

Les langages PHP 5 et MySQL

IUT A - Département Informatique
Université Paul Sabatier

Table des matières

1. Introduction

2. Les fonctionnalités du langage PHP

3. Passage et transmission de variables

4. Exploitation du SGBD MySQL

5. Accès à des fichiers

6. Variables persistantes : *cookies* et *sessions*

Généralités

- PHP: Hypertext Preprocessor
- Langage interprété (pas compilé)
- Principalement utilisé pour être exécuté par un serveur HTTP
- Supporte nombreux SGBD
- Langages concurrents pour les sites web dynamiques : JSP, ASP, Python, CGI

Fonctionnement

- Un site web développé en PHP comprend :
 - ▶ l'interpréteur PHP
 - ▶ le serveur web



Avantages / Limitations

- MultiOS (Windows, Linux, Unix, MacOS, etc)
- Multi plates-formes (Apache, IIS, Netscape, etc)
- Gratuit et *open source* (licence GNU GPL)
- Simplicité d'écriture / disponibilité de codes
- Langage interprété
- COO limité (mieux en PHP5)
- Rapidité et maintenabilité à grande échelle

Table des matières

1. Introduction
- 2. Les fonctionnalités du langage PHP**
3. Passage et transmission de variables
4. Exploitation du SGBD MySQL
5. Accès à des fichiers
6. Variables persistantes : *cookies* et *sessions*

Premiers éléments

- Code placé entre balises `<?php ... ?>`
- Une instruction se termine TOUJOURS par un ;
- Commentaires : *//ligne commentée* ou */* ... */*

```
<?php  
echo 'Hello world';  
// une ligne de commentaire  
/* plusieurs  
lignes de  
commentaires */  
?>
```

PHP et HTML

```
<html>
  <body>
    <?php
      echo 'Du texte PHP';
    ?>
    Du texte HTML
    <font size="3">
      <?php echo 'Encore du texte PHP'; ?>
    </font>
  </body>
</html>
```


Variables (1/2)

- Définies sous la forme *\$nom_variable*
- Affectation : *\$nom_variable = valeur ;*
- *Type de données :*
 - ▶ *integer (entiers)*
 - ▶ *double (réels à virgule flottante)*
 - ▶ *string (chaîne de caractères)*
 - ▶ *array (tableaux)*
 - ▶ *objets (cf. Concepts Orientés objets)*

Variables (2/2)

```
<?php
    $foo = 'bonjour 1'; //chaîne de caractères
    $foo = "bonjour 2"; //chaîne de caractères
    $foo = 5; //integer
    $foo = 1.5; //double

    $foo = "0"; //chaîne de caractères ("0")
    $foo++; //chaîne de caractères ("1")
    $foo += 1; //integer (2)
    $foo = $foo + 1.3 //double (3.3)
    $foo = 5 + "10 little pigs"; //integer (15)
?>
```

Opérateurs arithmétiques

$\$a + \b	Addition	Somme de $\$a$ et $\$b$
$\$a - \b	Soustraction	Reste de la différence de $\$b$ et $\$a$
$\$a * \b	Multiplication	Produit de $\$a$ par $\$b$
$\$a / \b	Division	Dividende de $\$a$ par $\$b$
$\$a \% \b	Modulo	Reste de la division entière de $\$a$ par $\$b$

Opérateurs sur les chaînes

- Un seul opérateur : concaténation “.”

```
<?php
$a = "Hello ";
$b = $a."world!"; //$b contient "Hello World!"
?>
```

- Chaîne encadrée de simples ou doubles quotes
- Doubles quotes : remplacement des variables par leur valeur

```
<?php
$foo = 'penguin ';
echo 'Hello $foo'; //affiche "Hello $foo"
echo "Hello $foo"; //affiche "Hello penguin"
echo "<a href=\"http://www.foo.org\">mon lien</a>";
?>
```

Opérateurs logiques

<code>\$a and \$b</code>	Et	Vrai si \$a ET \$b sont vrais
<code>\$a or \$b</code>	Ou	Vrai si \$a OU \$b sont vrais, ou les deux
<code>\$a xor \$b</code>	Ou exclusif	Vrai si \$a OU \$b est vrai, mais pas les deux
<code>!\$a</code>	Négation	Vrai si \$a est faux
<code>\$a && \$b</code>	Et	Vrai si \$a ET \$b sont vrais
<code>\$a \$b</code>	Ou	Vrai si \$a OU \$b sont vrais, ou les deux

Opérateurs de comparaison

$a == b$	Egal	Vrai si a est égal à b
$a != b$	Différent	Vrai si a est différent de b
$a < b$	Inférieur	Vrai si a est strictement inférieur à b
$a > b$	Supérieur	Vrai si a est strictement supérieur à b
$a \leq b$	Inférieur ou égal	Vrai si a est inférieur ou égal à b
$a \geq b$	Supérieur ou égal	Vrai si a est supérieur ou égal à b

Condition *if...else...elseif*

```
<?php
    if ($expr1) {
        echo "$expr1 est vrai";
    }
    elseif ($expr2) {
        echo "$expr2 est vrai";
    }
    ...
    else {
        echo "tout est faux";
    }
?>
```

```
<?php
    $i = 100;
    if ($i >= 0 && $i < 200) {
        echo $i." est compris entre 0 et 200";
    }
    elseif ($i >= 200 && $i < 500) {
        echo $i." est compris entre 200 et 500";
    }
    else {
        echo $i." est supérieur à 500";
    }
?>
```

Condition *switch*

```
<?php
    switch ($foo) {
        case condition1 :
            //Traitement condition1
            break;
        case condition2 :
            //Traitement condition2
            break;
        ...
        default :
            //Traitement par défaut
    }
?>
```

```
<?php
    $i = 1;
    switch ($i) {
        case 0 :
            echo "La variable vaut 0";
            break;
        case 1 :
            echo "La variable vaut 1";
            break;
        default :
            echo "La variable n'appartient pas à [0-1]";
    }
?>
```


Itération avec *while*

```
<?php
    while (condition) {
        //Traitement
    }
?>
```

```
<?php
    $i = 0;
    $max = 10;
    while ($i < $max) {
        echo "$i est inférieur à $max";
        $i++;
    }
    echo "$i est égal à $max";
?>
```

Itération avec *for*

```
<?php
    for (cond init; cond sortie; iter) {
        //Traitement
    }
?>
```

```
<?php
    $max = 10;
    for ($i = 0; $i < $max; $i++) {
        echo "$i est inférieur à $max";
    }
    echo "$i est égal à $max";
?>
```

Fonctions spécifiques

- Déclaration

```
<?php
    function nomFonction($params) {
        //Traitement de la fonction
    }

    function nomFonction($params) {
        //Traitement de la fonction
        return ($resultat);
    }
?>
```

- Appel

```
<?php
    nomFonction($params);
    $resultat = nomFonction($params);
?>
```

```
<?php
    function sayHello($prenom) {
        echo "Bonjour $prenom";
    }

    function additionner ($i, $j) {
        return ($i + $j);
    }
?>
```

```
<?php
    //Affiche "Bonjour Dupont"
    sayHello("Dupont");

    $res = additionner (2, 3);
    echo $res;
?>
```

Fonction include()

- Permet d'inclure un fichier B lors de l'exécution d'un fichier A
- A utiliser pour les portions de code ou motifs répétitifs

Fichier A

```
<html>
  <body>
    <?php
      include("fichierB.php");
    ?>
  </body>
</html>
```

Fichier B

```
<?php
  // code exécuté lors de
  // l'appel au fichier A;
?>
```

Fonction require()

- Fonctionne de la même façon que *include()*
- L'exécution du script s'arrête si le fichier à inclure est introuvable

```
<?php
    //Stoppera l'exécution du script
    //si le fichier est introuvable
    require("verif.php");

    /*
    Code exécuté si le fichier est
    présent
    */
?>
```

Les tableaux

- Les tableaux “classiques” (à index numérique)

```
<?php
//Tableau à index numéroté
$tab = array (valeur0, valeur1, valeur2, ...);

//Accès à chacune des valeurs
$valeur0 = $tab[0];
$valeur1 = $tab[1];
?>
```

- Les tableaux associatifs (à index associatif)

```
<?php
//Tableau associatif
$tab = array (index1=>valeur1, index2=>valeur2, ...);

//Accès à chacune des valeurs
$valeur1 = $tab['index1'];
$valeur2 = $tab['index2'];
?>
```

Parcours de tableaux

- La fonction *foreach()*

```
<?php
//Cas du tableau à index numéroté
$tab = array (valeur0, valeur1, valeur2, ...);
foreach ($tab as $val){
    echo $val; //Retourne valeur0, valeur1, ...
}

//Cas du tableau associatif
$tab = array (index1=>valeur1, index2=>valeur2, ...);
foreach ($tab as $val){
    echo $val; //Retourne valeur1, valeur2, ...
}
//Consultation des index et valeur
foreach ($tab as $index=>$val){
    echo "$index a pour valeur $val";
}
?>
```

Recherche dans un tableau

- La fonction *array_key_exists()*

```
<?php
//Cas du tableau associatif
$tab = array (index1=>valeur1, index2=>valeur2, ...);
if array_key_exists("nom_index", $tab){
    echo "L'index \"nom_index\" existe dans le tableau";
}
?>
```

- La fonction *in_array()*

```
<?php
//Cas du tableau associatif
$tab = array (index1=>valeur1, index2=>valeur2, ...);
if in_array("valeur1", $tab){
    echo "La valeur \"valeur1\" existe dans le tableau";
}
?>
```


Table des matières

1. Introduction
2. Les fonctionnalités du langage PHP
- 3. Passage et transmission de variables**
4. Exploitation du SGBD MySQL
5. Accès à des fichiers
6. Variables persistantes : *cookies* et *sessions*

Transmission par formulaire

- Quand un formulaire est rempli et envoyé, le contenu des champs saisis est transféré à la page destination sous formes de variables en utilisant les méthodes GET ou POST

```
<html>
  <body>
    <!-- envoi d'un formulaire en POST -->
    <form action="destination.php" method="post">
      ....
    </form>
    <!-- envoi d'un formulaire en GET -->
    <form action="destination.php" method="get">
      ....
    </form>
  </body>
</html>
```

Récupération du formulaire

- Méthode GET : les paramètres apparaissent dans la barre d'adresse du navigateur
- Méthode POST : les paramètres sont invisibles
- Récupération des variables : tableau `$_POST` ou `$_GET`

```
<?php
//dans le cas d'un envoi des paramètres en POST
$var1 = $_POST['nom_champ'];
//dans le cas d'un envoi des paramètres en GET
$var1 = $_GET['nom_champ'];
?>
```

Exemple

- Le fichier *formulaire.html*

```
<html>
  <body>
    <form action="traitement.php" method="post">
      Nom : <input type="text" name="nom"><br />
      Prenom : <input type="text" name="prenom"><br />
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

- Le fichier *traitement.php*

```
<?php
//Récupération des paramètres du formulaire
$nom = $_POST['nom'];
$prenom = $_POST['prenom'];
//Affichage des paramètres
echo "Bonjour $nom $prenom !";
?>
```

Transmission par hyperlien

- Les hyperliens peuvent être utilisés pour faire passer des paramètres ou variables d'une page source vers une page destination

```
<a href="destination.php?var1=contenu1&var2=contenu2&...">  
    mon lien avec des paramètres  
</a>
```

- La récupération des paramètres dans la page destination se fait avec le tableau `$_GET`, ou directement avec l'appel à `$var1`, `$var2`, ...

```
<?php  
    $variable1 = $_GET['var1'];  
    $variable2 = $_GET['var2'];  
?>
```

Table des matières

1. Introduction
2. Les fonctionnalités du langage PHP
3. Passage et transmission de variables
- 4. Exploitation du SGBD MySQL**
5. Accès à des fichiers
6. Variables persistantes : *cookies* et *sessions*

MySQL

- Système de Gestion de Base de Données (SGBD) pour entreposer des données de manière structurée
- Langage spécifique : Simple Query Language



- (1) Le serveur Web transmet la page PHP à l'interpréteur PHP
- (2) L'interpréteur PHP exécute le script et envoie les requêtes à MySQL
- (3) MySQL exécute les requêtes et retourne les résultats à l'interpréteur PHP
- (4) L'interpréteur PHP renvoie la page HTML générée au serveur Web

Rappels du langage SQL (1/2)

- Création d'une base de données *exercice*

```
CREATE DATABASE exercice
```

- Création d'une table *t_personne*

```
CREATE TABLE 't_personne' (  
  'id' INT(6) AUTO_INCREMENT,  
  'nom' VARCHAR(50),  
  'prenom' VARCHAR(50),  
  'age' TINYINT (3),  
  UNIQUE('id')  
)
```


Rappels du langage SQL (2/2)

- Ajout d'un enregistrement

```
INSERT INTO t_personne (id,nom, prenom, age)
VALUES (NULL, 'Jean', 'Dupont', 25)
```

- Consultation d'un enregistrement

```
SELECT * FROM t_personne WHERE id=1
```

- Mise à jour d'un enregistrement

```
UPDATE t_personne SET age=35 WHERE id=1
```

- Suppression d'un enregistrement

```
DELETE FROM t_personne WHERE id=1
```

Accès à MySQL via PHP

- Connexion à un serveur MySQL

```
<?php
//Connexion au serveur MySQL localhost
$lien = mysql_connect("localhost", "login", "mdp");
if ($lien == 0)
    exit("Connexion impossible au serveur localhost");
?>
```

- Connexion à une base de données

```
<?php
//Connexion à la base de données exercice
if (mysql_select_db("exercice", $lien) == 0)
    exit("Connexion impossible à la base exercice");
?>
```

Exécution d'une requête SQL

- Fonction *mysql_query()*

```
<?php
//On suppose la connexion à la base exercice déjà
//établie
$requete = "INSERT INTO t_personne (id, nom, prenom,
age) VALUES (NULL, \"Jean\", \"Dupont\", 25)";
if (mysql_query($requete) == 0)
    exit("Impossible d'exécuter la requête !");
?>
```

- *mysql_query()* retourne 0 ou 1 (échec ou succès)
- Dans le cas d'une requête SELECT, *mysql_query()* retourne un identifiant de résultat exploitable par d'autres fonctions

Parcours du résultat SELECT

- Plusieurs fonctions sont disponibles
 - ▶ *mysql_fetch_array()*
 - ▶ *mysql_fetch_assoc()*
 - ▶ *mysql_fetch_object*
 - ▶ *mysql_fetch_row()*

```
<?php
    $requete = "SELECT nom,prenom FROM
t_personne";
    if (($res = mysql_query($requete)) == 0)
        exit("Impossible d'exécuter la
requête !");
    while ($data = mysql_fetch_array($res))
        echo $data['nom']. ' '. $data['prenom'];
?>
```

```
<?php
    $requete = "SELECT nom,prenom FROM
t_personne";
    if (($res = mysql_query($requete)) == 0)
        exit("Impossible d'exécuter la
requête !");
    while ($data = mysql_fetch_row($res))
        echo $data[0]. ' '. $data[1];
?>
```

Fonctions MySQL (I/2)

- *mysql_close()* : ferme la connexion à la base
- *mysql_connect()* : établit une connexion avec la base spécifiée dans les arguments
- *mysql_create_db()* : permet de créer une nouvelle base de données
- *mysql_drop_db()* : permet de supprimer une base de données
- *mysql_error()* : retourne la description textuelle de la dernière erreur générée

Fonctions MySQL (2/2)

- *mysql_list_dbs()* : retourne le pointeur de résultat d'une liste de bases de données pour le serveur spécifié
- *mysql_list_tables()* : retourne le pointeur de résultat d'une liste de tables pour la base de données spécifiée
- *mysql_num_fields()* : retourne le nombre de champs dans un résultat
- *mysql_num_rows()* : retourne le nombre de rangées dans un résultat

Table des matières

1. Introduction
2. Les fonctionnalités du langage PHP
3. Passage et transmission de variables
4. Exploitation du SGBD MySQL
- 5. Accès à des fichiers**
6. Variables persistantes : *cookies* et *sessions*

Ouverture/fermeture d'un fichier

- La fonction *fopen()*

```
<?php
    //Ouverture d'un fichier
    $file = fopen ("mon_fichier.txt", "mode_ouverture");
?>
```

- La fonction *fclose()*

```
<?php
    //Fermeture d'un fichier
    $file = fclose ($file);
?>
```


Les modes d'ouverture

- 'r' : lecture seule - pointeur en début de fichier
- 'r+' : lecture/écriture - pointeur en début de fichier
- 'w' : écriture seule - pointeur en début de fichier et réduit la taille à 0 - création si le fichier n'existe pas
- 'w+' : lecture/écriture - pointeur en début de fichier et réduit la taille à 0 - création si le fichier n'existe pas
- 'a' : écriture seule - pointeur en fin de fichier - création si le fichier n'existe pas
- 'a+' : lecture/écriture - pointeur en fin de fichier - création si le fichier n'existe pas

Lecture d'un fichier (1/5)

- La fonction *fgets()* :
 - ▶ string **fgets** (resource \$handle, [int \$length])
 - ▶ pointeur retourné par *fopen()*
 - ▶ taille en octets de lecture (optionnel)

```
<?php
    $i = 1;
    $file = fopen ("mon_fichier.txt", "r");
    if ($file) {
        while (!feof($file)) { //tant qu'on est pas à la fin
            $line = fgets($file); //lecture ligne par ligne
            echo "Ligne $i : $line";
            $i++;
        }
        fclose($file);
    }
?>
```

Lecture d'un fichier (2/5)

- La fonction *fread()* :
 - ▶ string **fread** (resource \$handle , int \$length)
 - ▶ pointeur retourné par *fopen()*
 - ▶ taille en octets de lecture (optionnel)

```
<?php
//lit un fichier et le place dans une chaîne
$file = fopen ("mon_fichier.txt", "r");
$content = fread($file, filesize($file));
echo $content;
fclose($file);
?>
```

Lecture d'un fichier (3/5)

- La fonction *file_get_contents()* :
 - ▶ string **file_get_contents** (string \$filename, [bool \$use_include_path, [resource \$context, [int \$offset, [int \$maxlen]]]])
 - ▶ nom du fichier à lire
 - ▶ retourne le fichier dans une chaîne, à partir de la position *offset*, et jusqu'à *maxlen* octets

```
<?php
//Lit une page web et la copie dans une chaîne
$html = file_get_contents ("http://www.example.com");
echo $html;
?>
```

Lecture d'un fichier (4/5)

- La fonction *file()* :
 - ▶ array **file** (string \$filename [, int \$flags [, resource \$context]])
 - ▶ nom du fichier à lire
 - ▶ retourne le fichier dans un tableau : chaque élément du tableau correspond à une ligne du fichier
- Valeurs du paramètre *flags* :
 - ▶ `FILE_USE_INCLUDE_PATH` : recherche le fichier dans l'`include_path`
 - ▶ `FILE_IGNORE_NEW_LINES` : n'ajoute pas de nouvelle ligne à la fin de chaque élément du tableau
 - ▶ `FILE_SKIP_EMPTY_LINES` : ignore les lignes vides

Lecture d'un fichier (5/5)

- Utilisation de la fonction *file*

```
<?php
// Lit une page web dans un tableau.
$lines = file ('http://www.example.com/');

// Affiche toutes les lignes du tableau comme code HTML,
// avec les numéros de ligne
foreach ($lines as $line_num => $line) {
    echo 'Ligne No <strong>'.$line_num.'</strong>:'
        .htmlspecialchars($line).'<br />'. "\n";
}
?>
```

Ecriture dans un fichier (1/3)

- La fonction *fwrite()* :
 - ▶ int ***fwrite*** (resource \$handle , string \$string [, int \$length])
 - ▶ écrit le contenu de la chaîne *string* dans le fichier pointé par *handle*
 - ▶ l'écriture s'arrête après *length* octets, ou à la fin de la chaîne (le premier des deux)
 - ▶ retourne le nombre d'octets écrits ou FALSE en cas d'erreur.

Ecriture dans un fichier (2/3)

```
<?php
$filename = 'test.txt';
$somecontent = "Ajout de chaîne dans le fichier \n";
// Si le fichier est accessible en écriture
if (is_writable($filename)) {
    // Ouverture en mode ajout, pointeur à la fin du fichier
    if (!$handle = fopen($filename, 'a')) {
        echo "Impossible d'ouvrir le fichier ($filename)";
        exit;
    }
    // Ecriture dans le fichier
    if (fwrite($handle, $somecontent) === FALSE) {
        echo "Impossible d'écrire dans le fichier ($filename)";
        exit;
    }
    echo "L'écriture dans le fichier a réussi";
    fclose($handle);
} else {
    echo "Le fichier $filename n'est pas accessible en écriture.";
}
?>
```


Ecriture dans un fichier (3/3)

- La fonction *file_put_contents()* :
 - ▶ int **file_put_contents** (string \$filename, mixed \$data, [int \$flags, [resource \$context]])
 - ▶ revient à appeler les fonctions *fopen()*, *fwrite()*, et *fclose()* successivement
 - ▶ Possibilité de spécifier le paramètre *data* sous forme de tableau (tableau non multi-dimensionnel).
Equivalent à *file_put_contents(\$filename, implode("\$", \$array))*.
 - ▶ retourne le nombre d'octets écrits ou FALSE en cas d'erreur.

Fonctions diverses

- *bool **feof** (resource \$handle)* : retourne TRUE si le pointeur *handle* est à la fin du fichier ou si une erreur survient, sinon, retourne FALSE.
- *int **filesize** (string \$filename)* : renvoie la taille du fichier *filename*, ou FALSE en cas d'erreur
- *bool **file_exists** (string \$filename)* : retourne TRUE si le fichier *filename* existe, et FALSE sinon
- *bool **unlink** (string \$filename, [resource \$context])* : efface *filename*. Retourne TRUE en cas de succès, FALSE en cas d'échec.

Table des matières

1. Introduction
2. Les fonctionnalités du langage PHP
3. Passage et transmission de variables
4. Exploitation du SGBD MySQL
5. Accès à des fichiers
- 6. Variables persistantes : *cookies* et *sessions***

Inconvénients des variables “classiques”

- **Durée de vie limitée** : celle du script
- Passage de variables/paramètres par les méthodes POST et GET
 - ▶ **Contraignant** : codage par le développeur
 - ▶ **Sécurité** : informations disponibles pour l'internaute

Les cookies

- Mécanisme transparent d'enregistrement et de lecture d'informations sur le poste client
- Définition d'un cookie : **setcookie()**

```
int setcookie (string nom_variable, [string  
valeur_variable], [int expiration], [string chemin],  
[string domaine], [int securite])
```

nom_variable : nom de la variable à stocker

valeur_variable : valeur de la variable

expiration : durée avant l'expiration du cookie (en sec)

chemin : le chemin du répertoire où doit être lu le cookie

domaine : le nom du domaine

securite : le type d'entête (HTTP ou HTTPS)

- Doit impérativement être appelée avant tout affichage de texte

Ecriture, lecture et destruction de cookies

```
<?php
//Ecriture dans le cookie testCookie
$val = "ceci est la valeur du cookie";
if (setcookie("testCookie", $valeur, time() + 3600) == 0) {
    exit ("impossible de créer le cookie");
}

//Lecture du cookie testCookie
echo $HTTP_COOKIE_VARS["testCookie"];

//Destruction du cookie testCookie
if (setcookie("testCookie", "", time() - 100) == 0) {
    exit ("impossible de détruire le cookie");
}
?>
```

Limitation des cookies

- Le client a le pouvoir de le refuser (par configuration du navigateur)
- Usurpation d'identité : fichier texte facilement recopiable et modifiable sur une autre machine

Les sessions

- Mécanisme pour sauvegarder des variables
- A chaque visiteur est attribué un numéro unique (identifiant de session)
- Enregistrées sur le serveur même
- Une session est détruite lorsque
 - ▶ Aucune action (POST ou GET) n'est exécutée au-delà de la durée de vie définie
 - ▶ L'utilisateur ferme son navigateur
 - ▶ La commande `session_destroy()` est appelée

Démarrage de sessions

```
<?php
//Démarrage de l'environnement de session
session_start();

//Ecriture dans une variable de session
$_SESSION["prenom"] = "jean";
$_SESSION["nom"] = "dupont";
?>
```

- ***session_start()*** doit figurer dans chaque page (perpétue le transfert des variables)
- Identifiant de session : ***session_id()***
- Durée de vie de la session : ***session.cache_expire***

Lecture et destruction de variables de session

```
<?php
    session_start();
    //Lecture d'une variable de session
    $var = $_SESSION[ 'nom_variable' ];

    //Destruction d'une variable de la session
    unset($_SESSION[ 'nom_variable' ]);

    //Destruction de l'environnement de session
    session_destroy();
?>
```