
        {
                $revueid = $ligne["revueid"];
                $nom = $ligne["nom"];
                echo "$revueid -&gt; $nom<br />";
        }
}
```

Ici, on accède aux valeurs de la ligne par **la clé** (attribut) dans le tableau associatif.

mysql_fetch_object($result) : retourne un objet. Les attributs de l'objet correspondent à ceux de la ligne de résultat. Et les valeurs des attributs de l'objet correspondent à ceux de la ligne de résultat.
Retourne FALSE s'il n'y a plus aucune ligne.

Exemple

```
$requete = "SELECT * FROM revues";
if($result = mysql_query($requete))
{
        while($ligne = mysql_fetch_object($result))
        {
                $revueid = $ligne->revueid;
                $nom = $ligne->nom;
                echo "$revueid -&gt; $nom<br />";
        }
}
```

Ici, on accède aux valeurs par leur **attribut** dans l'objet.

Fonctions utiles

Quelques fonctions supplémentaires très utiles :

- **mysql_free_result($result)** : efface de la mémoire du serveur les lignes de résultat de la requête identifiées par **$result**.
- **mysql_insert_id([$id])** : retourne l'identifiant d'un attribut clé primaire AUTO_INCREMENT de la dernière insertion.
- **mysql_num_fields($result)** : retourne le nombre d'attributs du résultats.
- **mysql_num_rows($result)** : retourne le nombre de lignes du résultats.

Généralement, on préfixe les appels aux fonctions mysql_ avec @ pour éviter les affichages dans le navigateur des messages de WARNING ou d'erreur (sauf en débuggage).

Par contre, il faut systématiquement tester et traiter les erreurs en provenance de la base de données. Pour cela, on pourra utiliser :

- **mysql_errno()** : retourne le numéro de message d'erreur de la dernière opération MySQL
- **mysql_error()** : retourne le texte associé à l'erreur générée lors de la dernière requête.

Exemple

```
mysql_connect("TF1"); // ajouter le if
echo mysql_errno().": ".mysql_error()."<br>";

mysql_select_db("StarAcademy"); // ajouter le if
echo mysql_errno().": ".mysql_error()."<br>";

$result = mysql_query("SELECT * FROM artistes"); // ajouter le if
echo mysql_errno().": ".mysql_error()."<br>";
```

Liste des fonctions pour MySQL

Lien : http://fr.php.net/manual/fr/ref.mysql.php

mysql_affected_rows -- Retourne le nombre de lignes affectées lors de la dernière requête SQL.
mysql_change_user -- Change le nom de session de l'utilisateur actif.
mysql_character_set_name -- Returns the name of the character set
mysql_close -- Ferme la connexion MySQL.
mysql_connect -- Ouvre une connexion à un serveur MySQL.
mysql_create_db -- Crée une base de données MySQL.
mysql_data_seek -- Déplace le pointeur interne de résultat.
mysql_db_name -- Lit les noms des bases de données
mysql_db_query -- Envoie une requête MySQL à un serveur MySQL.
mysql_drop_db -- Efface une base de données MySQL.
mysql_errno -- Retourne le numéro de message d'erreur de la dernière opération MySQL.
mysql_error -- Retourne le texte associé avec l'erreur générée lors de la dernière requête.

mysql_escape_string -- Protège une chaîne pour la passer à mysql_query.
mysql_fetch_array -- Retourne une ligne de résultat sous la forme d'un tableau associatif.
mysql_fetch_assoc -- Lit une ligne de résultats dans un tableau associatif
mysql_fetch_field -- Retourne les données enregistrées dans une colonne sous forme d'objet.
mysql_fetch_lengths -- Retourne la taille de chaque colonne d'une ligne de résultat.
mysql_fetch_object -- Retourne les lignes résultats sous la forme d'un objet.
mysql_fetch_row -- Retourne une ligne de résultat sous la forme d'un tableau.
mysql_field_flags -- Retourne le sémaphore associé à la colonne spécifiée dans le résultat courant.
mysql_field_len -- Retourne la longueur du champs spécifié.
mysql_field_name -- Retourne le nom d'une colonne
mysql_field_seek -- Déplace le pointeur de résultat
mysql_field_table -- Retourne le nom de la table où se trouve une colonne
mysql_field_type -- Retourne le type de la colonne spécifiée dans le résultat courant.
mysql_free_result -- Efface le résultat de la mémoire.
mysql_get_client_info -- Lit les informations sur le client MySQL
mysql_get_host_info -- Lit les informations sur l'hôte MySQL
mysql_get_proto_info -- Lit les informations sur le protocole MySQL
mysql_get_server_info -- Lit les informations sur le serveur MySQL
mysql_info -- Get information about the most recent query
mysql_insert_id -- Retourne l'identifiant généré par la dernière requête INSERT.
mysql_list_dbs -- Liste les bases de données disponibles sur le serveur MySQL.
mysql_list_fields -- Liste les champs du résultat MySQL.
mysql_list_processes -- List MySQL processes
mysql_list_tables -- Liste les tables d'une base de données.
mysql_num_fields -- Retourne le nombre de champs d'un résultat.
mysql_num_rows -- Retourne le nombre de lignes d'un résultat.
mysql_pconnect -- Ouvre une connexion persistante à un serveur MySQL.
mysql_ping -- Ping a server connection or reconnect if there is no connection
mysql_query -- Envoie une requête SQL à un serveur MySQL.
mysql_real_escape_string -- Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection.
mysql_result -- Retourne un champs d'un résultat.
mysql_select_db -- Sélectionne une base de données MySQL.
mysql_stat -- Get current system status
mysql_tablename -- Lit le nom de la table qui contient le champs spécifié.
mysql_thread_id -- Return the current thread id
mysql_unbuffered_query -- Exécute une requête SQL sans mobiliser les résultats

Séquence 2 : base de données MySQL

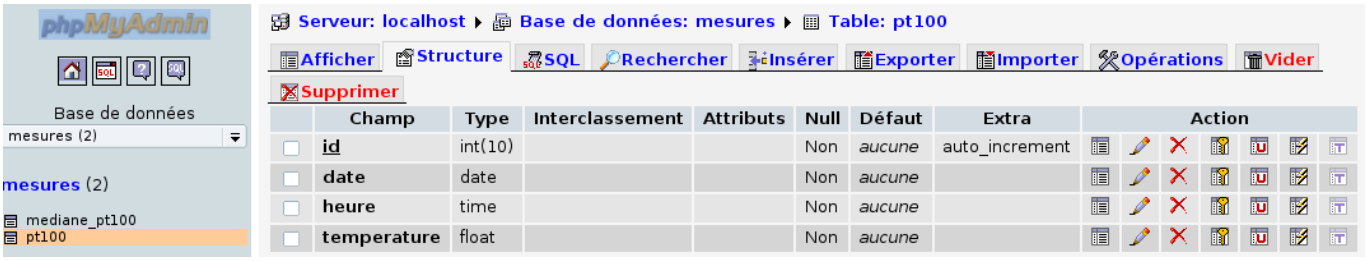
Objectifs

S'initier à l'utilisation des bases de données en langage PHP.

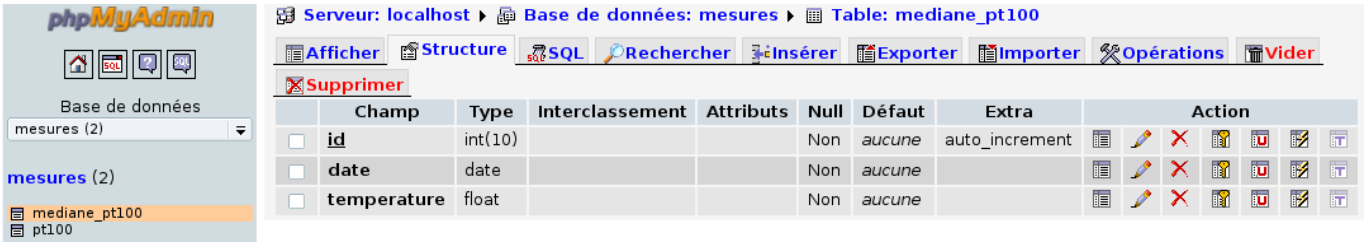
Remarques

On utilise le même contexte que la séquence précédente mais les mesures sont maintenant stockées dans une base de données **mesures**. Cette base de données contient deux tables :

- La table **pt100** contient les mesures de températures datées :



- La table **mediane_pt100** permet de conserver la valeur mediane (une fois calculée) d'une série de mesures datées :



Un fichier **mesures.sql** permet de créer cette base de données (avec des mesures déjà effectuées) sur une serveur MySQL. Ce fichier est fourni sur le serveur de la section.

Manipulations

1 . Ecrire le script `bd_mediane.php` qui permet de calculer et d'écrire dans une table MySQL `mediane_pt100` la mesure médiane d'une série de mesures lues dans une table MySQL `pt100` dont la date (format AAAA-MM-JJ) est passée en argument dans l'URL du script. L'appel du script sera du type : `bd_mediane.php?date=2009-09-08`

Contraintes :

- Deux séries de mesures sont disponibles dans la table `pt100` en fonction de la **date**. Le script doit fonctionner correctement pour ces deux séries (nombre pair et impair de mesures). Le champ *heure* de la table `pt100` n'est pas utilisé dans cette séquence.
- Le calcul de la mediane se fera dans une fonction CalculerMediane(). Les mesures doivent être préalablement triées (la fonction de tri est fournie sur le serveur).
- La table `mediane_pt100` n'accepte qu'une seule mediane par date de mesures. On se limitera à une insertion unique mais une gestion plus fine devrait être faite.

Aller plus loin

Modifier le script précédent afin d'afficher un tableau principal

Mesures pt100

Date	Mediane
2009-09-08 (lien vers `bd_mediane.php?date=2009-09-08`)	35.10 (affiche la valeur si elle est présente dans la table `mediane_pt100`)
2009-09-09 (lien vers `bd_mediane.php?date=2009-09-09`)	(ou rien si la médiane n'a pas encore été calculée)

Annexe 1 : les mesures dans l'industrie

Dans le cas des mesures dans l'industrie, on considérer trois sources d'erreur (source wikipedia) :

- la précision de la mesure ou l'incertitude ;
- la dispersion statistique ;
- l'erreur systématique.

L'erreur totale étant la somme des trois sources d'erreurs.

Si l'on fait la comparaison avec des flèches que l'on tire sur une cible :

- la précision de mesure désigne la taille de la pointe de la flèche ;
- la dispersion statistique désigne le fait que les flèches sont proches les unes des autres, ou bien au contraire éparpillées sur la cible ;
- l'erreur systématique indique si les flèches visaient bien le centre, ou bien un autre point de la cible.

Pour la dispersion statistique, on estime que si l'on mesure plusieurs fois le même phénomène avec un appareil suffisamment précis, on obtiendra chaque fois un résultat différent. Ceci est dû à des phénomènes perturbateurs ou, pour les mesures extrêmement précises, à la nature aléatoire du phénomène.

Parmi les phénomènes perturbateurs, on peut dénombrer :

- l'erreur d'échantillonnage : c'est lorsque l'on prélève un échantillon qui n'est pas représentatif de ce que l'on veut mesurer ; le résultat dépend alors de la manière dont on choisit l'échantillon;
- l'erreur de préparation : l'échantillon s'altère pendant le transport, le stockage ou la manipulation (pollution, dégradation, transformation physique ou chimique) ;
- la stabilité de l'appareil : celui-ci peut être sensible aux variations de température, de tension d'alimentation électrique, aux vibrations, aux perturbations électromagnétiques des appareils environnants ou bien présenter un défaut de conception ou une usure (bruit de fond électronique, pièce instable ...)

Le calcul d'erreur, ou calcul d'incertitudes est un ensemble de techniques permettant d'estimer l'erreur faite sur un résultat numérique, à partir des incertitudes ou des erreurs faites sur les mesures qui ont conduit à ce résultat. L'erreur de mesure détermine la sensibilité (capacité à sélectionner les bons « candidats ») et la sélectivité (capacité à éliminer les mauvais « candidats ») d'une méthode.

Annexe 2 : moyenne vs médiane

L'utilisation de la médiane à la place de la moyenne est fréquent pour les mesures dans l'industrie.

Exemple
Soit deux listes de mesures provenant d'un capteur sur une période de 1mn30s :
L1 : 35,53°C, 35,23°C, 35,10°C, 35,02°C, 34,45°C
L2 : 35,53°C, 35,23°C, 35,10°C, 34,45°C, 12,22°C

Remarque : la série L2 possède une mesure incohérente.

La mesure incohérente dans la série L2 est **12,22°C** (l'écart-type est de 10.23°C pour cette série de mesures).

N° Liste	L1	L2
Moyenne	35,066°C	30,506°C
Valeur médiane	35,10°C	35,10°C

Dans la série L2, la mesure incohérente (12,22°C) serait pris en compte dans la moyenne et fausserait donc le résultat obtenu (la moyenne sans cette valeur est de 35,07°C contre 30,50°C si on en teint compte).

Ici, l'utilisation de la médiane comme technique de sélectivité permet d'atténuer ce type de problème.

L'utilisation de la valeur médiane est donc préférable à la valeur moyenne. Cependant, son utilisation implique le tri des données au préalable.