TP PHP

Sommaire

Les objectifs de ce tp sont d'être capable de réaliser des scripts serveurs pour un site web en respectant les bonnes pratiques.

Il existe de très nombreux sites dédiés au PHP ! Il faut au moins l'accès au manuel (notamment pour les fonctions) : www.php.net/manual/fr/index.php

Séquence n°1 : les bases

Client/Serveur HTTP

Pour réaliser un développement PHP, il vous faut la chaîne complète client/serveur HTTP. Plusieurs solutions s'offrent à vous :

- le serveur est présent en local sur votre machine de développement (*localhost*). Le plus souvent sous Linux, la racine des documents du serveur se trouvent en `/var/www/html/` . Et l'accès par le navigateur se fait à l'adresse : http ://localhost/ ou http ://votre-adresse-ip/
- le serveur est présent sur l'intranet de votre structure de développement (entreprise, école, université, domicile, ...). Le serveur de la section est configuré pour un accès pour chaque compte. La racine se trouve dans votre répertoire personnel `$HOME/public_html/` et l'accès client se fait par exemple à l'adresse : http ://intranet/~$LOGIN/ ou http ://192.168.52.85/~$LOGIN/
- le serveur est présent sur l'internet, le plus souvent chez un hébergeur. Dans ce cas, il faut transférer les documents de votre poste de développement vers le serveur Internet (par FTP par exemple).

LAMP est un acronyme désignant un ensemble de logiciels libres permettant de construire des serveurs de sites web. L'acronyme original se réfère aux logiciels suivants : **L**inux (l'OS GNU/Linux), **A**pache (le serveur Web), **M**ySQL (le serveur de base de données) et **P**HP (le langage de script). Il existe aussi une architecture **WAMP** utilisée pour développer des sites web sur une machine *Windows*.

Exercice n°1.1 : généreration de code HTML

L'objectif de ce premier exercice est de vérifier que la chaîne client/serveur fonctionne correctement en générant du code HTML à partir d'un script PHP.

Question 1. Créer le script à partir du code source fourni ci-dessous. Quelle est l'extension à donner à ce script ?

```
<html>
  <head>
    <title>Exercice PHP 1.1</title>
  </head>
  <body>
   <?php
     echo "Hello world !";
   ?>
  </body>
</html>
```

Exercice 1.1.a

Question 2. Tester dans un navigateur. Afficher le code source de la page à partir du navigateur. Pourquoi le code PHP n'apparaît-il pas ?

On peut tout à fait générer du code HTML avec le script PHP, comme ceci par exemple :

```
<?php
$nom = "Robert"; // en PHP, toutes les variables sont préfixées par un dollar ($)
echo "<p>Bonjour $nom !</p>"; // des balises HTML !
?>
```

Exercice 1.1.b

Question 3. Vérifier l'exécution de ce script dans un navigateur et le code source de la page ainsi générée. Est-ce que le code HTML généré par le script PHP (balise `P` ici) a-t-il été envoyé au client ?

Une autre approche pour générer du code HTML est de concaténer l'intégralité du code HTML dans une variable et de réaliser un simple `echo` de cette variable en fin de script.

```php
<?php
$out = "<html>\n";

// On concatène avec l'opérateur . en PHP
$out .= "\t<head><title>Exercice PHP 1.1.c</title></head>\n";
$out .= "\t<body><p>Du code HTML généré par PHP</p></body>\n";
$out .= "</html>";

// On affiche le contenu de la variable et donc de la page
echo $out;
?>
```

Exercice 1.1.c

Il existe d'autres techniques plus élaborées pour générer du code HTML à partir de PHP : *buffer* de sortie (ob), *template*, DOM, ...

Question 4. Vérifier l'exécution de ce script dans un navigateur et le code source de la page ainsi générée. Quelle est l'utilité des "\$n" et des "\$t" générés par le script PHP ?

Exercice n°1.2 : des informations sur le serveur

Le langage PHP vous fournit la fonction `phpinfo()` qui affiche de nombreuses informations sur la configuration de PHP : options de compilation, extensions, version, informations sur le serveur, et l'environnement (lorsqu'il est compilé comme module), environnement PHP, informations sur le système, chemins, valeurs générales et locales de configuration, en-têtes HTTP et la licence PHP.

```php
<html>
  <head>
    <title>phpinfo.php</title>
  </head>
  <body>
   <?php
     phpinfo();
   ?>
  </body>
</html>
```

phpinfo.php

Question 5. Créer le script `phpinfo.php` à partir du code source fourni ci-dessus. Tester dans un navigateur et identifier la version du moteur PHP que vous utilisez.

Conservez ce script car `phpinfo()` est un bon outil de débogage qui permet d'afficher le contenu de toutes les variables **EGPCS** (Environnement, GET, POST, Cookie, Serveur) que l'on utilise tout le temps en PHP.

Question 6. À partir des informations sur les variables PHP affichées par la fonction `phpinfo()`, donner les noms des variables contenant l'adresse IP du serveur et du client.

Les variables PHP affichées par la fonction `phpinfo()` sont appelées des **variables super-globales** en PHP. Depuis la version 4 de PHP, ces variables pré-définies sont stockées dans des **tableaux superglobaux** tels que `$_GET`, `$_POST` et `$_SERVER`, etc. Pour plus d'informations, lisez la section *superglobals* du manuel PHP : www.php.net/manual/fr/language.variables.predefined.php.

Question 7. Écrire un script PHP qui permet d'obtenir l'affichage suivant :

```
Adresse IP du serveur : 192.168.52.1
Votre adresse IP est : 192.168.52.2
Et la signature de votre navigateur est : Mozilla/5.0 (X11; Linux i686; rv:10.0.5) Gecko
    /20100101 Firefox/10.0.5
```

Les adresses IP affichées ci-dessus sont fournies à titre d'exemple.

Exercice n°1.3 : débogage de variables

Comme d'autres langages de script, PHP utilise un **typage dynamique faible** pour ses variables. De ce fait, PHP ne nécessite pas de déclararation de type ni d'initialisation pour manipuler des variables. Cette spécificité affectera la sécurité de vos scripts.

Le langage vous fournit plusieurs fonctions pour déboguer les variables à l'exécution :

- `print_r()` affiche des informations à propos d'une variable, de manière à ce qu'elle soit lisible.
- `var_dump()` (et `var_export()`) affiche les informations structurées d'une variable, y compris son type et sa valeur. Les tableaux et les objets sont explorés récursivement, avec des indentations, pour mettre en valeur leur structure.

```php
<?php
$booleen = true; // un booléen
$nbr_i = 10; //un nombre entier
$nbr_r = 3.141; //un nombre réel
$str = "hello"; //une chaîne de caractèrestring
echo "Quelques variables affichées avec var_dump() :<br />";
echo "<pre>"; var_dump($booleen); var_dump($nbr_i); var_dump($nbr_r); var_dump($str); echo "
    </pre>"; echo "<br />";

$jour = array('dimanche', 'lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi'); // 'dimanche'
    est l'index 0 de ce tableau
$jour[6] = "samedi"; // j'avais oublié samedi !
echo "Un tableau avec print_r() :<br />";
echo "<pre>"; print_r($jour); echo "</pre>";
print_r($jour);
echo "<br /><br />";
echo "Un tableau avec var_dump() :<br />";
echo "<pre>"; var_dump($jour); echo "</pre>";
var_dump($jour);
?>
```

Exercice 1.3.a


```
Quelques variables affichées avec var_dump() :
bool(true)
int(10)
float(3.141)
string(5) "hello"

Un tableau avec print_r() :
Array
(
    [0] => dimanche
    [1] => lundi
    [2] => mardi
    [3] => mercredi
    [4] => jeudi
    [5] => vendredi
    [6] => samedi
)

Array ( [0] => dimanche [1] => lundi [2] => mardi [3] => mercredi [4] => jeudi [5] =>
    vendredi [6] => samedi )

Un tableau avec var_dump() :
array(7) {
  [0]=>
  string(8) "dimanche"
  [1]=>
  string(5) "lundi"
  [2]=>
  string(5) "mardi"
  [3]=>
  string(8) "mercredi"
  [4]=>
  string(5) "jeudi"
  [5]=>
  string(8) "vendredi"
  [6]=>
  string(6) "samedi"
}

array(7) { [0]=> string(8) "dimanche" [1]=> string(5) "lundi" [2]=> string(5) "mardi" [3]=>
    string(8) "mercredi" [4]=> string(5) "jeudi" [5]=> string(8) "vendredi" [6]=> string(6)
    "samedi" }
```

Question 8. Testez le script ci-dessus. Qu'indiquent les valeurs affichées entre parenthèses pour `array` et `string` ?

PHP étant un langage avec un typage souple des variables, il vous faudra vous méfier des tests d'égalités entre variables de type différent.

```php
<?php
$i = 10; //un nombre entier
$a = "10"; //une chaîne de caractère
echo "Deux variables qui ont la même valeur mais pas le même type :<br />";
```



```php
echo "<pre>\$i -&gt; "; var_dump($i); echo "<pre>\$a -&gt; "; var_dump($a); echo "</pre>";
    echo "<br />";

// Teste l'égalité de valeur entre deux variables :
if($i == $a)
     echo "Test valeur : égal !<br />";
else  echo "Test valeur : pas égal !<br />";

// Teste l'égalité de valeur et de type entre deux variables :
if($i === $a)
     echo "Test valeur et type : égal !<br />";
else  echo "Test valeur et type : pas égal !<br />";
?>
```

Exercice 1.3.aa

Ce qui donnera :

```
Deux variables qui ont la même valeur mais pas le même type :
$i -> int(10)
$a -> string(2) "10"

Test valeur : égal !
Test valeur et type : pas égal !
```

Les tableaux PHP peuvent contenir des **clés** de type entier (`integer`) et chaîne de caractères (`string`) en même temps. Dans certains langages, ce type de tableaux sont nommés tableau associatif (ou encore hashage ou *hash*) et permettent une association clé-élément.

Les tableaux PHP peuvent aussi être multidimensionnel.

```php
<?php
echo "Un tableau multidimensionnel affiché avec var_export() :<br />";
$a = array(array("a", "b", "c"),array("d", "e", "f"),array("g", "h", "i"));
echo "<pre>"; var_export($a); echo "</pre>";

$jour = array('dimanche', 'lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi', 'samedi');
$date["jour"] = $jour;
$date["mois"] = array('janvier', 'février', 'mars', 'avril', 'mai', 'juin', 'juillet','aout'
    ,'septembre','octobre','novembre','décembre');

echo "Un tableau associatif affiché avec var_dump() :<br />";
echo "<pre>"; var_dump($date); echo "</pre>";
echo "<br />";

// Une exploitation de ce tableau
$j = date("w");
$m = date("n");

$message = "Le ".$date["jour"][$j]." ".date("j")." ".$date["mois"][$m-1]."<br />";
echo "Utilisation d'un tableau associatif :<br />";
echo $message;
?>
```

Exercice 1.3.b

Question 9. Testez le script ci-dessus. En utilisant le manuel PHP pour la fonction `date()`, que permettent de faire les paramètres passés en argument de cette fonction ?

Question 10. Écrire un script PHP qui permet d'obtenir l'affichage (en français) suivant :

```
Bonjour,
on est le samedi 16 février 2013 et il est 11:51:36
```

PHP possède un grand nombre de fonctions ! Il vous faut donc constamment utiliser le manuel PHP. Pour les fonctions concernant les dates et heures : www.php.net/manual/en/ref.datetime.php. Par exemple pour formater des dates dans d'autres langues, utilisez les fonctions `setlocale()` et `strftime()`.

Exercice n°1.4 : les fonctions

En PHP, Une fonction peut être définie en utilisant la syntaxe ci-dessous et s'utilise de la même manière qu'en C/C++ :

```php
<?php
// Définiton d'une fonction
function foo($arg1, $arg2="b")
{
   $retval = 0;
   echo "Je suis la fonction foo()." . "<br />\n";
   $numargs = func_num_args();
   echo "Nombre d'arguments au moment de l'appel : $numargs" . "<br />\n";
   $arg_list = func_get_args();
   for ($i = 0; $i < $numargs; $i++)
   {
      echo "L'argument n°$i est : " . $arg_list[$i] . "<br />\n";
   }
   if($numargs != 2)
      echo "L'argument n°1 est : " . $arg2 . " (par défaut ici)<br />\n";
   echo "<br />\n";

   // je suis capable de retourner un résultat
   return $retval; // tous les types de variables peuvent être renvoyés, tableaux et objets
       compris.
}

// Appel d'une fonction (utilisation d'un argument par défaut)
Foo("a"); // insensible à la casse !

// Appel d'une fonction et récupération de la valeur retournée
$res = foo("x", "y");
echo "La fonction foo() a retourné $res" . "<br /><br />\n";

// Le PHP est riche en fonctions utiles !
if (function_exists('foo'))
     echo "La fonction foo est disponible.<br />\n";
else  echo "La fonction foo n'est pas disponible.<br />\n";
?>
```

Exercice 1.4.a

PHP ne supporte pas le surchargement de fonction, ni la destruction ou la redéfinition de fonctions déjà déclarées. Par contre, les valeurs par défaut d'arguments sont supportés. Les noms de fonctions sont insensibles à la casse.

Les variables **globales** déclarées dans un script sont visibles sur l'ensemble du script, mais pas directement au sein des fonctions. Pour utiliser une variable globale au sein d'une fonction, on doit le préciser à l'aide de l'instruction `global`, pour faire référence à la variable globale du même nom. On pourra alors accéder à cette variable par son nom ou directement à l'aide du tableau `$GLOBALS`.

Les variables **locales** sont déclarées et utilisées au sein d'une fonction.

Une variable déclarée à l'intérieur d'une fonction à l'aide de l'instruction `static` permet à une variable de garder sa valeur à chaque appel de la fonction. L'initialisation d'une variable **statique** se fait au début de la fonction et à chaque appel de la fonction dans le script elle gardera la valeur du dernier appel.

Les **constantes** sont définies grâce à la fonction `define()` et on conseille toujours d'utiliser des majuscules pour les noms de constante.

```php
<?php
define("MA_FONCTION", "carre()"); // une constante

function carre()
{
   static $cpt = 1; // une variable statique conserve sa valeur entre chaque appel
   global $val; // permet l'accès à la variable globale $val
   $calcul = 0; // une variable de portée locale

   echo "La fonction ". MA_FONCTION . " a été appelée $cpt fois." . "<br />\n";
   echo "Je vais calculer le carré de " . $GLOBALS["val"]. " ...<br />\n";
   $cpt++;
   $calcul = $val*$val;

   return $calcul;
}

for($val = 2; $val < 10; $val++)
{
   $c = carre();
   echo "Le carré de $val est $c" . "<br /><br />\n";
}
?>
```

Exercice 1.4.b

Par défaut, les arguments sont passés à la fonction **par valeur**. Ainsi changer la valeur d'un argument dans la fonction ne change pas sa valeur à l'extérieur de la fonction.

Si les fonctions doivent changer la valeur des arguments, il faut passer les arguments **par référence**. Pour cela, il faut ajouter un '`&`' devant l'argument dans la fonction.

```php
<?php
// &$a : passage par référence
// $b : passage par valeur
function raz(&$a, $b)
{
   $a = 0;
   $b = 0;
```



```php
}

$x = 2;
$y = 2;
echo "Avant : x=$x et y=$y <br />"; // Affiche : Avant : x=2 et y=2
raz($x, $y);
echo "Après : x=$x et y=$y <br />"; // Affiche : Après : x=0 et y=2
?>
```

Exercice 1.4.b

Question 11. Écrire un script qui affiche le nombre d'années bissextiles que vous avez vécu depuis votre naissance en indiquant lesquelles. Vous écrirez une fonction `estAnneeBissextile()` qui recevra en argument une année et qui retournera vrai (`true`) si l'année est bissextile et faux (`false`) sinon.

Exemple de résultat attendu :

```
Vous avez vécu 10 années bissextiles : 1968 1972 1976 1980 1984 1988 1992 1996 2000 2004
```

La règle des années bissextiles :
Le pape Grégoire XIII a mis au point en 1582 un calendrier, appelé encore aujourd'hui la calendrier grégorien. Il introduit les règles de calcul des années bissextiles : tous les 4 ans donc chaque année divisible par 4 (comme 1992 ou 1996). Mais ces règles ne suffisent pas. En effet, lorsqu'il s'agit de la première année d'un nouveau siècle (par exemple 2000, 1900 ou 2100), cette année doit être divisible non pas par 4 mais par 400. Ainsi 1700, 1800 et 1900 ne sont pas des années bissextiles.

Conclusion

Le code *php* se mélange au code HTML et s'exécute côté serveur. PHP est un langage de script libre principalement utilisé pour produire des pages web dynamiques.

Question 12. Proposer une définition de "pages web dynamiques".

Séquence n°2 : dialogue avec le client

Exercice n°2.1 : passage de paramètres dans l'URL

Cet exercice a pour objectif de montrer comment on récupère des données passées en paramètres de l'*url*. Cette technique est très utilisée dans la réalisation d'applications PHP.

Jusqu'à la version PHP 4.2.0, les paramètres passés dans l'*url* étaient automatiquement connus sous forme de variables globales du script destinataire. Mais depuis la version 4.2.0, ce n'est plus le cas à cause du changement de la valeur par défaut (auparavant à `On` et désormais à `Off`) du paramètre `register_globals` du fichier de configuration `php.ini` du serveur. Ce changement impose de recourir désormais aux tableaux dit superglobaux de PHP (`$_GET[]`, `$_POST[]`, etc ...). Ces variables superglobales sont accessibles de partout dans un script php (ne pas mettre `global`).

Le principe est le suivant :

```
paramètre id dans url : exemple.php?id=4

alors dans le script exemple.php : $_GET['id'] sera égal à 4
```

En phase d'apprentissage ou de débuggage, il est recommandé de faire un `var_dump($_GET);`

```php
<?php
// vérifie que le paramètre year n'est pas vide
if(!Empty($_GET["year"]))
{
   // récupère le paramètre year
   $year = $_GET["year"];

   // Attention les paramètres d'url sont passées sous forme de chaîne de caractères
   echo "<pre>Débogage variable year : "; var_dump($year); echo "</pre>";

   // vérifie que le paramètre year est valide
   if(ctype_digit($year)) // cf. is_numeric()
   {
      echo "est-ce-que $year est bissextile ?<br /><br />";
      // est-ce-que cette année est bissextile ?
      // TODO
   }
   else echo "Paramètre year invalide !<br /><br />";
}
else echo "Paramètre year manquant !<br /><br />";

echo "Essayez avec ces paramètres :<br />";
echo "<a href=\"exercice-2-1-a.php?year=\">vide</a><br />";
echo "<a href=\"exercice-2-1-a.php?year=2000\">2000</a><br />";
echo "<a href=\"exercice-2-1-a.php?year=2007\">2007</a><br />";
echo "<a href=\"exercice-2-1-a.php?year=year\">invalide</a><br />";
?>
```

Exercice 2.1.a

Question 13. Testez le script ci-dessus. Que permettent de faire les fonctions `Empty()` et `ctype_digit()` utilisées dans ce script ?

Question 14. Compléter le script ci-dessus afin qu'il affiche toutes les années depuis votre naissance sous forme de lien avec en paramètre l'année en question et qui affichera si cette année est bissextile ou non. Pour cela, utilisez la fonction `estAnneeBissextile()` écrite à l'exercice 1.4.

Exemple d'affichage attendu :

```
2000 est une année bissextile !

1966  1967  1968  1969  1970  1971  1972  1973  1974  1975
1976  1977  1978  1979  1980  1981  1982  1983  1984  1985
1986  1987  1988  1989  1990  1991  1992  1993  1994  1995
1996  1997  1998  1999  2000  2001  2002  2003  2004  2005
2006  2007
```

Contrainte : Les lignes comportent seulement dix liens (années).

Exercice n°2.2 : traitement de formulaire

Cet exercice a pour objectif de montrer comment on récupère des données envoyées par un formulaire. Cette technique est très utilisée dans la réalisation d'applications PHP.

Le principe est le suivant :

```
<form action="exemple.php" method="POST" name="form">
   <input type="hidden" name="id" value="4">
   <input type="submit" value="Envoyer">
</form>

alors dans le script exemple.php : $_POST['id'] sera égal à 4 .
```

En phase d'apprentissage ou de débuggage, il est recommandé de faire un `var_dump($_POST)` ou `var_dump($_GET)`.

```
<html>
   <head>
      <title>Exercice 2.2.a</title>
   </head>
   <body>
      <?php
         if(!Empty($_POST["nom"]))
         {
            $nom = $_POST["nom"];
            $heure = date("H");
            if($heure >= 18)
                  $message = "Bonsoir $nom,<br />";
            else  $message = "Bonjour $nom,<br />";
            echo $message;
         }
         echo "<br />";
```

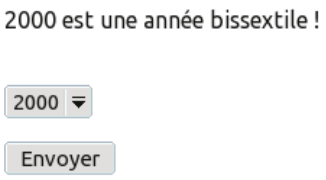
```
      ?>
      <form action="" method="POST" name="form">
         <input type="text" name="nom" value="">
         <input type="submit" value="Envoyer">
      </form>
   </body>
</html>
```

Exercice 2.2.a

Question 15. Testez le script ci-dessus. Si on change la `method` d'envoi du formulaire (`POST` en `GET`), que faut-il modifier dans le script pour qu'il fonctionne ?

Question 16. Écrire un script qui envoie par un formulaire une année choisie dans une liste déroulante et qui affichera si cette année est bissextile ou non. La liste déroulante contiendra toutes les années depuis votre naissance.

Exemple d'affichage attendu :



Conclusion

L'utilisation de PHP en tant que générateur de pages Web dynamiques est la plus répandue. Elle commence par la récupération de données (*url*, formulaire, ...) en provenance des requêtes du client.

Question 17. Pouvez-vous faire confiance aux données fournies par le client ?

Question 18. Lire l'article fr.wikipedia.org/wiki/php et conclure sur le niveau de fiabilité des applications PHP ?

Séquence n°3 : manipulation de fichiers

L'objectif de cette séquence est de s'initier à la manipulation des fichiers à partir du langage PHP.

Les fonctions de manipulation de fichiers

PHP fournit de nombreuses fonctions pour manipuler des fichiers d'un système de fichiers : fr.php.net/manual/fr/book.filesystem.php.

Les appels de base pour la gestion des E/S fichiers sont : `fopen`, `fread`, `fwrite`, `fclose`, ...

L'association entre une ressource nommée et un nom physique s'effectue à l'ouverture du fichier. La ressource nommée représentant le flux est en fait un pointeur de fichier. Le nom physique du fichier est une chaîne de caractères contenant son nom et éventuellement son chemin dans l'arborescence du système de fichiers géré par l'OS. L'ouverture d'un fichier se fait suivant un mode qui spécifie le type d'accès désiré au flux (lecture seule, écriture seule, lecture/écriture, ...).

```php
<?php
$filename = "datas.txt";
// Vérifier que le fichier existe
if (@file_exists($filename))
{
   // Ouvrir le fichier en lecture seule
   $fichier = @fopen($filename, "r");
   //Remarque : l'@ placé devant une fonction bloquera les messages d'erreurs (nuisibles)
       envoyés au navigateur client

   // Pas d'erreur à l'ouverture ?
   if($fichier != FALSE)
   {
      // Lire 10 octets dans le fichier
      $datas = fread($fichier, 10);
      echo "Dix premiers octets du fichier $filename : " . $datas . "<br />";
      // Remarque : la position du pointeur de fichier s'est déplacé de 10 octets

      // Lire (encore) 10 octets dans le fichier
      $datas = fread($fichier, 10);
      echo "Dix octets suivants du fichier $filename : " . $datas . "<br />";
      // Remarque : la fonction fseek() permettrait de modifier la position du pointeur de
          fichier

      // Fermer le fichier
      fclose($fichier);
   }
   else die("Erreur : ouverture impossible du fichier $filename !<br />");
}
else die("Erreur : le fichier $filename n'existe pas !<br />");
// Remarque : la fonction die() ou exit() affiche un message et termine le script courant
?>
```

Lire des octets dans un fichier

Les fonctions de lecture `fgetc()`, `fgetcsv()`, `fgets()`, `fscanf()` et `fgetss()` peuvent être intéressantes pour réaliser certains traitements spécifiques.

Certaines fonctions de lecture n'ont pas besoin de réaliser d'ouverture préalable du fichier à traiter. C'est le cas de :

– **`file()`** qui lit le fichier et renvoie le résultat dans un tableau
– `file_get_contents()` qui lit tout un fichier dans une chaîne

Pour écrire dans un fichier, on pourra utiliser au choix les fonctions **`fwrite()`**, `fputs()` ou même `file_put_contents()` et `fputcsv()` dans certains cas précis.

Travail demandé

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.

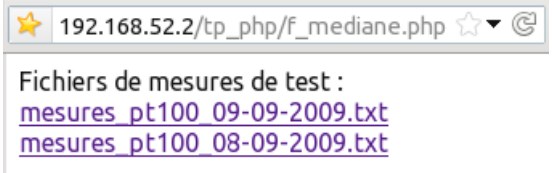
L'acquisition de mesures de température (capteur pt100) va mettre dans un fichier texte une valeur toutes les minutes. Ces séries de mesures peuvent comporter des mesures incohérentes (lire l'annexe 1). Après traitement, on ne conservera que la médiane (et non la moyenne) de ces séries de mesures. La 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.

Contraintes :
– Deux fichiers de mesures sont disponibles pour les tests. Le script doit fonctionner correctement pour ces deux fichiers (qui contiennent un 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).
– 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*) afin que le script puisse écrire dedans.

```php
<?php
// cf. http://fr.php.net/manual/fr/function.include.php
include("tri.inc.php");

// TODO
// ...

?>
```

f_mediane_todo.php

Question 19. Écrire le script `f_mediane.php` qui permet de calculer et d'écrire dans un fichier `mediane_<fichier_de_mesures>.txt` la 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>`



Annexe 1 : les mesures dans l'industrie

Dans le cas des mesures dans l'industrie, on considère 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 et 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

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 tient compte) :

Liste	L1	L2
Moyenne	35,066°C	30,506°C
Médiane	35,10°C	35,10°C

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.

Séquence n°4 : accès à une base de données

L'objectif de cette séquence est de s'initier à l'utilisation des bases de données en langage PHP.

Les fonctions d'accès aux bases de données

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**, SQLite, 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 (Système de Gestion de Base de Données Relationnelle) GPL implémentant le langage de requête SQL (*Structured Query Language*).

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

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

PHP offre 3 APIs différentes pour se connecter à MySQL : les extensions **mysql**, **mysqli** et **PDO**. PDO fournit une interface d'abstraction à l'accès de données, ce qui signifie que vous utilisez les mêmes fonctions pour exécuter des requêtes ou récupérer les données quelque soit la base de données utilisée.

Il est recommandé d'utiliser soit l'extension **mysqli**, soit l'extension **PDO_MySQL** car l'ancienne extension **mysql** est obsolète depuis PHP 5.5 et sera supprimée dans un futur proche.

PHP fournit un grand choix de fonctions permettant de manipuler une base de données MySQL. Toutefois, parmi celles-ci quatre fonctions sont essentielles :
- La fonction de connexion au serveur (`mysqli_connect` ou `mysqli_real_connect`)
- La fonction de choix de la base de données (`mysqli_select_db`)
- La fonction de requête (`mysqli_query`)
- La fonction de déconnexion (`mysqli_close`)

```php
<?php
// mysqli : http://fr.php.net/manual/fr/book.mysqli.php
// Style fonctionnel
if (!extension_loaded('mysqli'))
   die("L'extension mysqli n'est pas présente !");

$link = mysqli_connect('localhost', 'root', 'password', 'test');
if (!$link)
   die('Echec de connexion au serveur de base de données : ' . mysqli_connect_error() . '(' .
     mysqli_connect_errno() . ') ');

echo 'Fonctions mysqli : succès ... ' . mysqli_get_host_info($link) . " - MySQL server
    version : " . mysqli_get_server_info($link) . "<br />\n";

mysqli_close($link);
?>
```

Les fonctions mysqli

Avec les autres APIs :

```php
<?php
// mysqli : http://fr.php.net/manual/fr/book.mysqli.php
// Style POO
if (!class_exists('mysqli'))
   die("La classe mysqli n'est pas présente !");
// ou :
if(!in_array("mysqli", get_declared_classes()))
   die("La classe mysqli n'est pas présente !");

$mysqli = new mysqli("localhost", "root", "password", "test");
if ($mysqli->connect_error)
    die('Echec de connexion au serveur de base de données : ' . $mysqli->connect_error . '('
     . $mysqli->connect_errno . ') ');

echo 'Classe mysqli : succès ... ' . $mysqli->host_info . " - MySQL server version : " . $
    mysqli->server_info . "<br />\n";

$mysqli->close();
?>
```

La classe mysqli

```php
<?php
// MySQL Functions : http://www.php.net/manual/en/ref.mysql.php
// This extension is deprecated as of PHP 5.5.0, and will be removed in the future.
if (!extension_loaded('mysql'))
   die("L'extension mysql n'est pas présente !");

if($link = mysql_connect("localhost", "root", "password"))
{
   $id_db = mysql_select_db("test");
   if(!$id_db) die('Echec de connexion à la base : ' . mysql_error() . '(' . mysql_errno() .
       ') ');
}
else die('Echec de connexion au serveur de base de données : ' . mysql_error() . '(' .
    mysql_errno() . ') ');

echo 'Fonctions mysql : succès ... ' . mysql_get_host_info($link) . " - MySQL server version
    : " . mysql_get_server_info() . "<br />\n";

mysql_close($link);
?>
```

Les fonctions mysql (obsolète depuis PHP 5.5)

```php
<?php
// PDO : http://www.php.net/manual/en/book.pdo.php
// pdo_mysql : http://www.php.net/manual/en/ref.pdo-mysql.php

if(!in_array("PDO", get_loaded_extensions()))
   die("L'extension PDO n'est pas présente !<br><br>");

if(!in_array("pdo_mysql", get_loaded_extensions()))
   die("L'extension pdo_mysql n'est pas présente !<br><br>");
```



```php
$pdo_db = new PDO('mysql:host=localhost;dbname=test', 'root', 'password') or die("Echec de
    la création de l'instance PDO !");

echo "Classe PDO : succès ... <br />\n";

unset($pdo_db);
?>
```

La classe PDO

L'exécution d'une requête `SELECT` avec `mysqli_query()` retournera retournera un objet résultat de type `mysqli_result` (ou `TRUE` pour les autres types de requêtes).

Les fonctions de traitements de résultat d'une requête sont au choix :
- `mysqli_fetch_row()` : récupère une ligne de résultat sous forme de tableau indexé
- `mysqli_fetch_array()` : retourne une ligne de résultat sous la forme d'un tableau associatif, d'un tableau indexé, ou les deux
- `mysqli_fetch_assoc()` : récupère une ligne de résultat sous forme de tableau associatif
- `mysqli_fetch_object()` : retourne la ligne courante d'un jeu de résultat sous forme d'objet
- et `mysqli_free_result()` : libère la mémoire associée à un résultat

```php
<?php
$link = mysqli_connect('localhost', 'root', 'password', 'mysql');
if (!$link)
   die('Echec de connexion au serveur de base de données : ' . mysqli_connect_error() . '(' .
     mysqli_connect_errno() . ') ');

if ($result = mysqli_query($link, "SELECT Host, User FROM `user` ORDER BY User DESC LIMIT 0
    , 30"))
{
   printf("Fonctions mysqli : la requête a retourné %d enregistrement(s).<br />\n",
       mysqli_num_rows($result));
   /* Tableau indexé */
   //$row = mysqli_fetch_array($result, MYSQLI_NUM);
   //printf ("%s - %s<br />\n", $row[0], $row[1]);
   // ou tous les résultats de la requête
   //while(list($host, $user) = mysqli_fetch_row($result))
   //{
   //   echo "$host - $user<br />";
   //}

   /* Tableau associatif */
   //$row = mysqli_fetch_array($result, MYSQLI_ASSOC);
   //printf ("%s - %s<br />\n", $row["Host"], $row["User"]);
   // ou tous les résultats de la requête :
   while($row = mysqli_fetch_array($result))
   {
      $host = $row["Host"];
      $user = $row["User"];
      echo "$host - $user<br />";
   }

   /* Libération des résultats */
   mysqli_free_result($result);
```



```
}

mysqli_close($link);
?>
```

Traitement des résultats avec les fonctions mysqli

Avec les autres APIs :

```
<?php
$mysqli = new mysqli('localhost', 'root', 'password', 'mysql');
if ($mysqli->connect_error)
    die('Echec de connexion au serveur de base de données : ' . $mysqli->connect_error . '('
     . $mysqli->connect_errno . ') ');

if ($result = $mysqli->query("SELECT Host, User FROM `user` ORDER BY User DESC LIMIT 0 , 30"
    ))
{
   printf("Classe mysqli : la requête a retourné %d enregistrement(s).<br />\n", $result->
       num_rows);
   /* Tableau indexé */
   //$row = $result->fetch_array(MYSQLI_NUM);
   //printf ("%s - %s<br />\n", $row[0], $row[1]);
   // ou tous les résultats de la requête :
   //while(list($host, $user) = $result->fetch_array())
   //{
   //   echo "$host - $user<br />";
   //}

   /* Tableau associatif */
   //$row = $result->fetch_array(MYSQLI_ASSOC);
   //printf ("%s - %s<br />\n", $row["User"], $row["Host"]);
   // ou tous les résultats de la requête :
   while($row = $result->fetch_array())
   {
      $host = $row["Host"];
      $user = $row["User"];
      echo "$host - $user<br />";
   }

   /* Libération des résultats */
   $result->free();
}

$mysqli->close();
?>
```

Traitement des résultats avec la classe mysqli

```
<?php
if($link = mysql_connect("localhost", "root", "password"))
{
   $id_db = mysql_select_db("mysql", $link);
   if(!$id_db) die('Echec de connexion à la base : ' . mysql_error() . '(' . mysql_errno() .
     ') ');
}
```



```
else die('Echec de connexion au serveur de base de données : ' . mysql_error() . '(' .
    mysql_errno() . ') ');

if ($result = mysql_query("SELECT Host, User FROM `user` ORDER BY User DESC LIMIT 0 , 30", $
    link))
{
   printf("Fonctions mysql : la requête a retourné %d enregistrement(s).<br />\n",
    mysql_num_rows($result));
   /* Tableau indexé */
   //$row = mysql_fetch_array($result, MYSQLI_NUM);
   //printf ("%s - %s<br />\n", $row[0], $row[1]);
   // ou tous les résultats de la requête :
   //while(list($host, $user) = mysql_fetch_row($result))
   //{
   //   echo "$host - $user<br />";
   //}

   /* Tableau associatif */
   //$row = mysql_fetch_array($result, MYSQLI_ASSOC);
   //printf ("%s - %s<br />\n", $row["Host"], $row["User"]);
   // ou tous les résultats de la requête :
   while($row = mysql_fetch_array($result))
   {
      $host = $row["Host"];
      $user = $row["User"];
      echo "$host - $user<br />";
   }

   /* Libération des résultats */
   mysql_free_result($result);
}

mysql_close($link);
?>
```

Traitement des résultats avec les fonctions mysql (obsolète depuis PHP 5.5)

```
<?php
$pdo_db = new PDO('mysql:host=localhost;dbname=mysql', 'root', 'password') or die("Echec de
    la création de l'instance PDO !");

if ($result = $pdo_db->query("SELECT Host,User FROM `user` LIMIT 0 , 30"))
    printf("Classe PDO : la requête a retourné %d enregistrement(s).<br />\n", $result->
        rowCount());

if ($result = $pdo_db->query("SELECT Host, User FROM `user` ORDER BY User DESC LIMIT 0 , 30"
    ))
{
   printf("Classe PDO : la requête a retourné %d enregistrement(s).<br />\n", $result->
       rowCount());
   /* Tableau indexé */
   //$row = $result->fetch(PDO::FETCH_NUM);
   //printf ("%s - %s<br />\n", $row[0], $row[1]);
```

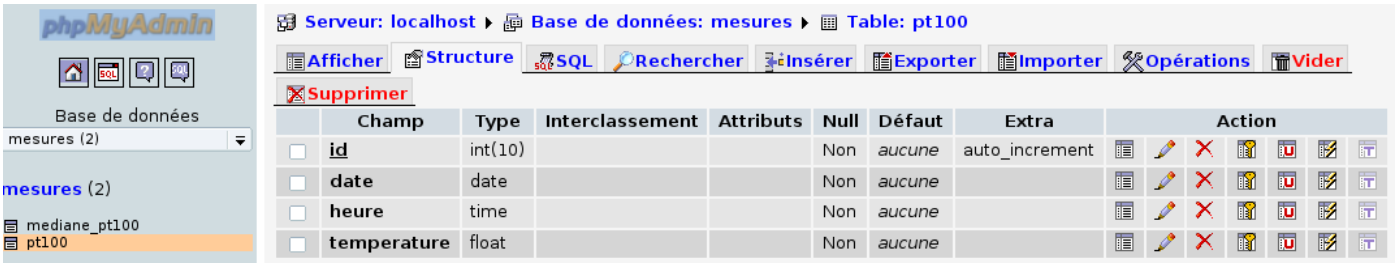
```php
   /* Tableau associatif */
   //$row = $result->fetch(PDO::FETCH_ASSOC);
   //printf ("%s - %s<br />\n", $row["User"], $row["Host"]);

   // ou tous les résultats de la requête :
   $datas = $result->fetchAll();
   echo "<pre>"; print_r ($datas); echo "</pre>";
}

unset($pdo_db);
?>
```

Traitement des résultats avec la classe PDO

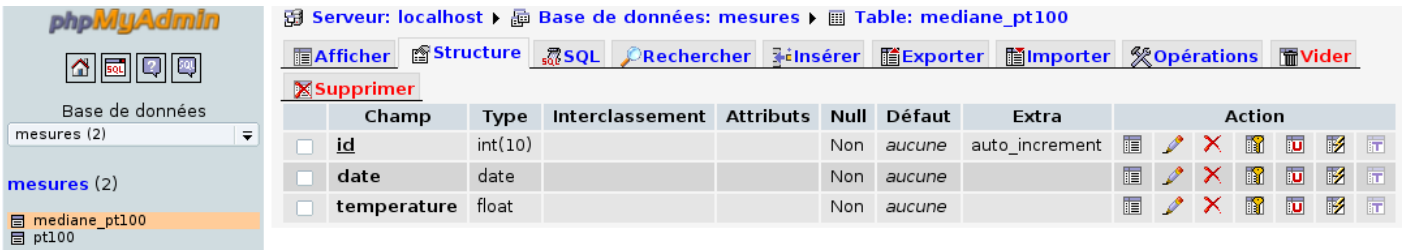
Travail demandé

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 :



Le fichier `mesures.sql` permettant de créer cette base de données (avec des mesures déjà effectuées) sur un serveur MySQL vous est fourni.

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).
– 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.

```php
<?php
// cf. http://fr.php.net/manual/fr/function.include.php
include("tri.inc.php");

// TODO
// ...

?>
```

bd_mediane_todo.php

Question 20. Écrire le script `bd_mediane.php` qui permet de calculer et d'écrire dans une table MySQL **mediane_pt100** la médiane d'une série de mesures lues dans une table MySQL **pt100** dont la date (format AAAA-MM-JJ) est passée par l'envoi d'un formulaire.