Partie 4 - Le langage PHP

Olivier GLÜCK
Université LYON 1/UFR d'Informatique
Olivier.Gluck@ens-lyon.fr
http://www710.univ-lyon1.fr/~ogluck

Copyright

Remerciements

- Quelques transparents sont directement tirés des supports de cours de :
  - Dominique Bouillet (INT)
  - Eric Guerin (LYON 1)
- Merci à eux !
- Des figures et exemples sont issus des livres ou sites Internet cités en bibliographie

Bibliographie

- «*Webmaster in a nutshell*», S. Spainhour & R. Eckstein, 3ième édition, O'REILLY, ISBN 0-596-00357-9
- «*PHP Pocket Reference*», 2nd Edition November 2002, R. Lerdorf, O'REILLY, ISBN 0-596-00402-8
- «*PHP en action*», Edition Française, D. Sklar & A. Trachtenberg, O'REILLY, 2003, ISBN 2-84177-231-4
- «*Programming PHP*», March 2002, R. Lerdorf & K. Tatroe, O'REILLY, ISBN 1-56592-610-2
- Internet…
  - http://www.php.net (site officiel) ou http://fr.php.net (miroir français)
  - http://www.phpfrance.com/
  - http://www.hotscripts.com/PHP/
  - http://www.linuxguruz.com/
  - http://php.resourceindex.com/

Plan de la partie 4 (2 séances)

- Principes et introduction du langage PHP
  - Principes/Avantages de PHP, Inconvénients, Caractéristiques de PHP, Historique
- Mise en place de PHP
  - Installation et configuration, Activation de code PHP, Intégration de fichiers externes
- Les bases du langage
  - Types, Chaînes de caractères, Tableaux, Opérateurs, Instructions conditionnelles, Boucles, Fonctions, Variables, Objets

Plan de la partie 4 (2 séances)

- Fonctionnalités avancées
  - Authentification
  - Gestion des connexions HTTP
  - Gestions des sessions PHP
  - Gestions des cookies
  - Transmission d'une variable
  - Liens avec les bases de données : PHP/MySQL
  - Création dynamique d'image
  - Manipulation de fichiers/répertoires
- Conclusion

Principes et introduction de PHP

Principes/Avantages de PHP
Inconvénients
Caractéristiques de PHP
Historique

Qu'est-ce que PHP ?

- "PHP is a **server-side**, **HTML-embedded**, **cross-platform** scripting language"
  - langage interprété et indépendant de la plate-forme d'exécution,
  - qui s'exécute sur le serveur,
  - les instructions sont intégrées au code source d'une page HTML,
  - qui permet de générer des pages HTML dynamiques
- Le pendant de JavaScript côté serveur

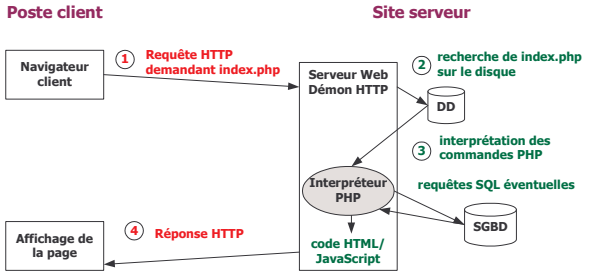
Exemple

```
<!-- 1.php -->
<HTML><HEAD>
        <TITLE>Premier exemple PHP</TITLE>
</HEAD><BODY>
<H2>Voici les informations que je connais à votre sujet</H2>
<?php
/* $HTTP_USER_AGENT sera remplacée par sa valeur
avant l'envoi de la réponse au client */
echo $_SERVER['HTTP_USER_AGENT'];
?>
</BODY></HTML>
```


Principe/Avantages (1)

- L'interpréteur de code PHP est intégré au serveur HTTP
- Le serveur lit les instructions PHP intégrées à la page HTML, interprète ces instructions et les remplace par le résultat de leur exécution
- Avantages
  - le client n'a pas accès au code source car il est interprété avant envoi (pas le cas de JavaScript)
  - le client ne reçoit que le résultat de l'exécution
  - la qualité dynamique des pages est masquée au client
  - le code n'est pas alourdi par des commandes destinées à générer du HTML (pas le cas de CGI)

Principe/Avantages (2)



Principe/Avantages (3)

- Fonctionnement de l'interpréteur PHP
  - un bloc PHP est un groupe de lignes encadré par deux balises <?php ?>
  - toute ligne située à l'extérieur de ces balises n'est pas interprétée : elle est recopiée à l'identique sur la sortie standard
  - toute ligne située à l'intérieur de ces balises est interprétée par le moteur comme une instruction PHP
  - les instructions PHP n'apparaissent pas dans le flux de sortie généré
  - lorsqu'une erreur se produit, un message explicatif est intégré dans le flux de sortie et l'analyse du code est interrompue (sauf si *warning*)

Inconvénients

- Trous de sécurité
  - idem scripts CGI : on exécute quelque chose sur le serveur
  - il faut bien écrire ses scripts, prévoir tous les cas de saisie du client
- Rapidité d'exécution
  - langage interprété par le serveur avec plusieurs requêtes simultanées donc pas très rapide
  - tend à s'améliorer grâce à des optimisations au niveau de l'interpréteur
- Pas d'interactivité au niveau du client (JavaScript reste nécessaire pour cela)

Caractéristiques de PHP (1)

- Très populaire et très utilisé
- Très portable (fonctionne sous Windows/Unix…)
- Syntaxe héritée du C, du shell et du Perl
- Extensible par de nombreuses bibliothèques
  - calcul mathématique,
  - création dynamique d'images,
  - connexions sécurisées,
  - accès aux bases LDAP,
  - accès à la plupart des SGBD
- Logiciel Open Source (donc plus facilement extensible) et disponible gratuitement

Caractéristiques de PHP (2)

- Supporte pratiquement tous les standards du WEB (pdf, manipulation d'URLs et de paramètres de connexions HTTP, formulaires, IMAP, SMTP, …)
- Nombreux FAQ, tutoriaux, forum disponibles en ligne
- Un fichier PHP (.php) peut contenir
  - du code HTML
  - du code PHP
  - du code JavaScript
- Conçu pour fonctionner efficacement avec le serveur Apache lui aussi en *open source*

PHP et CGI

- Tous les deux s'exécutent sur le serveur mais
  - le code PHP est contenu dans la page HTML

```
<?php $username = "toto"; ?>
<HTML><BODY><H1>
Bonjour Monsieur <?php echo $username; ?>
</H1></BODY></HTML>
```

  - le code HTML est contenu dans le code CGI

```
#! /usr/bin/perl
$username = "toto";
print "Content-type: text/html\n\n";
print "<HTML><BODY><H1>\n";
print "Bonjour Monsieur $username\n";
print "</H1></BODY></HTML>\n";
```


Les semblables de PHP

- JSP : Java-Server Pages
  - semblable à PHP mais la partie dynamique est écrite en JAVA
- ASP : Active Server Pages
  - version Microsoft de PHP et JSP
  - contenu dynamique écrit en Visual Basic Script, langage de script propriétaire de Microsoft
- Le choix entre PHP, JSP et ASP est plus "politique" que technique !
  - PHP : open source
  - JSP : Java Sun
  - ASP : Microsoft

Historique (1)

- 1994 : créé par Rasmus Lerdorf pour ses besoins personnels
  - très succinct
  - savoir qui venait consulter son CV en ligne
- 1995 : première version publique
  - *Personnal Home Page Tools*
  - interpréteur du code très simple (quelques macros) et quelques utilitaires
  - gestion d'un livre d'or
  - gestion d'un compteur d'accès

Historique (2)

- Eté 1995 : deuxième version
  - PHP/FI version 2
  - réécriture de l'interpréteur
  - ajout de la gestion des formulaires (*Form Interpreter*)
  - support pour mSQL
- Eté 1997 : plus de 50000 sites Web utilisent PHP/FI
- Fin 1997 : PHP devient un projet d'équipe
  - réécriture complète de l'interpréteur par Zeev Suraski et Andi Gutmans
  - donne naissance à PHP3

Historique (3)

- Eté 1999 : succès de PHP3
  - 150 000 sites Web utilisent PHP3
  - PHP : **H**ypertext **P**re**P**rocessor
- 2000 : PHP4
  - le projet est maintenant chapeauté par Zend, société privée créée par Suraski et Gutmans qui commercialise des logiciels complémentaires à PHP
  - interpréteur optimisé par Zend et ouverture de PHP à d'autres serveurs HTTP qu'Apache
- Aujourd'hui, plus de 5 000 000 de sites Web dans le monde entier

Mise en place de PHP

Installation et configuration
Activation de code PHP
Intégration de fichiers externes

Installation de PHP

- Trois types d'installation de PHP
  - en tant que module dynamique du serveur HTTP (mod_php pour Apache)
    - configuration la plus courante
    - le module se charge chaque fois que cela est nécessaire
  - en tant que module statique du serveur HTTP
    - un peu plus performant mais nécessite la recompilation du serveur HTTP
  - en tant que CGI
    - du CGI avec comme langage PHP
    - pour les serveurs HTTP qui ne supportent pas PHP

Configuration d'Apache

- PHP comme module dynamique intégré à Apache
- Configuration PHP4.x dans *httpd.conf*

```
AddType application/x-httpd-php .php
```

  - les fichiers .php seront alors analysés pour que les instructions PHP soient interprétées par le parser intégré au module PHP d'Apache

```
LoadModule php4_module modules/libphp4.so
```

  - permet le chargement dynamique du module PHP quand cela est nécessaire

Vérification du bon fonctionnement

```
<!-- phpinfo.php -->
<!-- Vérifier que PHP fonctionne sur le serveur -->
<?php phpinfo(); ?>
```

- phpinfo() permet d'afficher les informations relatives à la configuration de PHP sur le serveur
  - options de compilation, extensions, version, informations sur le serveur, environnement PHP, chemins, utilisateur, en-têtes HTTP, et licence GNU Public License
  - contenu de toutes les variables EGPCS (Environnement, GET, POST, Cookie, Serveur)

Le fichier de configuration de PHP

- Le fichier php.ini
  - généralement dans /etc/php<version>/apache
  - contrôle les principaux comportements de PHP
  - très clair et très facilement configurable
  - syntaxe : clef = "valeur"
    - ; pour les commentaires
    - valeur booléenne :
      - vrai = 1, On, True ou Yes
      - faux = 0, Off, False ou No
  - segmenté en parties : options du langage, ressources maximales allouées à PHP, gestion des erreurs, gestion des données HTTP, fichiers et répertoires, ajout et configuration des modules PHP (mail, SGBD, debugger, …)

Intégrer du PHP dans une page HTML

- La façon la plus portable

```
<?php echo "Bonjour !"; ?>
```

- Autre méthode tout autant portable

```
<script language="php">echo "Bonjour !";</script>
```

- On trouve aussi couramment

```
<? echo "Bonjour !"; ?>
```

mais pose un problème de compatibilité avec XML (il faut short_open_tag = On dans php.ini)

```
<% echo "Bonjour !"; %>
```

désactivé par défaut mais utilisé par Microsoft FrontPage (style ASP) (il faut asp_tags = On dans php.ini)

Intégrer du PHP dans une page HTML

- Le ; est le séparateur d'instructions

```
<?php
  echo "Bonjour Monsieur";
  echo $nom;
?>
```

le ; peut être omis juste avant ?> mais pas conseillé !

- Entrelacement de code HTML et PHP

```
<?php for ($i=0; $i<100; $i++) { ?>
  <br>
<?php } ?>
```


Intégration de fichiers externes (1)

- PHP a été pensé pour la conception d'applications Web
  - permet de définir des "briques de base" qui peuvent être réutilisées plusieurs fois
  - permet de répercuter facilement un changement sur l'ensemble de l'application
- Exemple : chaque page commence par

```
<?php $titre="Rubrique 1"; include "header.inc"; ?>
```

avec le fichier header.inc qui contient

```
<HTML><HEAD><TITLE>
  <?php echo $titre; ?>
</TITLE></HEAD>
```


Intégration de fichiers externes (2)

- Le fichier qui est inclus hérite de l'ensemble des variables du fichier qui effectue l'inclusion
- Les fichiers inclus sont analysés comme du code HTML qui peut contenir du code PHP
- L'extension .inc (choisie arbitrairement ici) n'est pas reconnue comme du PHP par le serveur
  - problème : un utilisateur qui exécuterait la requête http://…/header.inc pourrait voir le code source PHP inclus dans le fichier
  - -> mettre une extension .php mais alors les utilisateurs pourraient directement exécuter les fichiers inclus (header.php dans notre exemple)
  - -> interdire au niveau de httpd.conf les requêtes sur les fichiers ayant .inc comme extension

Intégration de fichiers externes (3)

- Autres instructions d'inclusion
  - require : similaire à include mais provoque une erreur fatale si le fichier n'existe pas (include émet seulement un warning)
  - include_once et require_once : similaire à include (et require) mais en s'assurant que le fichier n'a pas déjà été inclus -> résout les problèmes de définitions multiples de fonctions…

Les bases du langage PHP

Types, chaînes de caractères, tableaux, opérateurs, instructions conditionnelles, boucles, fonctions, variables, objets

Commentaires et casse

- On peut mélanger les commentaires du C, C++ et shell !

```
<?php /* commentaire type C */
// commentaire type C++
# commentaire type shell
?>
```

- La casse n'intervient pas dans les noms de fonction
  - `echo "Bonjour";` et
  - `EcHo "Bonjour";` sont équivalents
- En revanche, elle intervient dans les noms de variables
  - `echo $nom;` et
  - `echo $NOM;` concernent deux variables différentes

Types de données et variables

- Les noms de variables sont précédés d'un $

```
$i = 5;
echo "$i";
```

- Pas de déclaration, l'affectation détermine le type de la variable
- PHP supporte les types de données suivants
  - booléens
  - nombres entiers
  - nombres à virgule flottante
  - chaînes de caractères
  - tableaux
  - objets (programmation orientée objet)

Booléens, entiers et flottants

- Booléens
  - true ou false
  - sont considérés comme faux : le booléen FALSE, l'entier 0, le flottant 0.0, "" ou "0", un tableau vide, un objet vide, la constante NULL
- Entiers positifs ou négatifs

```
$i = -5; # base 10
$i = 012; # base 8
$i = 0x12; # base 16
```

  - pas de division entière mais transtypage (cast) possible

```
$i = (int) (7/3) # $i vaut 2
```

- Les flottants

```
$f = -1.3;
$f = 1.2e-3; # 0,0012
```


Chaînes de caractères (1)

- Chaînes de caractères

```
$ch1 = 'Bonjour'; $ch2 = "$ch1 Monsieur !\n";
```

  - entre guillemets simples, rien n'est interprété sauf \' et \\ qui échappent le caractère ' et le caractère \
  - entre guillemets doubles, sont interprétés les variables, les caractères spéciaux \n \r \t \$ \" \\
  - protection des noms de variables

```
$ch1 = 'Bonjour '; $ch2 = "${ch1}Monsieur !\n";
```

  - `echo "\n";` provoque un saut de ligne dans le code HTML généré mais ne provoque de saut de ligne HTML (il faut utiliser
) !

Chaînes de caractères (2)

- Conversion de chaînes de caractères en valeur numérique

```
$i = 1 + "4.5"; # $i vaut 5.5
$i = 1 + "-1e3"; # $i vaut -999
$i = 1 + "toto + 9"; # $i vaut 1
$i = 1 + "9 + toto"; # $i vaut 10
```

  - la chaîne est de type flottant si elle contient '.', 'e' ou 'E' sinon elle est de type entier
  - la valeur est définie par la première partie de la chaîne (0 si c'est du texte)
- La constante NULL est convertie en "", un tableau en la chaîne "Array"

```
$i = NULL; echo "-$i-"; # affiche "--"
$tab[0] = 1; echo "$tab"; # affiche "Array"
```

Chaînes de caractères (3)

- Accès aux caractères d'une chaîne

```
$ch = 'Ceci est une chaîne.'; echo $ch{3}; # affiche "i"
```

- Affichage de chaînes
  - echo et print pour les affichages simples
  - printf pour les chaînes formatées (comme en C)
  - print_r et var_dump pour les variables complexes

```
$nom = "Monsieur";
echo "Bonjour", " $nom"; # affiche "Bonjour Monsieur"
print "Bonjour $nom"; # affiche "Bonjour Monsieur"
printf("Bonjour %s", $nom); # affiche "Bonjour Monsieur"
```


Chaînes de caractères (4)

- Substitutions de chaînes
  - addslashes(str) ajoute un \ devant tous les caractères spéciaux de la chaîne passée en argument - utile pour les requêtes aux bases de données
  - stripslashes(str) effectue l'opération inverse (enlève les \)
  - str_replace(search, replace, str) remplace dans str toutes les occurrences de search par replace

```
# la commande suivante affiche "Bonjour titi"
echo str_replace("toto", "titi", "Bonjour toto");
```


Chaînes de caractères (5)

- Découpage de chaînes
  - explode(sep, str) retourne un tableau de chaînes en scindant str à l'aide du séparateur sep
  - implode(sep, tab) retourne une chaîne fabriquée par la concaténation des éléments du tableau et du séparateur sep entre chaque élément (join() fait la même chose)
  - chop(str) ou rtrim(str), ttrim(str), ltrim(str) suppriment des caractères d'espacement dans str (" ", \t, \n, \r, \0)
  - parse_str(str) analyse une chaîne type QUERY_STRING et fabrique les couples $nom=valeur associés
- Comparaison et longueur
  - strcmp(str1, str2) : comparaison binaire (comme en C)
  - strlen(str) retourne la longueur de str

Chaînes de caractères (6)

- Web
  - rawurlencode(str)/rawurldecode(str) encode/décode la chaîne str selon la RFC1738 (%xy <-> caractère)
  - urlencode(str)/urldecode(str) idem mais adapté au formulaire (remplace également le + en espace…)
  - base64_encode(str)/base64_decode(str) encode/décode la chaîne en type MIME base64 (par exemple pour user:passwd)
  - parse_url(str) retourne dans un tableau les composants d'une URL (scheme, host, port, user, pass, path, query, fragment)
    http://username:password@hostname/path?arg=value#anchor

Chaînes de caractères (7)

- Expressions régulières (substitution et recherche complexes)
  - PHP gère les expressions régulières POSIX et celles du langage Perl
    - ^ : début de ligne
    - $: fin de ligne
    - . : n'importe quel caractère
    - [A-Z] : caractère appartenant à l'ensemble des majuscules
    - [^A-Z] : caractère n'appartenant pas à l'ensemble des majuscules
    - r* : 0, 1 ou plusieurs occurrences successives de l'expression régulière r

Chaînes de caractères (8)

- Expressions régulières (substitution et recherche complexes)
  - PHP gère les expressions régulières POSIX et celles du langage Perl
    - r{m,n} : entre m et n occurrences successives de l'expression régulière r
    - r{m} : exactement m occurrences successives de l'expression régulière r
    - r{m,} : au moins m occurrences successives de l'expression régulière r
    - \ : caractère d'échappement (m*n = m*n)
    - r+ : un ou plusieurs exemplaires de r
    - (r1|r2|r3) : r1 ou r2 ou r3

Chaînes de caractères (9)

- Expressions régulières (substitution et recherche complexes)
  - ereg(profil,str,$tab) recherche le profil dans str
    - retourne true ou false
    - remplit $tab avec les occurrences trouvées (max 10)
    - sensible à la casse

```
# $date est au format YYYY-MM-DD
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
  # $regs[0] contient une copie de la chaîne trouvée
  # affichage au format DD/MM/YYYY
 echo "$regs[3]/$regs[2]/$regs[1]"; } else {
 echo "Format de date invalide : $date"; }
```

  - eregi(profil,str,$tab) idem mais insensible à la casse

Chaînes de caractères (10)

- Expressions régulières (substitution et recherche complexes)
  - ereg_replace(profil,replace,str) recherche le profil dans str et le remplace par replace
    - retourne str modifiée
    - sensible à la casse
  - eregi_replace(profil,replace,str) idem mais insensible à la casse
  - split(profil,str) retourne un tableau de sous-chaînes de str délimitées par le profil

```
# MM/DD/YYYY ou MM.DD.YYYY ou MM-DD-YYYY
list ($month, $day, $year) = split ('[/.-]', $date);
```

  - spliti(profil,str) idem mais insensible à la casse

Date et heure

- Afficher la date et l'heure actuelle

```
echo "Nous sommes le " . date("j m Y") . " et il est "
  .date("H \h i") . " mn";
```

- Les fonctions date() et mktime()
  - date("format", $timestamp) retourne une chaîne de caractères qui contient la date $timestamp au format indiqué en premier argument
  - $timestamp est la date Unix (nombre de secondes depuis le 1er janvier 1970)
  - si $timestamp est omis, il s'agit de la date actuelle
  - si le format est omis, date() retourne $timestamp
  - mktime(h, m, s, M, J, A) retourne $timestamp associé à la date indiquée en paramètres

Tableaux (1)

- Les tableaux de PHP sont associatifs
  - l'index dans le tableau est appelé clé
  - la valeur associée à une clé est appelée valeur
  - un tableau est un ensemble d'associations clé/valeur
  - la clé peut être un entier ou une chaîne de caractères
- Création d'un tableau
  - soit directement en affectant des valeurs au tableau
  - soit en utilisant la fonction array()

```
$tab[0] = 1; # clé entière, valeur entière
$tab[1] = "toto"; # clé entière, valeur de type chaîne
$tab["toto"] = "titi"; # clé et valeur de type chaîne
```


Tableaux (2)

- Autres exemples

```
$tab["toto"][0] = 1; # tableau à 2 dimensions
$tab["toto"][0]["titi"][3] = "toto"; # tout est permis !
$tab[] = "tata"; # la clé sera un entier dont la valeur est la
  clé entière la plus grande du tableau + 1
$tab1 = array(1, "toto"); # 0=>1 et 1=>"toto"
$tab2 = array("toto"=>1, "titi"=>"toto"); # la clé est donnée
```

- Nombre d'éléments d'un tableau

```
sizeof($tab); count($tab); # retourne le nombre d'éléments
  du tableau $tab s'il existe, 1 si $tab n'est pas un tableau,
  0 si $tab n'existe pas
```


Tableaux (3)

- Suppression d'un élément

```
unset($tab["toto"]); # fonctionne aussi pour une variable
```

- Tris de tableaux
  - le tri peut se faire sur les clés et/ou les valeurs ; l'association clé/valeur peut être cassée
  - asort()/arsort() : trie le tableau par ordre croissant/décroissant de valeurs
  - ksort()/krsort() : trie le tableau par ordre croissant/décroissant de clés
  - sort() : trie le tableau par ordre croissant de valeurs et réassigne des clés (0,1,...) ; on perd l'association clé/valeur
  - uasort()/uksort()/usort() : identiques à leur homologue mais la fonction de comparaison est fournie

Tableaux (4)

- Le pointeur de tableau
  - à chaque tableau correspond un pointeur interne qui est une référence sur l'élément courant
  - current($tab) désigne l'élément courant
  - next($tab) déplace le pointeur vers l'élément suivant
  - prev($tab) déplace le pointeur vers l'élément précédent
  - end($tab) déplace le pointeur sur le dernier élément
  - reset($tab) déplace le pointeur sur le premier élément

```
$tab = array("a"=>1, "d"=>5, "b"=>8, "c"=>4);
$val = current($tab); echo "$val<br>"; # affiche "1<br>"
$val = next($tab); echo "$val<br>"; # affiche "5<br>"
asort($tab); end($tab); prev($tab); $val = prev($tab);
echo "$val<br>"; # affiche "4<br>"
```


Tableaux (5)

- Extraction d'éléments d'un tableau
  - list() permet d'extraire des valeurs d'un tableau

```
$tab = array(1, 8, 5); sort($tab);
list($v1, $v2) = $tab;
echo "$v1 $v2"; # affiche "1 5"
```

  - key($tab) permet d'extraire la clé de l'élément pointé par le pointeur interne du tableau

```
$tab = array("a"=>1,"b"=>8); next($tab);
$clé = key($tab); $val = $tab[$clé];
echo "$clé: $val"; # affiche "b: 8"
```

  - extract($tab) permet d'extraire d'un tableau toutes les valeurs, chaque valeur est recopiée dans une variable ayant pour nom la valeur de la clé

```
$tab = array("a"=>1,"b"=>8); extract($tab);
echo "$b $a"; # affiche "8 1"
```


Tableaux (6)

- Extraction d'éléments d'un tableau
  - each($tab) retourne la paire clé/valeur courante du tableau et avance le pointeur de tableau ; cette paire est retournée dans un tableau de 4 éléments : 0=>clé, 1=>valeur, key=>clé et value=>valeur
- Parcours de l'ensemble du tableau
  - avec list() et each()

```
$tab = array("a"=>1, "d"=>5, "b"=>8, "c"=>4);
reset ($tab);
while (list ($k, $v) = each ($tab)) {
echo "$k => $v<br />\n"; }
#quand le pointeur dépasse la fin de $tab, each retourne false
```


Tableaux (7)

- Parcours de l'ensemble du tableau
  - foreach() place le pointeur en tête du tableau et parcourt l'ensemble des éléments ; foreach() travaille sur une **copie** du tableau original

```
$tab = array("a"=>1, "d"=>5, "b"=>8, "c"=>4);
# parcours des valeurs uniquement : $v = 1, 5, 8, 4
foreach ($tab as $v) { echo "valeur : $v<br>\n"; }
# parcours des couples clé/valeur
foreach ($tab as $k => $v) { echo "$k : $v<br>\n"; }
# tableaux multidimensionnels
$tab2[0][0] = "a"; $tab2[0][1] = "b";
$tab2[1][0] = "y"; $tab2[1][1] = "z";
foreach($tab2 as $t) { foreach ($t as $v) { echo "$v\n"; } }
```


Les constantes

- Constantes définies par le programmeur

```
define ("MAX", 255); echo MAX;
define ("NOM", "Durand"); echo NOM;
```

- Principales constantes définies par PHP
  - __FILE__ : chemin absolu du fichier en cours d'exécution
  - __LINE__ : numéro de la ligne en cours d'exécution
  - PHP_VERSION : version de PHP qui est utilisée
  - PHP_OS : OS de la machine (ex : Linux)
  - TRUE et FALSE
  - E_* : gestion des erreurs (E_ALL = toutes les erreurs)
  - il y en a plein d'autres !

Les erreurs

- 4 types d'erreurs/alertes en PHP
  - E_ERROR (1) : erreur d'exécution
  - E_WARNING (2) : alerte
  - E_PARSE (4) : erreur d'analyse
  - E_NOTICE (8) : notes (alertes pouvant être ignorées)
- Changement du niveau d'erreurs
  - error_reporting(cst) permet de changer le niveau d'erreurs reportées (par défaut 7=1+2+4) dans le script
  - directive error_reporting dans php.ini ou httpd.conf
- error_log() permet d'envoyer un message d'erreur dans les logs du serveur web, à un port TCP, dans un fichier, par mail, ...

La valeur NULL

- Constante spéciale qui représente l'absence de valeur
- Valeur insensible à la casse, introduite en PHP4

```
$var = NULL;
isset($var) # retourne FALSE
is_null($var) # retourne TRUE
empty($var) # retourne TRUE
$ch = "";
isset($ch) # retourne TRUE
is_null($ch) # retourne FALSE
empty($ch) # retourne TRUE
```


Les opérateurs (1)

- Opérateurs arithmétiques
  - addition : $a + $b
  - soustraction : $a - $b
  - multiplication : $a * $b
  - division : $a / $b
  - modulo (reste de la division entière) : $a % $b

```
$i = 7/3; echo "<br>$i"; # affiche 2.3333333333333
$j = 7%3; echo "<br>$j"; # affiche 1
```

- Concaténation de chaîne de caractère (.)

```
$ch1 = "Bonjour "; $ch2 = "Monsieur";
$ch = $ch1.$ch2
echo $ch . " Durand" . ' !';
# affiche "Bonjour Monsieur Durand !"
```


Les opérateurs (2)

- Opérateurs binaires
  - ET bit à bit : $a & $b
  - OR bit à bit : $a | $b
  - XOR bit à bit : $a ^ $b
  - NON bit à bit : ~$a
  - décalage à droite de $b bits : $a >> $b
  - décalage à gauche de $b bits : $a << $b
- Opérateurs logiques
  - ET logique : and, && (les deux sont possibles)
  - OU logique : or, ||
  - XOR logique : xor
  - NON logique : !

Les opérateurs (3)

- Opérateurs d'affectation
  - affectation avec le signe =

```
$a = ($b = 4) + 1; # $b vaut 4 et $a vaut 5
```

  - les opérateurs combinés : +=, -=, *=, /=, .=, &=, |=, ^=, <<=, >>=, ~=

```
$a += 1; # équivalent à $a = $a + 1;
$ch .= "!"; # équivalent à $ch = $ch . "!";
```

  - ++ est équivalent à += 1, -- est équivalent à -= 1

```
$i = 0;
echo ++$i; # incrémente $i puis affiche 1
echo $i++; # affiche 1 puis incrémente $i
echo $i; # affiche 2
```


Les opérateurs (4)

- Opérateurs de comparaison
  - égal à : $a == $b
  - différent de : $a != $b
  - inférieur à : $a < $b
  - supérieur à : $a > $b
  - inférieur ou égal à : $a <= $b
  - supérieur ou égal à : $a >= $b
- Opérateur ternaire
  - (condition) ? (expr1) : (expr2);
  - renvoie expr1 si condition est vraie sinon renvoie expr2

```
$max = ($a >= $b) ? $a : $b; # $max <- max($a,$b)
```


Instructions conditionnelles (1)

- if/then/else comme en C

```
if (condition1) {
  # si condition1 est vraie
} elseif (condition2) {
  # si condition2 est vraie (et pas condition1)
} elseif (condition3) {
  # si condition3 est vraie (et ni condition1, ni condition2)
} else {
  # si ni condition1, ni condition2, ni condition3 ne sont vraies
}
```

Instructions conditionnelles (2)

- if/then/else avec prototype simplifié

```
if (condition1) :
  # si condition1 est vraie
elseif (condition2) :
  # si condition2 est vraie (et pas condition1)
elseif (condition3) :
  # si condition3 est vraie (et ni condition1, ni condition2)
else :
  # si ni condition1, ni condition2, ni condition3 ne sont
   vraies
endif;
```


Instructions conditionnelles (3)

- Le switch comme en C

```
switch(expr) {
case (val1) :
  # à exécuter si expr vaut val1
break;
case (val1) :
  # à exécuter si expr vaut val2
break;
default :
  # à exécuter dans tous les autres cas
}
```


Les boucles (1)

- Boucle "tant que"
  - comme en C

```
while (condition) {
  # exécuté tant que la condition est vraie
}
```

  - prototype simplifié

```
while (condition) :
  # exécuté tant que la condition est vraie
endwhile;
```

- Boucle "do…while" comme en C

```
do { /*…*/ } while (condition);
```


Les boucles (2)

- Boucle "for"
  - comme en C

```
for ($i = 1; $i <= 10; $i++) {
  # exécuté tant que ($i <= 10) est vraie
}
```

  - prototype simplifié

```
for ($i = 1; $i <= 10; $i++) :
  # exécuté tant que ($i <= 10) est vraie
endfor;
```

- L'instruction break permet de sortir d'une boucle
- L'instruction continue permet de passer directement à l'itération suivante de la boucle

Les fonctions (1)

- En PHP3, une fonction doit être définie avant d'être utilisée ; ca n'est plus le cas en PHP4

```
function max($a, $b) {
  # corps de max()
  if ($a>$b) { return $a; } else { return $b; }
}
# appel de la fonction max()
$m = max(5, 8); # $m vaut 8
```

- Il est possible de retourner n'importe quel type de variable (tableau, objet, …)
- Une fonction peut ne rien retourner

Les fonctions (2)

- Les arguments de fonction - PHP supporte :
  - le passage d'arguments par valeur (par défaut)
  - le passage d'arguments par référence
  - les valeurs par défaut des arguments
  - un nombre variable d'arguments (PHP4 uniquement)
- Deux types de passage par référence
  - de façon permanente en ajoutant un & devant le nom de la variable dans la définition de la fonction
  - de façon ponctuelle en ajoutant un & devant le nom de la variable lors de l'appel de la fonction

Les fonctions (3)

```
function plus3($a) {
  $a += 3;
}
function plus5(&$a) {
  $a += 5;
}
# passage par valeur
$x = 1; plus3($x); echo $x; # affiche "1"
# passage par référence ponctuel
$y = 1; plus3(&$y); echo $y; # affiche "4"
# passage par référence "forcé"
$z = 1; plus5($z); echo $z; # affiche "6"
```


Les fonctions (4)

- PHP permet de définir comme en C++ des valeurs par défaut pour les arguments
  - la valeur par défaut d'un argument doit être une constante (ni une variable, ni un membre de classe)
  - les arguments avec valeur par défaut doivent être placés après les arguments sans valeur par défaut

```
function concat($ch1, $ch2 = "beau") {
  return $ch1.$ch2;
}
echo concat("Il fait "); # affiche "Il fait beau"
echo concat("Il fait ", "froid"); # affiche "Il fait froid"
```


Les fonctions (5)

- PHP4 supporte les fonctions à nombre d'arguments variable
  - func_num_args() retourne le nombre d'arguments passé à la fonction
  - func_get_arg(arg_num) retourne l'argument à la position arg_num dans la liste des paramètres (à partir de 0) ; retourne false si arg_num est trop grand
  - func_get_args() retourne un tableau qui contient la liste des arguments

```
function foo() {
  $numargs=func_num_args(); $arg_list=func_get_args();
  for ($i = 0; $i < $numargs; $i++) {
  echo "L'argument $i est: " . $arg_list[$i] . "<br>\n"; }
}
foo(1, 2, 3); foo("toto");
```


Les fonctions (6)

- PHP4 permet le retour de fonction par référence

```
function &retourne_reference() {
  return $une_ref;
} # $new_ref est une référence sur $une_ref
$new_ref =& retourne_reference();
```

- Les variables fonctions

```
function call_back($arg = 'dans call_back') {
echo "$arg"; }
$func = 'call_back';
$func(); # affiche "dans call_back"
$func("func est call_back"); # affiche "func est call_back"
```


Les variables (1)

- Les noms de variables ne doivent pas commencer par un chiffre
- PHP4 permet des affectations par référence

```
$x = 2; $y = &$x; $y++; echo $x; #affiche 3
```

- Variables prédéfinies
  - depuis PHP 4.1.0, des tableaux "super-globaux" sont prédéfinis
    - accessibles dans n'importe quel contexte d'exécution
    - infos sur les variables du serveur, les variables d'environnement, les variables de PHP, ...
  - le contenu de ces tableaux dépend de la version et de la configuration du serveur et de PHP

Les variables (2)

- Variables prédéfinies - tableaux globaux PHP4.1.0
  - $GLOBALS : variables globales de l'exécution en cours
  - $_SERVER : variables fournies par le serveur WEB
    - 'PHP_SELF' (chemin du script sur le serveur),
    - 'argc', 'argv' (exécution en ligne de commande ou GET),
    - les variables d'environnement CGI
  - $_GET : variables fournies par HTTP/méthode GET
  - $_POST : variables fournies par HTTP/méthode POST
  - $_COOKIE : variables issues des cookies HTTP reçus
  - $_FILES : variables fournies par HTTP suite à un téléchargement de fichier(s) par la méthode POST

Les variables (3)

- Variables prédéfinies - tableaux globaux PHP4.1.0
  - $_ENV : variables d'environnement positionnées au démarrage du serveur Web
  - $_REQUEST : tableau associatif constitué de toutes les variables en entrée (contient en particulier $_GET, $_POST, $_COOKIE, $_FILES)
  - $_SESSION : variables relatives à la session en cours
  - avant PHP4.1.0 (toujours disponibles mais obsolète) :
    - $HTTP_SERVER_VARS, $HTTP_GET_VARS, $HTTP_POST_VARS, $HTTP_COOKIE_VARS, $HTTP_POST_FILES, $HTTP_ENV_VARS, $HTTP_SESSION_VARS
    - pas super-globaux -> global $HTTP_GET_VARS
  - Si register_globals = on dans php.ini, les variables prédéfinies sont directement accessibles par leur nom (ex: echo $PHP_SELF;)

Les variables (4)

- import_request_variables(type, prefix) permet d'importer les variables de _GET, _POST, ou _COOKIE en tant que variables globales (type = 'G', 'P' ou 'C') : utile si register_globals = off
- Variables globales

```
<?php $a = 1; include "file1.inc"; ?>
```

  - $a est globale ; elle est accessible dans file1.inc
- Variables locales à une fonction

```
<?php
$a = 1; # portée globale
function affiche() {
    echo $a; # $a est locale à la fonction
}
affiche(); # n'affiche rien !
?>
```


Les variables (5)

- Accès à une variable globale dans un bloc
  - déclarer la variable comme global dans le bloc (crée une référence locale sur la variable globale)
  - utiliser le tableau associatif $GLOBALS

```
<?php $a1 = 1; $a2 = 2; # portée globale
function affiche() {
    global $a1; # $a1 est la variable globale
    echo $a1." et ".$GLOBALS['a2'];
    $a1++;
}
affiche(); # affiche "1 et 2"
echo $a1; # affiche "2"  ($a1 a été modifiée)
?>
```


Les variables (6)

- Variables statiques dans un bloc
  - une variable statique a une portée locale à un bloc mais conserve sa valeur entre deux exécutions du bloc
  - elles sont utiles pour les fonctions récursives

```
function up_to_10() {
  static $cpt = 0;
  $cpt++;
  echo "$cpt ";
  if ($cpt<10) up_to_10();
}
up_to_10(); # affiche 1 2 3 4 5 6 7 8 9 10
```


Les variables (7)

- Variables dynamiques
  - une variable dynamique prend la valeur d'une variable et l'utilise comme nom d'une autre variable

```
$nom_var = "x"; $$nom_var = 5;
echo "$nom_var $x"; # affiche "x 5"
echo "$nom_var ${$nom_var}"; # affiche aussi "x 5"
```

  - ambiguïté avec les tableaux : $$a[1] peut représenter ${$a}[1] ou bien ${$a[1]} (il faut utiliser les accolades)

Les variables (8)

- Affichage des variables complexes
  - var_dump() affiche les informations structurées d'une variable (type, valeur, ...)
  - print_r() affiche les informations lisibles pour une variable (par exemple affiche les clés et les valeurs d'un tableau)
  - attention : print_r place le pointeur de tableau à la fin

```
$a = array (1, 2, "a"); var_dump ($a);
/* affichage :
array(3) {
  [0]=> int(1)
  [1]=> int(2)
  [2]=> string(1) "a"
} */
```

Les variables (9)

- Variables et types
  - is_array(), is_bool(), is_callable(), is_double(), is_float(), is_int(), is_integer(), is_long(), is_null(), is_numeric(), is_object(), is_real(), is_resource(), is_scalar(), is_string(), gettype(), settype()
  - gettype($var) retourne le type de $var : integer, double, string, array, object, ou unknown type

```
$foo = "5bar"; // chaîne
$bar = true;   // booléen
settype($foo, "integer"); // $foo vaut maintenant 5
settype($bar, "string");  // $bar vaut maintenant "1"
```


Les transtypages

- Comme en C - les transtypages possibles sont
  - (int) ou (integer) pour le type entier
  - (bool) ou (boolean) pour le type booléen
  - (real), (double) ou (float) pour le type réel
  - (string) pour le type chaîne
  - (array) pour le type tableau
  - (object) pour le type objet
- Attention : le transtypage n'a pas toujours une valeur prévisible

Les objets (1)

- Définition d'une classe

```
class Caddie {
   # attributs
   var elements;
   var owner;
   # constructeur
   function Caddie() {
     $this->owner = $_POST['prenom']. $_POST['nom'];
     $this->ajoute(1, "cadeau");
   }
   # méthode qui ajoute $nb articles de type $type
   function Ajoute($nb, $type) {
     $this->elements[$type] += $nb;
   }
}
```


Les objets (2)

- Création d'un objet de la classe

```
$client = new Caddie; # appel le constructeur
```

- Accès aux attributs et méthodes

```
echo $client->owner;
$client->Ajoute(2, "cadeau");
```

- Héritage de classe

```
class Caddie_prix extends Caddie {
   var $prix; var $les_prix;
   function set_price($type, $value) {
     $this->les_prix[$type] = $value;
   }
   function calcul_prix() {
      $this->prix = 0;
      foreach ($this->elements as $t => $n) {
            $this->prix += $n*$this->les_prix[$t]; }
   }
}
```


Les objets (3)

- Particularités de PHP
  - tous les attributs et méthodes sont publics
  - les objets sont libérés automatiquement quand ils ne sont plus utilisés
  - il n'y a pas de destructeur

Fonctionnalités avancées

Authentification,
Gestion des connexions HTTP,
Sessions, Cookies, Transmission d'une variable,
PHP/MySQL,
Création dynamique d'image,
Manipulation de fichiers/répertoires

Authentification HTTP

- Uniquement si PHP est exécuté comme module Apache
- Demande d'authentification au client
  - header() permet de spécifier un en-tête HTTP lors de l'envoi de la réponse

```
if (!isset($_SERVER['PHP_AUTH_USER'])) {
  header('WWW-Authenticate: Basic realm="Bienvenue" ');
  header('HTTP/1.0 401 Unauthorized');
  echo "Vous avez cliqué sur annuler"; exit;
} else {
  echo "Utilisateur: {$_SERVER['PHP_AUTH_USER']}";
  echo "Passwd: {$_SERVER['PHP_AUTH_PW']}";
}
```


Gestion des connexions HTTP (1)

- Le statut des connexions est conservé en interne par PHP
- Un script PHP a trois états possibles
  - 0/NORMAL : le script est en cours d'exécution
  - 1/ABORTED : le client s'est déconnecté
  - 2/TIMEOUT : la durée maximale d'exécution du script est dépassée
- Par défaut, le script PHP se termine dès que le client se déconnecte mais il se peut que le script ait encore du travail à effectuer

Gestion des connexions HTTP (2)

- Plusieurs possibilités pour que le script continue son exécution
  - directive ignore_user_abort dans php.ini
  - fonction PHP ignore_user_abort()
  - il est également possible de pré-enregistrer une fonction de fermeture avec register_shutdown_function(func)
    - la fonction de fermeture est appelée
      - à la fin du script s'il se termine normalement
      - à la prochaine exécution du script s'il se termine anormalement
  - connection_aborted() retourne TRUE si le script est dans l'état ABORTED
  - connection_status() retourne l'état du script (0, 1, ou 2)

Gestion des connexions HTTP (3)

- Gestion du timeout d'exécution du script
  - par défaut, au bout de 30 secondes d'exécution (directive max_execution_time), le script passe à l'état TIMEOUT
  - set_time_limit(sec) permet de modifier la valeur du timeout
  - connection_timeout() retourne TRUE si le timeout est arrivé à échéance
  - la fonction de terminaison pré-enregistrée avec register_shutdown_function(func) est également appelée quand le timeout expire

Gestion des sessions (1)

- PHP permet de sauver des informations entre deux requêtes HTTP (valeur de variables)
  - avec PHP3, il faut une bibliothèque complémentaire
  - inclus dans PHP4
- Principe avec PHP4
  - chaque nouvel utilisateur se voit attribuer un identificateur de session (SID)
  - par défaut, ce numéro est transmis à l'aide d'un cookie ; sinon, si le client n'accepte pas les cookies, PHP propage automatiquement le SID via l'URL
  - quand un utilisateur accède à la page, PHP4 vérifie si un identificateur de session est présent dans la requête HTTP

Gestion des sessions (2)

- La reconnaissance d'une session se fait dans les cas suivants
  - session.auto_start = on dans php.ini
  - lors de l'appel de session_start() ou de session_register()
  - dans le cas où le numéro de session est valide, tout l'environnement de celle-ci est restauré dans $_SESSION
- session_destroy() permet de détruire une session, session_unregister() de dé-enregistrer une variable
- Par défaut, les variables de session sont sauvegardées dans un fichier

Gestion des sessions (3)

- Exemple d'utilisation d'une session

```
<?php # dans fichier page1.php
  session_start();
  session_register('user'); # $user sera sauvegardé
  $user = "toto"; ?>
<?php # dans fichier page2.php
  session_start();
  echo "Vous êtes {$_SESSION['user']}";
  # affiche "Vous êtes toto"
?>
```

- Session et sécurité : attention, il est facile d'obtenir le SID donc ne pas l'utiliser (directement) pour l'authentification en particulier

Gestion des cookies (1)

- setcookie() permet de mettre en place un cookie (envoi vers le client)
- setcookie(), tout comme header(), doivent être exécutées avant que le contenu de la réponse n'ait commencé (pas d'affichage PHP et pas de code HTML avant l'appel)
- syntaxe : setcookie("name", "value", expire, "path", "domain", secure)
  - expire est un timestamp UNIX (temps en secondes depuis le 1er janvier 1970)
  - secure vaut 0 ou 1
- print_r($_COOKIE) permet d'afficher tous les cookies en provenance du client

Gestion des cookies (2)

- Mise en place d'un cookie

```
<?php
# le cookie expire dans 1h
setcookie("nb_pages", "1", mktime()+3600, "/~ogluck",
   ".univ-lyon1.fr"); ?>
<HTML><BODY>…</BODY></HTML>
```

- Effacement d'un cookie

```
setcookie("nb_pages", "", mktime()-3600);
```

- Lecture/modification d'un cookie

```
$nb = (string) ((int)${nb_pages} + 1);
setcookie("nb_pages", $nb, mktime()+3600,
   "/~ogluck", ".univ-lyon1.fr");
```


Transmission d'une variable

- Transmission via l'URL

```
<?php
 $var1 = "transmission vers index.php";
 echo '<A HREF="index.php?var1=', urlencode($var1),
  ' ">Cliquez ici</A>';
 /* l'url associée au lien est
  "index.php?var1=transmission+vers+index.php" */
?>
```

- Autres méthodes
  - utiliser une session PHP
  - utiliser le transfert via un cookie
  - utiliser les champs cachés d'un formulaire

PHP et MySQL (1)

- Un des points forts de PHP est de permettre des accès à de nombreux SGBD (Oracle, MySQL, …)
  - la plupart de ces SGBD sont accessibles via le langage SQL (*Structured Query Langage*) qui permet de réaliser des requêtes à la base de données de façon structurée et standardisée
  - ce qu'on retrouve dans l'API d'une base SQL
    - connexion au serveur de BD (machine, login, passwd)
    - sélection de la base à utiliser (nom de la base)
    - requête SQL vers la base
    - exploitation du résultat de la requête (affichage, …)
    - déconnexion

PHP et MySQL (2)

- Exemple de commandes SQL : une table qui contient 3 champs (Id, Nom, Prenom)

```
CREATE DATABASE MaBase
CREATE TABLE Personne (Id INT(2) NOT NULL, Nom
   VARCHAR(30), Prenom VARCHAR(30), PRIMARY KEY(Id))
INSERT INTO Personne VALUES ('1', 'Dupont', 'Paul')
INSERT INTO Personne VALUES ('2', 'Durand', 'Patrick')
SELECT Nom,Prenom FROM Personne
DELETE FROM Personne WHERE Nom = 'Dupont'
```

- PHP n'est qu'un intermédiaire entre le client et la base de données
- On se focalise désormais sur l'interface PHP/MySQL

PHP et MySQL (3)

- Configuration de PHP/MySQL dans php.ini
- Les fonctions PHP les plus couramment utilisées pour accéder à une base MySQL
  - mysql_connect() pour ouvrir une connexion avec le serveur de BD
  - mysql_select_db() pour changer la base active sur la connexion en cours
  - mysql_list_*() pour lister les bases (dbs), les tables (tables), les processus (processes), les champs d'une table (fields) disponibles sur le serveur de BD
  - mysql_query() pour envoyer une requête SQL ; un identifiant est retourné pour exploiter le résultat avec d'autres fonctions
  - mysql_close() pour fermer la connexion avec le serveur

PHP et MySQL (4)

- Exploitation du résultat d'une requête SQL
  - mysql_result() pour extraire le contenu d'un champ du résultat
  - mysql_fetch_*() pour transformer une ligne du résultat en tableau associatif (dont les clés sont les noms des champs ou l'indice de la colonne), en objet (dont les propriétés sont les noms des champs), … afin d'exploiter/manipuler le résultat
  - mysql_field_*() pour obtenir des informations sur les champs présents dans le résultat (type, nom, …)
  - mysql_free_result() permet d'effacer le résultat de la mémoire
  - mysql_num_fields() et mysql_num_rows() retournent respectivement le nombre de champs/lignes du résultat

PHP et MySQL (5)

- Connexions persistantes au serveur avec mysql_pconnect()
  - la connexion reste ouverte entre l'exécution de 2 scripts/requêtes différents
    - la connexion n'est plus fermée automatiquement à la fin du script PHP
    - accélère les accès à la base de données quand ces derniers sont fréquents
  - la connexion est fermée automatiquement après un certain temps d'inutilisation

PHP et MySQL (6)

- Traitements des erreurs - 3 possibilités
  - si la fonction échoue, le programme continue

```
$conn = mysql_connect($host, $user, $passwd)
```

  - si la fonction échoue, un message d'erreur est affiché et le programme se termine

```
$conn = mysql_connect($host, $user, $passwd)
    or die("Connexion impossible");
```

    - die("msg") est un alias de la fonction exit
  - si la fonction échoue, le message d'erreur MySQL est affiché et le programme se termine

```
if (! $conn = mysql_connect($host, $user, $passwd)) {
  $msg = mysql_error(); echo "$msg<br>\n";
  exit; }
```

    - mysql_error() permet de récupérer le message d'erreur diagnostiquant l'échec du dernier accès au serveur

Exemple PHP/MySQL (1)

```
<?php
/* ajout_personne.php : ajoute la personne ($nom,$prenom) dans la table
   Personne de la base MaBase (déjà créées) et affiche la table complète */
   /* Connexion et sélection de la base */
   $host = "localhost"; $user = "toto"; $passwd = "titi";
   $base = "MaBase"; $table = "Personne";
   $conn = mysql_connect($host, $user, $passwd) or die("Connexion");
   mysql_select_db("$base",$conn) or die("Sélection de MaBase");

   /* Recherche de l'Id suivant dans la table (en supposant que Id =
   numéro de la ligne) */
   $query = "SELECT * FROM $table";
   $result = mysql_query($query) or die("Echec de la requête1");
   $next_id = mysql_num_rows($result) + 1; mysql_free_result($result);

   /* Ajout de ($next,$nom,$prenom) dans la table */
   $query = "INSERT INTO $table VALUES ('$next_id','$nom','$prenom')";
   mysql_query($query) or die("Echec de la requête2");
```


Exemple PHP/MySQL (2)

```
   /* Affichage complet de la table */
   $query = "SELECT Nom,Prenom FROM $table";
   $result = mysql_query($query) or die("Echec de la requête3");
   echo "<table>\n";
   while ($ligne = mysql_fetch_array($result, MYSQL_ASSOC)) {
      echo "\t<tr>\n";
      foreach ($ligne as $valeur_champ) {
         echo "\t\t<td>$valeur_champ</td>\n";
      }
      echo "\t</tr>\n";
   }
   echo "</table>\n";

   /* Libération du résultat */
   mysql_free_result($result);
   /* Fermeture de la connexion */
   mysql_close($conn);
?>
```

Création dynamique d'image (1)

- PHP peut créer et manipuler des images de différents formats (gif, png, jpg, jpeg, …), en particulier s'il est associé à la librairie GD (http://www.boutell.com/gd) qui est intégrée depuis PHP4.3
- Exemple
  - une image est incluse dans une page HTML par
    <img src="button.php?text=mon+texte">
  - le script button.php récupère le contenu de la variable text, l'inscrit sur le fond d'une image et affiche l'image modifiée dynamiquement

Création dynamique d'image (2)

- Exemple (suite) : le script button.php

```
<?php
   header("Content-type: image/png");
   $string = $_GET['text'];
   $im     = imagecreatefrompng("images/button.png");
   # association d'une couleur RVB à l'image
   $orange = imagecolorallocate($im, 220, 210, 60);
   # imagesx retourne la largeur de l'image
   $px     = (imagesx($im) - 7.5 * strlen($string)) / 2;
   # écriture de la chaîne dans l'image (taille police = 3)
   imagestring($im, 3, $px, 9, $string, $orange);
    # envoi de l'image sur la sortie standard
   imagepng($im);
   imagedestroy($im);
?>
```


Manipulation de fichiers/répertoires (1)

- Manipulation de fichiers (proche du C !)
  - fopen() : ouverture d'un fichier (possibilité de fournir une URL HTTP ou FTP pour les fichiers distants)
  - fclose() : fermeture d'un fichier
  - fread()/fgets()/fwrite()/fputs() : lecture/écriture dans un fichier
  - fgetss() permet de lire une ligne dans un fichier en supprimant les balises HTML
  - rewind()/fseek()/ftell() : positionnement dans le fichier
  - fpassthru() : lit le fichier jusqu'à sa fin et l'affiche sur la sortie standard
  - file_exists() : teste l'existence d'un fichier
  - copy()/rename()/unlink() : copie/renomme/efface un fichier

Manipulation de fichiers/répertoires (2)

- Exemple : affichage d'une image

```
<?php
   header("Content-type: image/gif");
   if (! $fd = fopen("images/logo.gif", "rb"))
         echo "Ouverture impossible\n";
   else fpassthru($fd);
   fclose($fd); ?>
```

- Exemple : copie d'un fichier

```
<?php
   $fichier = "logo.gif";
   if (file_exists($fichier))
         copy($fichier, "/tmp" . $fichier);
   else echo "Fichier $fichier inexistant\n";
?>
```


Manipulation de fichiers/répertoires (3)

- Manipulation de répertoires
  - chdir() : changement de répertoire courant
  - opendir()/closedir()/mkdir()/rmdir() : ouverture, fermeture, création, suppression d'un répertoire
  - readdir() : lit l'entrée suivante dans le répertoire
  - rewinddir() : repositionnement au début du répertoire
  - dir() : instanciation d'un objet répertoire pour une manipulation objet de celui-ci
- Exemple : équivalent de la commande ls

```
<?php
   chdir("/etc");
   $rep = dir("."); $rep->rewinddir();
   while ($f = $rep->readdir()) echo "$f<br>\n";
   $rep->closedir() ?>
```


Téléchargement de fichiers (1)

- PHP offre la possibilité de recevoir des fichiers texte ou binaire en provenance du client et d'y associer un traitement
  - réception par la méthode POST : une boite de dialogue permet à l'utilisateur de sélectionner un fichier local

```
<FORM ENCTYPE="multipart/form-data" ACTION="..." METHOD="POST">
   <INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" VALUE="1000">
   Envoyez ce fichier : <INPUT NAME="userfile" TYPE="file">
         <INPUT TYPE="submit" VALUE="Envoyer le fichier"> </FORM>
```

  - le fichier téléchargé sera stocké temporairement dans un répertoire $TMPDIR sur le serveur
  - move_uploaded_file() permet de déplacer un fichier téléchargé par PHP
  - is_uploaded_file() permet de vérifier qu'un fichier a bien été téléchargé par la méthode POST

Téléchargement de fichiers (2)

- informations reçues par PHP dans $_FILES
  - le champ caché MAX_FILE_SIZE permet à PHP de faire des vérifications sur la taille du fichier téléchargé
  - $_FILES['userfile']['name'] : nom original du fichier sur la machine du client web
  - $_FILES['userfile']['type'] : type MIME du fichier, si le navigateur a fourni cette information
  - $_FILES['userfile']['size'] : taille, en octets, du fichier
  - $_FILES['userfile']['tmp_name'] : nom temporaire du fichier qui sera chargé sur la machine serveur
  - $_FILES['userfile']['error'] : 0 en cas de réussite du téléchargement sinon code d'erreur (taille trop grande…)

Conclusion

Petite conclusion ! (1)

- Ce qu'on peut faire avec PHP
  - des communications avec une base de données
  - des pages HTML dynamiques côté serveur
  - du traitement de formulaires
  - du traitement évolué de chaînes de caractères avec des profils d'expressions régulières
  - des sessions PHP
  - du traitement de tableaux
  - de la programmation objet
  - de l'accès à des fichiers distants ou locaux
  - de l'envoi de courrier électronique avec mail()
  - de la création dynamique d'images
  - du téléchargement de fichier
  - de l'authentification
  - … en bref des applications Web complètes !

Petite conclusion ! (2)

- Exemples typiques
  - Recherche de mots sur un site
  - Compteurs/statistiques des pages visitées sur le site en interaction avec une base de données
  - Traitement de formulaires
- Ce qui mériterait un approfondissement
  - PHP et la sécurité (safe_mode…)
  - PHP et XML (générer/analyser du code XML, requêtes XML-RPC ou SOAP, …)
  - PHP et LDAP, PHP et IMAP, PHP et FTP, PHP et DNS
  - PHP et ses extensions
  - PHP en ligne de commande
- -> Allez voir sur http://fr.php.net/manual