PHP/MySQL

École Nationale Supérieure des Mines de Nancy – 2nde Année

Table des matières

Première partie
PHP

1 Les bases

1.1 Introduction

Les curieux désirant obtenir des informations relatives à l'historique de php pourront se documenter sur *php.net*. La seule chose que vous devez savoir avant d'aborder la programmation php, c'est que le principal objectif de ce langage est de générer des pages html. Pour générer ces pages, il faut disposer d'un serveur qui va interpréter le code php, c'est pourquoi il est inutile d'essayer d'appeler une page php à partir son adresse d'origine [1]. Il faut donner à cette page une adresse virtuelle (via le serveur php) pour qu'à l'appel de la page [2], il puisse l'interpréter et envoyer au client une page html qu'il aura générée.

Exemple : Soit la page suivante

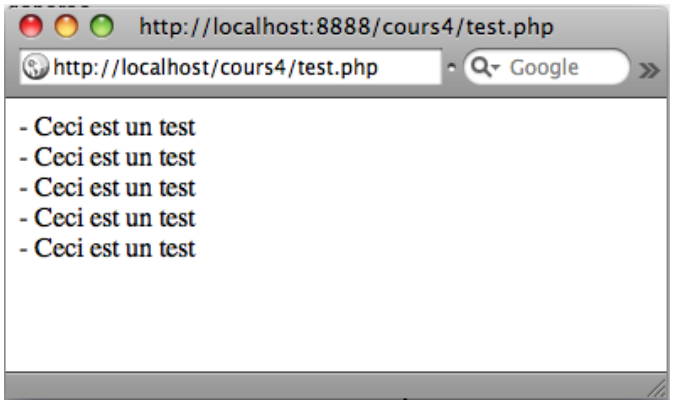
test.php

```
<?php
    $s = "Ceci est un test";
    for($i=0;$i<5;$i++)
         echo "- $s <br/> ";
?>
```

Si l'on appelle la page avec son adresse d'origine, on obtient :

Et si l'on appelle cette même page via le serveur, on obtiendra alors :

1. /home/blabla/test.php ou C :/blabla/test.php
2. adresse_du_serveur/alias_du_dossier/test.php



Ceci étant dit, entrons dans le vif du sujet.

1.2 Préparation de l'environnement

Comme vous l'avez compris, pour effectuer nos tests, nous allons avoir besoin d'un serveur php, c'est pourquoi il est nécessaire de lancer Wamp, EasyPHP ou Apache2 (selon votre OS) puis de créer un dossier `/var/www/cours4` ou `C:/Wamp/www/cours4`.

1.3 Php & html

Comme il l'a été dit dans l'introduction, le but de la programmation php est principalement de générer du code html. Par exemple, le code suivante ne produira rien du tout :

test_bidon.php

```
<?php
    $s = "Ceci est un test bidon";
    $var = 0;
    for($i=0;$i<5;$i++)
         $var = $var * 2;
?>
```

Le serveur aura toutefois executé tout le script, mais n'aura rien à afficher. Précisons qu'une page php peut avoir une autre utilité que l'affichage, comme la modification de bases de données, ... mais nous verrons cela dans la seconde partie consacrée à **MySQL**. Jusque là, notre objectif sera de générer du code html.
La plupart du temps, un fichier php ne possède qu'une petite proportion de php et est surtout composé de html. Le code source php est directement inséré dans le code html grâce aux balises **< ?php ... ?>** :

exemple.php

```
<html>
 <head>
   <title>Ceci est une page html/php</title>
 </head>
 <body>
 Ceci est de l'html <br/><br/>

<?php   /* Ceci est du php */

    $s = 1;
    for($i=0;$i<5;$i++){
```

```
        echo "- $s <br/> ";
        $s = $s * 2;
}
?>
 </body>
</html>
```

La page exemple_insertion.php sera traduite par le serveur en une page ne contenant que du html. Le client (= le navigateur) n'aura accès qu'à la traduction html de la page php et ne pourra jamais accéder au code source :

exemple.php vu du client

```
<html>
 <head>
   <title>Ceci est une page html/php</title>
 </head>
 <body>
 Ceci est de l'html <br/><br/>

- 1 <br/> - 2 <br/> - 4 <br/> - 8 <br/> - 16 <br/>   </body>

</html>
```

Trois commandes différentes assurent la fonction d'affichage :
- echo
- print
- printf($format$,$arguments$) : écriture formatée comme en C.

Dans ce cours, il sera uniquement fait usage de la commande **echo**. Elle a l'avantage de s'utiliser de plusieurs façons :
- echo("Une chaîne de caractère") ;
- echo "une chaîne de caractère" ;
- echo($*une_variable*) ;
- echo $*une_variable* ;
- echo "une chaîne $*une_variable*" ;
- echo 'une chaîne '.$*une_variable* ;

A noter que si vous utilisez une syntaxe délimitant les chaînes par des guillemets, il faudra protéger chaque guillemet à l'intérieur de la chaîne : `echo "c'est un \" exemple \" ";`. Idem avec l'usage des simples quotes : `echo 'c\'est un "exemple"'`.

2 Syntaxe et typage

2.1 Variables et types

Bonne nouvelle : le typage des variables est implicite en php ! Il n'est pas pas nécessaire de déclarer le type d'une variable avant son utilisation ! Les identificateurs de variables sont précédés du symbole "$".

Elles peuvent être de type entier (integer), réel (double), chaîne de caractère (string), tableau (array), objet (object), booléen (boolean).

En php, vous pouvez aussi programmer en objet exactement de la même façon qu'en Java. Mais nous ne verrons pas la programmation objet ici. Je vous conseille toutefois d'aller jeter un oeil à la page *http ://hachesse.developpez.com/objetphp/*.

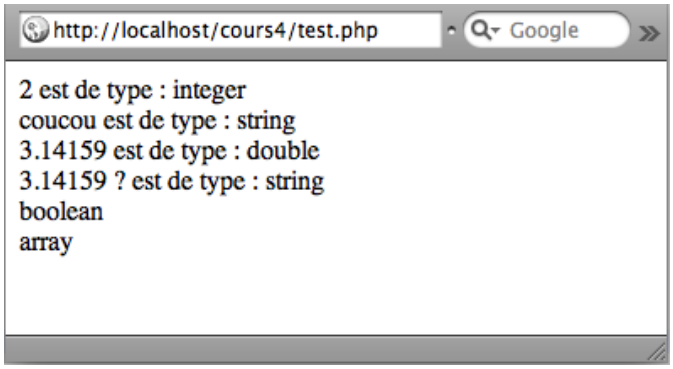
Une spécificité non négligeable du php est la possibilité d'avoir la valeur d'une variable pour identificateur d'une autre variable.
Exemple :
$test = "varname"
${$test} = "2A"
$varname contiendra alors la chaîne "2A" !

Comme on vient de le dire, le typage des variables est implicite, ce qui veut dire que le code suivant est tout à fait licite :

exemple_types

```
<?php
    $var = 2;          echo $var." est de type : ".gettype($var)."<br/>";
    $var = "coucou";   echo $var." est de type : ".gettype($var)."<br/>";
    $var = 3.14159;    echo $var." est de type : ".gettype($var)."<br/>";
    $var = $var." ?";  echo $var." est de type : ".gettype($var)."<br/>";
    $var = false;      echo gettype($var)."<br/>";
    $var = array(1,"coucou",$var);  echo gettype($var)."<br/>";
?>
```

Ce code produira la page suivante :



La variable $var prend successivement tous les types primitifs possibles. Vous remarquerez que hormis les types boolean et array, tous les types primitifs peuvent être affichés avec la commande *echo* sans avoir besoin d'être convertis en chaîne de caractères.

2.2 Opérateurs

Les opérateurs arithmétiques, logiques, d'affectation et de comparaison sont les mêmes qu'en C et en Java.

Exemple :

exemple_opérateurs.php

```
<?php
    $x = 2;
    $y = $x + 2;
```

```
    if($y<5 && $x>0){
        echo "1";
    }else{
        echo "2";
    }
?>
```

2.3 Tableaux

Les variables tableaux sont de type **array**. Un tableau accepte des éléments de types différents !

Il a y deux façons de créer un tableau :
– soit avec la fonction 'array' :

```
$tab = array("a",1,false);
```

– soit en affectant chaque case du tableau :

```
$tab[0] = "a";
$tab[1] = 1;
$tab[2] = false;
```

Voici quelques fonctions utiles :
– `count($tab)` ou `sizeof($tab)` : nombre d'éléments de `$tab`
– `in_array($var,$tab)` : retourne vrai si `$var` a une occurence dans `$tab`
– `sort($tab)` trie `$tab` dans l'ordre lexicographique
– `array_merge($tab_1,...,$tab_n)` : concatène les tableaux en argument

En php on a la possibilité d'utiliser des tableaux associatifs. On associe à chaque élément une clé de type string. La création d'un tableau associatif se fait quasiment de la même façon que pour un tableau traditionnel :
– soit avec la fonction 'array' :

```
$tab = array("lettre" => 'a',"chiffre" => 1,"autre" => false);
```

– soit en affectant chaque case du tableau :

```
$tab["lettre"] = "a";
$tab["chiffre"] = 1;
$tab["autre"] = false;
```

Il y a plusieurs façons de parcourir un tableau :
– soit avec une boucle for traditionnelle (on ne s'étendra pas sur cette syntaxe connue de tous)
– soit avec **foreach** :

```
foreach($tableau as $element)
    echo $element;
```

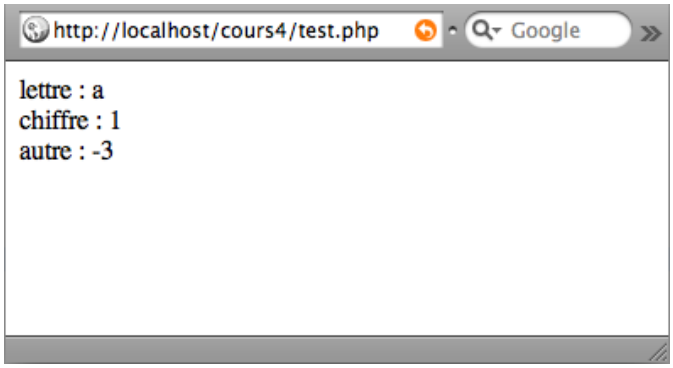
– et ce qui est encore plus intéressant dans le cas des tableaux associatifs

```
foreach($tableau as $cle => $element)
    echo "$cle : $element";
```

Exemple :

parcours_tableau.php

```
<?php
     $tab = array("lettre" => 'a',"chiffre" => 1,"autre" => 4-7);
     foreach($tab as $cle => $elt){
          echo "$cle : $elt<br/>";
     }
?>
```



À vous de jouer : Écrire une page php qui affiche un tableau à deux colonnes où la première colonne correspond à la clé et la seconde à l'élément à partir du tableau associatif suivant :

```
$ens = array("septembre" => "Intégration",
     "novembre" => "Gala",
     "décembre" => "Anim'EST",
     "janvier" => "Laser Game",
     "février" => "Fin 3A");
```

On rappelle la structure de base d'un tableau en html : le code suivant

```
<table>
      <tr>
            <td> l1,c1 </td> <td> l1,c2 </td>
      </tr>
      <tr>
            <td> l2,c1 </td> <td> l2,c2 </td>
      </tr>
...
</table>
```

produit ce tableau :

l1,c1	l1,c2
l2,c1	l2,c2

2.4 Fonctions

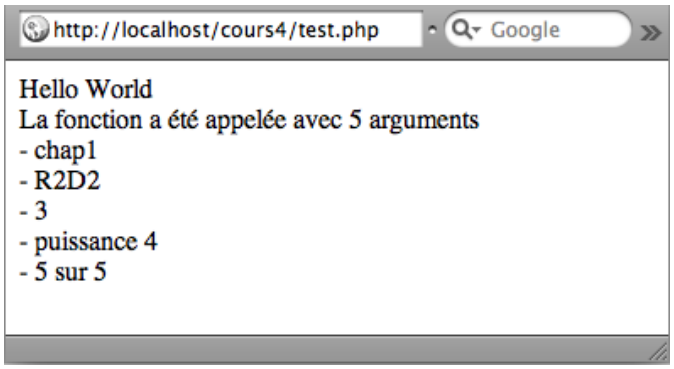
Comme tout langage de programmation, php permet d'écrire des fonctions, mais d'une façon particulière puisqu'il n'est pas nécessaire de spécifier le type des arguments.

test_fonction.php

```
<?php

function bonjour(){
    echo "Hello World<br/>";
}

function estAdmin($login, $pass){
    return($login == "Admin"  &&  $pass == "BDISI");
```

```
}

function listeArgument(){
    $n = func_num_args();
    echo "La fonction a été appelée avec $n arguments<br/>";
    for($i=0;$i<$n;$i++)
         echo " - ".func_get_arg($i)."<br/>";
}

bonjour();

if(estAdmin("Admin","BDISI"))
    listeArgument("chap1","R2D2",3,"puissance 4","5 sur 5");
else
    listeArgument("logique","predicats","proposition","atomes","clause");

?>
```

test_fonction.php produit la page suivante :



2.5 Inclusions

On peut inclure dans un script php le contenu d'un autre fichier de deux façons :
– **require**("fichier.php") : insert une seule fois dans le code le contenu de fichier.php.
– **include**("fichier.php") : insert à chaque appel le contenu de fichier.php et l'évalue.

L'inclusion est utile si vous écrivez un fichier contenant toutes vos fonctions. Il vous suffit alors d'inclure le fichier en question via la commande **require** pour pouvoir les appeler. Elle est aussi utile lorsque des morceaux de pages html se retrouvent sur plusieurs pages de votre site. Admettons que vous utilisiez le même fond et la même en-tête pour chacune de vos pages, il devient alors intéressant de créer une page html ou php qui ne contient que le fond et l'en-tête que vous pouvez inclure au début de chacune de vos pages web. De cette façon, il vous suffira de modifier ce fichier pour que toutes les pages de votre site le soient aussi.

2.6 Formulaires

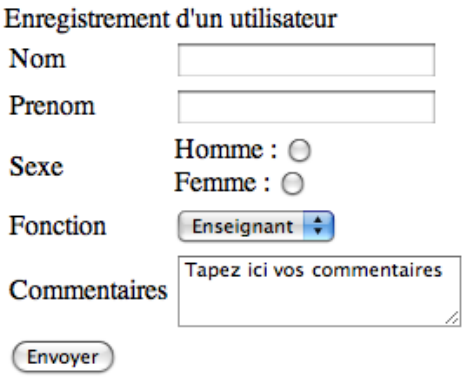
Les formulaires vous permettent d'envoyer des informations au serveur. Ceci est utilisé par exemple lorsque l'on a recours à des bases de données. L'usage des formulaires se décompose en deux parties :
– l'écriture du formulaire qui peut se faire entièrement en html
– la récupération des données qui se fait entièrement en php

2.6.1 Écriture de formulaires

formulaire.html

```
<FORM method=post action="pageCible.php">
Enregistrement d'un utilisateur
<TABLE BORDER=0>
<TR>
     <TD>Nom</TD>
     <TD>
          <INPUT type="text" name="nom"/>
     </TD>
</TR>

<TR>
     <TD>Prénom</TD>
     <TD>
          <INPUT type="text" name="prenom"/>
     </TD>
</TR>

<TR>
     <TD>Sexe</TD>
     <TD>
          Homme : <INPUT type="radio" name="sexe" value="M"/>
     <br/> Femme : <INPUT type="radio" name="sexe" value="F"/>
     </TD>
</TR>

<TR>
     <TD>Fonction</TD>
     <TD>
          <SELECT name="fonction">
              <OPTION VALUE="enseignant">Enseignant</OPTION>
              <OPTION VALUE="etudiant">Etudiant</OPTION>
              <OPTION VALUE="ingenieur">Ingénieur</OPTION>
              <OPTION VALUE="retraite">Retraité</OPTION>
              <OPTION VALUE="autre">Autre</OPTION>
           </SELECT>
     </TD>
</TR>
<TR>
     <TD>Commentaires</TD>
     <TD>
           <TEXTAREA rows="3" name="commentaires">
           Tapez ici vos commentaires</TEXTAREA>
     </TD>
</TR>

<TR>
     <TD>
     <INPUT type="submit" value="Envoyer"/>
     </TD>
</TR>
</TABLE>
</FORM>
```

L'exemple ci-dessus, dont le rendu html est le suivant :



permet de donner le principal de la syntaxe d'un formulaire. Pour plus de détails, vous pouvez vous rendre sur la page *http ://www.commentcamarche.net/contents/html/htmlform.php3*.
Par ailleurs, il peut être astucieux d'automatiser la création de formulaires via des fonctions php :

formulaire.php

```
<?php

function formulaire($titre,$fichierCible){
echo "<form method=\"post\" action=\"$fichierCible\">
      <b>$titre</b>
      <table border=0>";
}

function champ($titre,$nom,$taille){
 echo "<tr>
         <td>$titre </td>
         <td><input type=\"text\" name=\"$nom\" size=\"$taille\" ></td>
      <tr>";
}

function liste_deroulante($nom,$liste){
 echo "<tr>
        <td>$nom</td>
        <td><select name=\"$nom\">";
         for($i=0;$i<count($liste);$i++){
             echo "<option>".$liste[$i]."</option>";
         }
 echo "</select></td></tr>";
}

function fin_formulaire(){
  echo "</table></form>";
}
?>
```

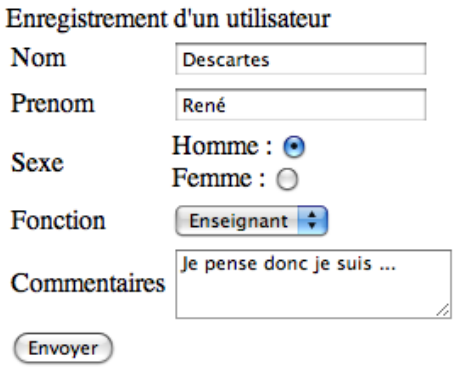
✎ **A vous de jouer** : Après avoir réécrit et éventuellement complété le fichier formulaire.php, créer un fichier php générant le formulaire précédent (uniquement en php).

2.6.2 Récupération des données

Ecrire un formulaire, c'est bien, récupérer les informations, c'est mieux. Il y a deux choses à remarquer :
- la méthode utilisée par le formulaire : `<form method="post" ...>`
- le fichier à qui sont envoyées les informations : `action="nom_du_fichier.php"`
- le nom de chaque élément du formulaire : `name = "nom_de_l_element"`

Ceci signifie que la valeur de l'élément de nom `nom_de_l_element` est accessible dans le fichier `nom_du_fichier.php` par la variable `$_POST["nom_de_l_element"]`.

A vous de jouer : Ecrire un fichier php récupérant les données envoyées par le formulaire précédemment créé. La validation du formulaire :



doit produire l'affichage suivant : « Je m'appelle Descartes René, je suis un homme et suis enseignant. J'ai laissé le commentaire suivant : je pense donc je suis ... »

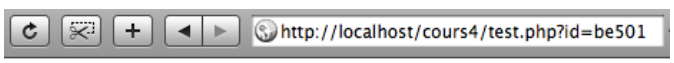
2.7 Transmission de valeurs

Des valeurs de variables peuvent également être passées par l'intermédiaire de l'URL et se récupèrent via la commande `$_GET["nom_de_la_variable"]`.

test_parametre_url.php

```
<?php
    echo "L'id passe en parametre dans l'URL est ".$_GET["id"]."<br/>";
?>
```

L'id passe en parametre dans l'URL est 5



L'id passe en parametre dans l'URL est be501

Deuxième partie
MySQL

MySQL est un système de gestion de base de données (SGBD) très simplement utilisable avec php. Cette partie requiert la connaissance du langage *SQL*.

Pour se connecter à une base depuis une page php, il faut spécifier un nom de serveur, un nom d'utilisateur, un mot de passe et un nom de base (par défaut, sur votre machine, respectivement 'localhost', 'root', '', 'BDISI'). Les fonctions de connexion sont :
- **mysql_connect($server,$user,$password)** : returne **true** en cas de succès et **false** en cas d'échec.
- **mysql_select_db($base)** : se connecte à la base $base du serveur sur lequel on s'est connecté via la commande précédente.
- **mysql_close()** : ferme la connection

Pour éviter de retenir ces fonctions, il est recommandé d'écrire une fonction exécutant les requêtes MySQL. Dans votre cas, il suffit d'inclure le fichier mysql.php (disponible à l'adresse `www.tony-bourdier.fr/BDISI/mysql.php`) et d'utiliser les fonctions suivantes :
- **requete($sql)** qui exécute la requête passée en paramètre.
- **affiche()** qui affiche le résultat de la dernière requête exécutée.
- **colonne($nom)** qui retourne un tableau contenant la colonne du résultat de la dernière requête exécutée dont le nom (ou le numéro) est passé en paramètre.
- **ligne($i)** qui retourne un tableau contenant la i^e ligne du résultat de la dernière requête exécutée.

Les variables `$_SQLnligne` et `$_SQLncolonne` contiennent respectivement le nombre de lignes et de colonnes du résultat de la dernière requête SQL exécutée.

Exemple : Le fichier suivant :

requete.php

```
require("mysql.php");
requete("SELECT * FROM cours");
affiche();
```

produit le résultat suivant :

SIGLE	INTITULE	RESPONSABLE	NOMBRESEANCES
ALG021	Algebre Lineaire	ch1893	16
ALGO510	Algorithmique	pb1623	12
ARITH014	Arithmetique	dfp1601	12
DIFF024	Calcul Differentiel	el1707	8
LOG015	Logique	jh1908	13
PROBA201	Probabilites	th1702	16
SI033	Bases de donnees	ft012	16
STAT010	Statistiques	ap0020	8
STOCH021	Modelisation Stochastique	bn1687	16

Les égalités suivantes seront alors effectives :
- `ligne(1) == array("ALG021","Algebre Lineaire","ch1893",16)`
- `colonne("responsable") == colonne(3) == array("ch1893","pb1623,"dfp1601",...)`

À vous de jouer : Écrire (en une ou plusieurs pages php) un formulaire comportant la liste (déroulante) des cours (Algèbre Linéaire, Algorithmique, ...) et dont la validation affiche l'ensemble des étudiants inscrits au cours sélectionné.