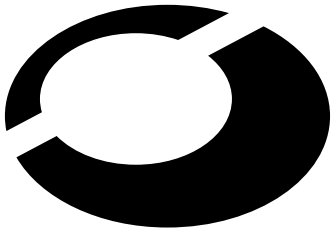
www.alcove.fr

Programmer en PHP

Julien Gaulmin
julien.gaulmin@alcove.fr

version 1.44



l'informatique est libre

Alcôve

Alcôve
Centre Paris Pleyel
153 bd Anatole France
93200 Saint-Denis, France

Tél. : +33 1 49 22 68 00
Fax : +33 1 49 22 68 01
E-mail : alcove@alcove.fr, Toile : www.alcove.fr

Table des matières

Introduction au langage

Objectifs du chapitre

Principes de la programmation web ;

Présentation de PHP ;

Installation et configuration.

La programmation web

Objectifs de la section

Rappels d'HTML ;

Le web interactif ;

Notions de CGI ;

Distinction PHP/CGI.

Rappels d'HTML

H yper **T** ext **M** arkup **L** anguage ;

Définit la structure *logique* d'un document web ;

Composé d'un ensemble de commandes de formatage ;

Basé sur la notion d' *environnement* possédant un début et une fin

-> délimiteurs : *tags* ou marqueurs.

Les marqueurs sont définis entre <> ;

Ex : `<marqueur>...</marqueur>`

La plupart des environnements peuvent être imbriqués selon des règles bien définies ;

Ex : `<H1><B>Mon titre</B></H1>`

Il n'est pas permis de faire chevaucher des environnements.

Ex : `<H1><B>Mon titre</H1></B>`

Structure générale d'une page HTML :

```
<HTML>
<HEAD>
<TITLE>Mon titre</TITLE>
<!-- Mes commentaires -->
</HEAD>
<BODY>
<H1>Mon chapitre</H1>
<P>Mon paragraphe</P>
<ADDRESS>Mes coordonnées</ADDRESS>
</BODY>
</HTML>
```


La programmation web

Alcôve - Programmer en PHP

Le web interactif

Deux principaux types d'interactions :

– côté serveur (*Server-side scripting*),
– côté client (*Client-side scripting*).

Chacun des types précédents se divise encore en sous-catégories.

Côté client :

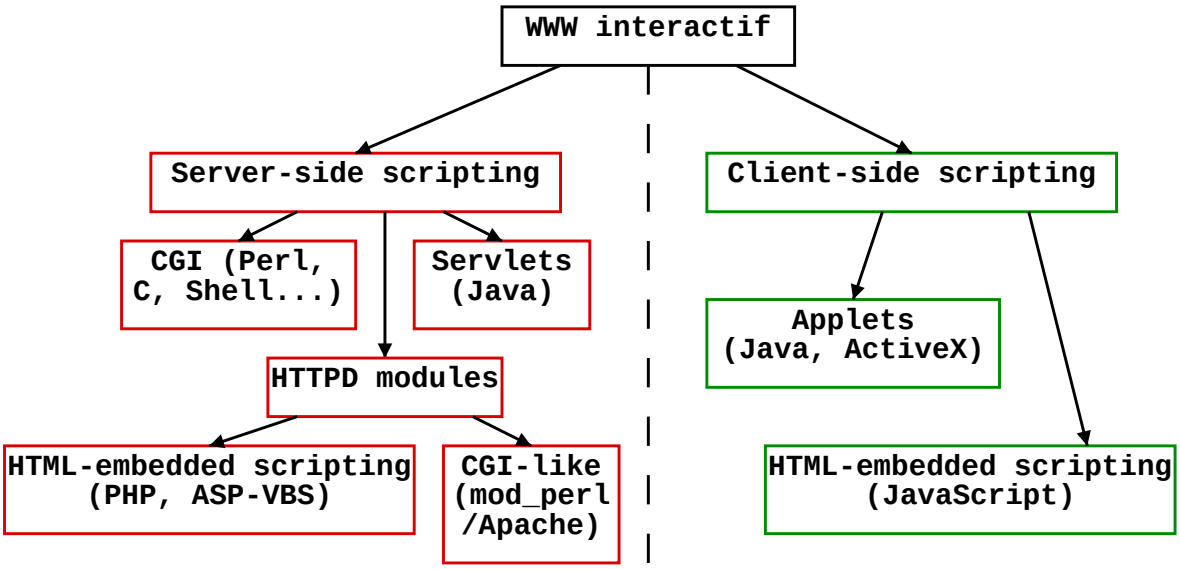
– scripts embarqués dans la page HTML (*HTML-embedded scripting*) -> Javascript...,
– applets -> Java, ActiveX...,
– *plugins* propriétaires.

La programmation web

Alcôve - Programmer en PHP

Côté serveur :

– CGI -> Perl, C, Shell...,
– Interpréteurs intégrés au serveur HTTP
  • scripts embarqués dans la page HTML (*HTTPD modules* , *HTML-embedded scripting*) -> **PHP** , ASP-VBScript...,
  • scripts à la CGI (*CGI-like*) -> mod_perl/Apache
– Servlets -> Java.



Notions de CGI

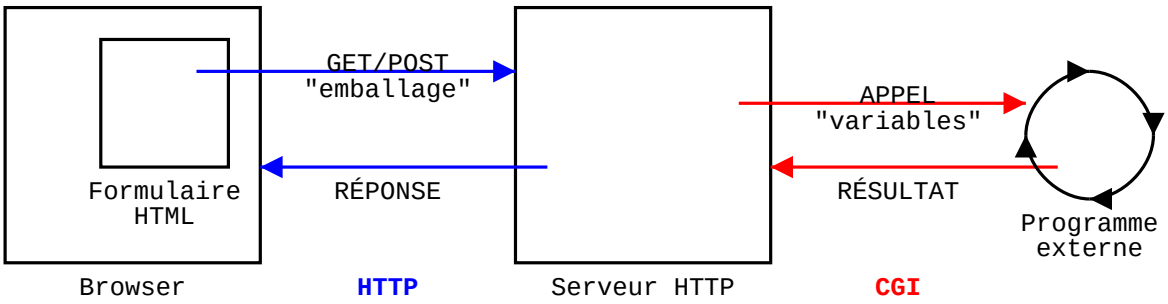
C ommon **G** ateway **I** nterface ;

Interface de communication pour la création de programmes capables d'être exécutés par un serveur HTTP ;

Permet la création dynamique de pages du côté serveur ;

Généralement utilisé pour servir les requêtes en provenance d'une page de formulaire HTML.

Principe de la CGI :



if ($want_page==1){ $want_page=0 ; }

Scénario de génération dynamique de page via la CGI :

- l'utilisateur remplit un formulaire HTML et le valide,
- le navigateur ouvre une connexion vers le serveur HTTP et lui envoie la requête correspondante (nom du CGI + paramètres),
- le serveur HTTP invoque le script CGI en lui transmettant les paramètres de la requête,
- le script CGI génère une page correspondant aux paramètres,
- le serveur envoie la page au navigateur et ferme la connexion,
- le navigateur affiche la page.

Distinction PHP/CGI

Tous les deux sont situés côté serveur mais :

PHP : le code PHP est contenu dans les pages HTML

Ex :
```
<?php $username = "toto";?> <HTML><BODY><H1>
Bonjour monsieur <?php print $username?>
</H1></BODY></HTML>
```

CGI : le code HTML est contenu dans le code du CGI

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


Présentation de PHP

Objectifs de la section ;

Vue d'ensemble du PHP ;

L'histoire du langage ;

Ses avantages ;

Et ses inconvénients.

Vue d'ensemble du PHP

A server-side, HTML-embedded scripting language

Langage de script côté serveur ;

Embarqué dans les pages HTML ;

Syntaxe héritée du C et du Perl ;

Extensible (nombreuses bibliothèques) ;

Supporte pratiquement tous les standards du web ;

Logiciel Open Source (Licence PHP de type BSD).

L’histoire du langage

Créé par Rasmus Lerdorf en 1994 pour des besoins personnels

(**P** ersonnal **H** ome **P** age) ;

Première version publique en 1995 sous le nom de *Personnal Home Page Tools* capable de gérer un livre d’or, un compteur d’accès... ;

Présentation de PHP

Alcôve - Programmer en PHP

Deuxième version plus complète (PHP/FI) en 1995 avec gestion des formulaires HTML et des bases mSQL ainsi qu’un nouvel interpréteur ;

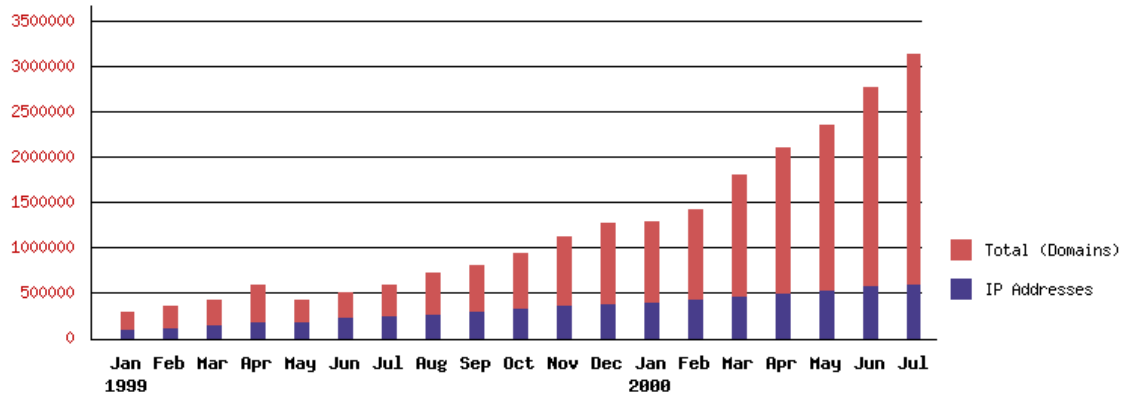
PHP/FI est utilisé par environ 50000 sites web en 1997 ;

En 1997, le projet devient un travail d'équipe et l'interpréteur est réécrit par Zeev Suraski et Andi Gutmans pour donner la version PHP3 ;

La version 3 de PHP s'est rapidement imposée pour atteindre environ 150000 sites web durant l'été 1999 (source : Netcraft) et devenir **P** HP : **H** ypertext **P** reProcessor ;

Aujourd'hui le projet est chapeauté par Zend, une société privée créée par Suraskyi et Gutmans, qui commercialise des logiciels complémentaires à PHP ;

La dernière version en date est la PHP4 (2000). Elle possède un interpréteur optimisé par Zend et s'ouvre à d'autres serveurs HTTP qu'Apache.



Les avantages de PHP

Le client n'a pas accès au code source puisque celui-ci est interprété avant envoi (!= Javascript) ;

Le client ne reçoit que le résultat de l'exécution du script ;

On peut configurer le serveur HTTP pour masquer complètement la qualité dynamique des pages ;

Le code n'est pas alourdi par des commandes destinées à générer la page HTML (!= CGI) ;

Le langage possède de nombreuses bibliothèques/modules pour :
– le calcul mathématique,
– la création dynamique d'images,
– la gestion de sessions,
– les connexions sécurisées (SSL),
– l'accès à la plupart des SGBD,
– l'accès aux bases LDAP...

Les inconvénients de PHP

Pas aussi rapide que mod_perl ou certains CGI mais plus portable ;

Pas aussi simple que ASP-VBScript mais plus respectueux des standards web et sous licence Open Source (-> plus facilement extensible) ;

Pas d'interactivité au niveau du client (-> on doit quand même utiliser du Javascript dans ces cas là).

Installation et configuration de PHP

Objectifs de la section

Installation de PHP ;

Configuration via `php3.ini/php.ini` ;

Premier test ;

Ajout de nouveaux modules.

Installation

Trois types d'installation :
– module dynamique du serveur HTTP (ex : `mod_php` pour Apache),
– module statique du serveur HTTP (20% plus rapide que `mod_php`) qui nécessite de patcher le serveur HTTP et de tout recompiler,
– sous forme de CGI (pour les serveurs HTTP non supportés).

Télécharger des sources sur `http ://fr.php.net` (mirroir français du site officiel de PHP) ;

Installer les sources dans un répertoire différent de celui du serveur HTTP (ex : `/usr/local/src/php/`) avec la commande

`tar -zxvf php-<version>.tar.gz -C /usr/local/src/php/` ;

Configuration de l'installation :
– `./setup` (configuration interactive), ou
– `./configure <options>` (`./configure --help` pour connaître toutes les options).

Compilation et installation :

```
make ; su ; make install
```

Association des fichiers contenant du code PHP avec l'interpréteur PHP dans la configuration du serveur HTTP

Ex : serveur Apache (`srm.conf` ou `httpd.conf`)

PHP3 -> `AddType application/x-httpd-php3 .php`

PHP4 -> `AddType application/x-httpd-php .php`

Sous Debian, tout ceci est simplifié grâce à l'utilitaire `apt-get` (remplacer `php3` par `php4` pour PHP4) :
– `apt-get install php3` pour le module dynamique Apache, ou
– `apt-get install php3-cgi` pour le CGI.

Configuration de PHP (`php3.ini/php.ini`)

Généralement dans `/etc/php<version>/apache/` ou `/usr/lib/php<version>/` ;

Ce fichier (`php3.ini` pour PHP3 et `php.ini` pour PHP4) contrôle les principaux comportements de PHP ;

Très clair et facilement configurable ;

Syntaxe :
– clef = valeur,
– clef = "valeur complexe",
– ; commentaire,
– booléen -> 1, On, True, Yes / 0, Off, False, No.

Segmenté en parties :
– options du langage,
– ressources maximales allouées à PHP,
– gestion des erreurs,
– gestion des données HTTP,
– fichiers et répertoires,
– extensions dynamiques (modules),
– configuration des modules.

Pour que PHP prenne les modifications en compte il faut relancer le serveur HTTP (ex : serveur Apache -> `apachectl restart`).

Premier test

Créer le fichier `test.php` :

```
<?php phpinfo();?>
```

Placer ce fichier à la racine du serveur HTTP (ou dans le répertoire `doc_root` si vous avez activé l'option `safe_mode`) ;

Charger l'URL `http ://localhost/test.php` sur votre navigateur web ;

Bravo ! Vous avez créé votre première page dynamique avec PHP et en plus elle vous résume toutes les fonctionnalités PHP qui sont configurées sur votre machine.

Installer un nouveau module PHP

Placer le nouveau fichier `<nom>.so` dans `extension_dir` (par défaut `/usr/lib/php<version>/apache` mais configurable dans `php3.ini/php.ini`) ou utiliser `apt-get install php<version>_<nom>` sous Debian ;

Ajouter la ligne `extension=<nom>.so` dans `php3.ini/php.ini` ;

Configurer les aspects relatifs au module dans `php3.ini/php.ini` ;

Redémarrer le serveur HTTP.

Les bases du langage

Objectifs du chapitre

Syntaxe du langage ;

Les structures de contrôle ;

Fonctions ;

Programmation orientée objet.

Section 1

Syntaxe du langage

Objectifs de la section

Syntaxe de base ;

Les types de données ;

Les variables ;

Les constantes ;

Les opérateurs.

Syntaxe de base

Quatre moyens équivalents pour passer du mode HTML au PHP :

– standard :
  `<?php ...mon code PHP...?>`,
– option `short_open_tag` :
  `<? ...mon code PHP...?>`,
– option `asp_tags` :
  `<% ...mon code PHP... %>`,
– marqueur `script` du HTML :
  `<script language="php"> ...mon code PHP... </script>`.

Bases de la syntaxe héritées du C et du Perl :

– séparateur d'instructions -> `;`
– commentaires :
  - `...code /* ...mes commentaires... */ code...`,
  - `...code... // ...mes commentaires...`,
  - `...code... # ...mes commentaires...`

Les types de données

PHP supporte les types de données suivants :

– nombres entiers,
– nombres à virgule flottante,
– chaînes de caractères,
– tableaux,
– objets (développés dans la section 'programmation orientée objet').

Tous les noms de variables sont précédés d'un $

(ex : `$toto = 5;`);

Il est possible de spécifier une variable de type *entier* de la façon suivante :

– `$toto = 123 ;` # est un entier en base 10,
– `$toto = -123 ;` # est un entier négatif,
– `$toto = 0123 ;` # est un entier en base 8,
– `$toto = 0x123 ;` # est un entier en base 16.

Il est possible de spécifier une variable de type *flottant* ou *double* de la façon suivante :

– `$titi = 1.234 ;` # est un nombre à virgule flottante,
– `$titi = 1.2e3 ;` # est aussi un nombre à virgule flottante.

Il est possible de spécifier une variable de type *chaîne de caratères* de la façon suivante :

– `$personne = 'M. Smith' ;` # est une chaîne de caractères,
– `$personne = "M. Smith" ;` # est aussi une chaîne de caractères.

Dans le deuxième cas, si la chaîne contient des noms de variables, celles-ci seront remplacées par leur valeur ;

Ex :

```
$type = 'M.' ;
$nom = "Smith" ;
$personne = "$type $nom" ;
```

Equivalent à `$personne = 'M. Smith' ;`.

Quand on utilise les "..." on doit donc *échapper* certains caractères avec un backslash (\) pour pouvoir les afficher comme tels :
– dollar ($) : `\$`,
– double quotes (") : `\"`,
– backslash (\) : `\\`.

De même, il existe des caractères spéciaux qui nécessitent d'être échappés :
– nouvelle ligne : `\n`,
– retour à la ligne : `\r`,
– tabulation : `\t`.

Tout autre caractère échappé génèrera un avertissement (*warning*) ;

Lorsqu'une *chaîne de caractères* est évaluée comme une valeur numérique, les règles suivantes s'appliquent :
– la chaîne est de type *flottant/double* 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),
– lorsque la première expression est une chaîne, le type de la variable dépend de la seconde expression.

Ex :

```
$toto = 1 + "4.5"; # $toto vaut 5.5
$toto = 1 + "-1e3"; # $toto vaut -999
$toto = 1 + "titi + 149"; # $toto vaut 1
$toto = 1 + "149 + titi"; # $toto vaut 150
```

Les tableaux de PHP ressemblent aux tableaux associatifs

(*hash-tables*) du Perl ;

L'index dans le tableau est appellé *clé* et peut être indifféremment un *entier* ou *une chaîne de caractères* ;

La valeur associée à une clé est appelée *valeur* ;

On peut utiliser la fonction `array()` pour créer un tableau ;

On peut aussi affecter directement les valeurs au tableau ;

La fonction `list()` permet d'affecter des variables comme si elles constituaient un tableau ;

Ex :

```
$tab[0] = 1;
$tab[1] = "toto"; # on peut mélanger les contenus
$tab["toto"] = "titi"; # on peut mélanger les clés
$tab["toto"][8]; # tableau à deux dimensions
$tab["toto"][8]["encore"][2]; # quatre dimensions
$suite = array(1, 2, 3, 4);
$personne = array("type" => "M.", "nom" =>
"Smith");
list($num1, $num2) = $suite; # $num1 vaut 1 et $num2
vaut 2
```

On peut connaître le nombre d'éléments d'un tableau grâce aux fonctions :

– `sizeof()` : retourne le nombre d'éléments d'un tableau, ou
– `count()` : retourne le nombre d'éléments d'un tableau s'il existe, 1 si la variable n'est pas un tableau et 0 si la variable n'existe pas.

Chaque tableau entretient un pointeur courant qui sert à naviguer en son sein grâce aux fonctions :

– `reset()` : place le pointeur interne sur le **premier élément** et retourne sa valeur,
– `current()` : retourne la valeur de l' **élément courant** ,
– `next()` : place le pointeur interne sur l' **élément suivant** et retourne sa valeur,
– `prev()` : place le pointeur interne sur l' **élément précédent** et retourne sa valeur,
– `each()` : retourne la **paire clé/valeur courante** du tableau et avance le pointeur sur l'élément suivant (c'est la seule fonction qui ne retourne pas faux si l'élément vaut 0 ou "").

Un tableau peut être trié en utilisant les fonctions suivantes :

– `asort()`/`arsort()` : trient le tableau en ordre croissant/décroissant de **valeurs** ,
– `ksort()`/`rsort()` : trient le tableau en ordre croissant/décroissant de **clés** ,
– `sort()` : trie le tableau en ordre croissant **clés et valeurs** (on perd la correspondance clé/valeur),
– `uasort()`/`uksort()`/`usort()` : trient le tableau de la même façon que leurs quasi-homonymes (u pour *user*) mais avec une fonction de comparaison fournie par l'utilisateur.

Ex :

```
function cmp($a, $b) {
if ($a == $b) return 0;
return ($a > $b)? -1 : 1;
}
$tab = array(3,2,5,6,1);
usort($tab, cmp);
while(list($cle, $valeur) = each($tab)) {
echo "tab[$cle] = $valeur<BR>\n";
}
```

Ce programme est équivalent à `rsort()`.

Les variables

La portée d'une variable dépend du contexte dans lequel elle est définie ;

La plupart des variables ont une portée qui s'étend sur l'intégralité du script PHP, elles sont *globales* ;

Cependant, les variables sont *locales* au sein d'une fonction ;

Ex :

```
$toto = 1; # portée globale
function affiche () {
echo $toto; # portée locale
}
affiche();
```

Cet exemple n'affichera rien car `echo()` utilise la variable locale `$toto`.

Deux façons existent pour accéder à une variable globale au sein d'un bloc :

– déclarer la variable comme `global` au sein du bloc,
– utiliser le tableau associatif `$GLOBALS` avec comme clé le nom de la variable globale.

Ex :

```
$toto = 1 ; # portée globale
$titi = 2 ; # portée globale
function affiche () {
global $toto ; # portée globale
echo "$toto et $GLOBALS['titi']" ;}
affiche() ;
```

Cet exemple affichera `1 et 2`.

Variables statiques :

Une variable statique est une variable locale qui ne perd pas sa valeur à chaque fois que le bloc est exécuté ;

On utilise, comme en C, l'attribut `static` pour déclarer une telle variable :

Ex : `static $toto;`

Ce type de variables est très utile pour la création de fonctions récursives ;

Ex : compter récursivement jusqu'à 10

```
function compte () {
static $compteur = 0;
$compteur++;
echo "$compteur ";
if ($compteur < 10) compte();
}
compte();
```

Affiche `1 2 3 4 5 6 7 8 9 10`.

Variables dynamiques :

Une variable dynamique prend la valeur d'une variable et l'utilise comme nom d'une autre variable ;

Ex :

```
$toto = "Hello"; # $toto vaut Hello
$$toto = "World"; # $Hello vaut World
echo "$toto $Hello!"; # affiche Hello World!
echo "$toto ${$toto}!"; # affiche aussi Hello World!
```


Le nom d'une variable peut être placé entre accolades pour éviter les ambiguïtés ;

Ex :

`$a[1] = "V1";`	`$a = "V2";`
`$b = "a";`	`$b[1] = "a";`
`Résultat : ${$b}[1] == "V1"`	`Résultat : ${$b[1]} == "V2"`

Conclusion : `${$b}[1]` et `${$b[1]}` s'utilisent dans 2 contextes différents.

Variables extérieures :

Toutes les variables transmises par les méthodes `POST` ou `GET` sont accessibles via les tableaux associatifs `$HTTP_POST_VARS` et `$HTTP_GET_VARS` (si l'option `track_vars` est activée dans `php3.ini/php.ini`) ;

Les variables d'environnement sont accessibles comme des variables PHP classiques (ex : `$HOME`) ;

On peut aussi utiliser les fonctions `getenv()` et `putenv()` pour manipuler les variables d'environnement ;

Variables prédéfinies :

Certaines variables sont prédéfinies par PHP, elles sont disponibles pendant toute l'exécution du script.

PHP_SELF	URI du script en cours d'exécution.
GLOBALS	Tableau des variables globales.
HTTP_GET_VARS	Tableau des variables issues de `GET`.
HTTP_POST_VARS	Tableau des variables issues de `POST`.
HTTP_COOKIE_VARS	Tableau des variables issues des cookies.

Variables d'environnement CGI :

SERVER_SOFTWARE	SERVER_NAME
GATEWAY_INTERFACE	SERVER_PROTOCOL
SERVER_PORT	REQUEST_METHOD
PATH_INFO	PATH_TRANSLATED
SCRIPT_NAME	QUERY_STRING
REMOTE_HOST	REMOTE_ADDR
AUTH_TYPE	REMOTE_USER
REMOTE_IDENT	CONTENT_TYPE
CONTENT_LENGTH	

`http ://hoohoo.ncsa.uiuc.edu/cgi/env.html`

Pour déterminer le type d'une variable on peut utiliser les fonctions suivantes :

- `gettype()` : retourne une chaine décrivant le type de la variable passée en argument (*integer* , *double* , *string* , *array* , *object* , *unknown type*),
- `is_long()`/`is_int()`/`is_integer()` : retourne *true* si la variable passée en argument est un entier et *false* dans le cas contraire,
- `is_double()`/`is_float` : idem mais avec les flottants/doubles,
- `is_string()` : idem mais avec les chaînes de caractères,
- `is_array()` : idem mais avec les tableaux,
- `is_object()` : idem mais avec les objets.

Les constantes

PHP définit certaines constantes et propose un mécanisme pour en définir d'autres durant l'exécution du script ;

On définit une constante en utilisant la fonction `define()` ;

Ex :

```
define("MA_CONSTANTE", "Bonjour");
echo MA_CONSTANTE; # affiche Bonjour
```


Syntaxe du langage

Alcôve - Programmer en PHP

Les principales constantes définies par PHP sont les suivantes :

- `__FILE__` : nom du fichier actuellement exécuté,
- `__LINE__` : numéro de la ligne qui est actuellement exécutée,
- `PHP_VERSION` : présentation de la version du PHP utilisée (ex : `3.0.8-dev`),
- `PHP_OS` : système d'exploitation utilisé par la machine qui fait tourner le PHP (ex : `Linux`),
- `TRUE` : vrai,
- `FALSE` : faux,
- `E_*` : gestion des erreurs (cf. section correspondante).

Les opérateurs

Les opérateurs arithmétiques :
- addition : `$a + $b`,
- soustraction : `$a - $b`,
- multiplication : `$a * $b`,
- division : `$a / $b`,
- modulo (reste de la division entière) : `$a % $b`.

L'opérateur sur les chaînes de caractères :
- concaténation : `chaine1 . chaine2`

Ex :

```
$jour = "Lundi";
echo 'Nous sommes ' . $jour . "!";
```

Affiche `Nous sommes Lundi!`

Les opérateurs sur les bits :

– ET bit à bit : `$a & $b`,
– OU bit à bit : `$a | $b`,
– OU-Exclusif/XOR bit à bit : `$a ^ $b`,
– NON bit à bit : `~$a`,
– décalage à droite de $b bits : `$a > > $b`,
– décalage à gauche de $b bits : `$a < < $b`.

Les opérateurs logiques :

– ET (vrai si $a et $b vrais) :
  - `$a and $b`,
  - `$a && $b`.
– OU (vrai si $a ou $b vrai(s)) :
  - `$a or $b`,
  - `$a || $b`.
– OU-Exclusif/XOR (vrai si seul $a ou $b vrai) : `$a xor $b`,
– NON (vrai si $a est faux) : `!$a`.

Les opérateurs d'affectation :

– l'opérateur d'affectation le plus simple est le signe =,
– il ne signifie pas "égal à" mais que l'opérande à gauche du signe = se voit affecté de la valeur de l'opérande de droite,
– la valeur retournée par une expression d'assignement est la valeur assignée,

Ex : `$a = ($b = 4) + 3;` # $a vaut 7 et $b vaut 4

– il existe en plus des *opérateurs combinés* pour tous les opérateurs arithmétiques, les opérateurs bits à bits et l'opérateur de concaténation,

+=	-=	*=	/=	%=	.=
&=	|=	^=	< <=	> >=	~=

– ceux-ci permettent d'utiliser la valeur d'une variable dans une expression et d'affecter le résultat de cette expression à cette variable,

Ex : `$toto += 3` # équivaut à `$toto = $toto + 3`

– l'opérateur ++ est équivalent à += 1,
– l'opérateur -- est équivalent à -= 1,
– ces deux opérateurs peuvent être placés avant (pré-exécution) ou après (post-exécution) la variable à laquelle ils s'appliquent.

Ex :

```
$toto = 0;
echo ++$toto; # affiche 1
echo $toto++; # affiche 1
echo $toto; # affiche 2
```


Les opérateurs de comparaison :
– égal à : $a == $b,
– différent de : $a != $b,
– supérieur à : $a > $b,
– inférieur à : $a < $b,
– supérieur ou égal à : $a >= $b,
– inférieur ou égal à : $a <= $b.

L'opérateur ternaire :

– issu du langage C,
– `(condition) ? (expression1) : (expression2);`,
– renvoie `expression1` si `condition` est vraie et `expression2` dans le cas contraire.

Ex :

```
echo $toto == 0? "Nul" : "Non nul";
```

affiche `Nul` si `$toto` vaut 0 et `Non nul` sinon

Les structures de contrôle

Objectifs de la section

Introduction ;

Instructions conditionnelles ;

Boucles ;

Inclusions.

Introduction

Tous les scripts PHP sont une suite d’instructions ;

Une instruction peut être :
– un assignement,
– un appel de fonction,
– une instruction conditionnelle, ou
– une instruction qui ne fait rien (une instruction vide).

Une instruction se termine habituellement par un point virgule (`;`) ;

Plusieurs instructions peuvent être regroupées en *bloc* , délimité par des accolades (`{...}`) ;

Un bloc est considéré comme une instruction ;

Les différents types d'instructions sont décrits dans ce chapitre.

Instructions conditionnelles

L'instruction `if` est une des plus importantes instructions de tous les langages, PHP inclus ;

Elle permet l'exécution conditionnelle d'une partie de code ;

Les fonctionnalités de l'instruction `if` sont les mêmes en PHP qu'en C ;

Prototype :

```
if (condition) {
```

instructions à exécuter si la condition est vraie...

```
}
```

Une chaine de caractère ne contenant que `0` sera évaluée comme fausse ;

Souvent on souhaite exécuter une instruction si une condition est remplie, et une autre instruction si cette condition n’est pas remplie ;

C’est le rôle du `else` ;

Prototype :

```
if (condition) {
```

instructions à exécuter si la condition est vraie...

```
} else {
```

instructions à exécuter si la condition est fausse...

```
}
```

`elseif` permet, comme l'expression `else`, d'exécuter une instruction après un `if` dans le cas où ce dernier est évalué comme faux ;

A la différence de l'expression `else`, il n'exécutera les instructions que si l'expression condionnelle `elseif` est évaluée comme vraie ;

Prototype :

```
if (condition1) {
# instructions à exécuter si la condition1 est vraie...
} elseif (condition2) {
# instructions à exécuter si la condition2 est vraie...
} elseif (condition3) {
# instructions à exécuter si la condition3 est vraie...
...
} else {
# instructions à exécuter si aucune des conditions n'est vraie...
}
```

La syntaxe du `if` que nous venons de voir est directement héritée du langage C ;

Pas toujours très adaptée pour mélanger code PHP et HTML ;

PHP propose donc une autre syntaxe pour le `if` qui permet d'améliorer la lisibilité ;

Prototype simplifié :

```
if (condition1) :
# instructions à exécuter si la condition1 est vraie...
elseif (condition2) :
# instructions à exécuter si la condition2 est vraie...
elseif (condition3) :
# instructions à exécuter si la condition3 est vraie...
...
else :
# instructions à exécuter si aucune des conditions n'est vraie...
endif;
```

Ex :

```
if ($type == "Femme") {
echo "Bonjour Madame";
} elseif ($type == "Homme") {
echo "Bonjour Monsieur";
} else {
echo "Bonjour, vous êtes bizarre!";
}
```


Boucles

La boucle `while` est le moyen le plus simple d'implémenter une boucle en PHP ;

Cette boucle se comporte de la même manière qu'en C : les instructions sont exécutées tant que l'expression est vraie ;

La valeur de l'expression est vérifiée à chaque début de boucle ;

Si la valeur change durant l'exécution de l'instruction, l'exécution ne s'arrêtera qu'à la fin de l'itération ;

Comme avec le `if`, PHP définit une syntaxe simplifiée, en plus de celle héritée du langage C ;

Prototype :

```
while (condition) {
# instructions à exécuter tant que la condition est vraie...
}
```

Prototype simplifié :

```
while (condition) :
# instructions à exécuter tant que la condition est vraie...
endwhile;
```

Ex : compter jusqu'à 10

```
$i = 1;
while ($i <= 10) {
echo "$i ";
$i++;
}
```

Affiche `1 2 3 4 5 6 7 8 9 10`.

La boucle `do..while` ressemble beaucoup à la boucle `while` ;

La différence est que l'expression est testée à la fin de chaque itération au lieu du début ;

La première itération de la boucle est donc toujours exécutée ;

Il n'y a qu'une syntaxe possible pour les boucles `do..while` qui est, elle aussi, héritée du langage C ;

Prototype :

```
do {
```

instructions à exécuter la première fois et ensuite tant que la condition est vraie...

```
} while (condition);
```


Ex : compter jusqu’à 10

```
$i = 1;
do {
echo "$i ";
$i++;
} while ($i <= 10);
```

Affiche `1 2 3 4 5 6 7 8 9 10`.

La boucle `for` est, en PHP tout comme en C, la boucle la plus complexe mais aussi la plus puissante ;

Elle fonctionne comme la boucle `for` du langage C ;

Prototype :

```
for (expression1; condition; expression2) {
# instructions à exécuter tant que la condition est vraie...
}
```


L'`expression1` est évaluée (exécutée) quoi qu'il arrive au début de la boucle ;

Au début de chaque itération, la `condition` est évaluée ;
- si elle est vraie, la boucle continue et les instructions sont à nouveau exécutées,
- si elle est fausse, l'exécution de la boucle s'arrête.

A la fin de chaque itération, l'`expression2` est évaluée (exécutée) ;

Les expressions/conditions peuvent éventuellement être laissées vides (-> attention aux boucles infinies) ;

Comme avec le `if` et le `while`, PHP définit une syntaxe simplifiée, en plus de celle héritée du langage C ;

Prototype simplifié :

```
for (expression1; condition; expression2) :
# instructions à exécuter tant que la condition est vraie...
endfor;
```


Ex : compter jusqu'à 10

```
for ($i = 1; $i <= 10; $i++) {
echo "$i ";
}
```

Ex2 : compter jusqu'à 10

```
for ($i = 1; $i <= 10; print "$i ", $i++);
```

Ex3 : compter jusqu'à 10

```
for ($i = 1; $i <= 10; print $i++);
```

Ces 3 exemples affichent `1 2 3 4 5 6 7 8 9 10`.

PHP4 définit une boucle `foreach`, comme en Perl, pour réaliser une boucle sur les éléments d'un tableau ;

En PHP3 on peut réaliser l'équivalent avec une boucle `while` et les fonction `list()` et `each()` que nous avons vu précédemment ;

Ex : afficher tous les arguments d'un formulaire `POST`

```
while (list($cle, $valeur) =
each($HTTP_POST_VARS)) {
echo "$cle => $valeur, ";
}
```


Avec l'instruction `foreach` cette boucle devient :

```
foreach ($HTTP_POST_VARS as $cle => $valeur) {
echo "$cle => $valeur, ";
}
```

Ou bien, si l'on ne veut récupérer que les valeurs :

```
foreach ($HTTP_POST_VARS as $valeur) {
echo "$valeur, ";
}
```

L'instruction `break` permet de sortir d'une boucle à n'importe quel moment ;

Ex : compter jusqu'à 10

```
for ($i = 1, , $i++) {
if ($i > 10) break;
echo "$i ";
}
```

Affiche `1 2 3 4 5 6 7 8 9 10`.

L'instruction `continue` permet d'ignorer les instructions restantes dans la boucle et de passer directement à l'itération suivante ;

Ex : compter 2 par 2 jusqu'à 10

```
for ($i = 1, $i <= 10, $i++) {
if ($i % 2) continue; # $i impaire
echo "$i ";
}
```

Affiche `2 4 6 8 10`.

L'instruction `switch` équivaut à une serie d'instructions `if..elseif..elseif....else` ;

Elle est utilisée pour comparer la même variable (ou expression) avec un grand nombre de valeurs différentes et d'exécuter différentes parties de code suivant la valeur à laquelle elle est égale ;

Prototype :

```
switch (expression) {
case resultat1 :
# instructions à exécuter si l'expression vaut resultat1...
break;
case resultat2 :
# instructions à exécuter si l'expression vaut resultat2...
break;
...
default :
# instructions à exécuter en dernier recours...
}
```

L'instruction `switch` exécute chacune des clauses dans l'ordre d'écriture ;

Elle est exécutée ligne par ligne ;

Aucun code n'est exécuté jusqu'à ce que le premier `case` soit vérifié ;

Lorsqu'il trouve le premier `case` vérifié, PHP exécute alors les instructions correspondantes et continue jusqu'à la fin du bloc `switch` ;

Pour reproduire le fonctionnement d'un `if..elseif..elseif....else`, il faut terminer tous les `cases` par des `break` ;

On peut définir un `case` exécuté par défaut (équivalent au `else`) grâce à `default` ;

Ex :

```
switch ($type) {
case "Femme" :
echo "Bonjour Madame";
break;
case "Homme" :
echo "Bonjour Monsieur";
break;
default :
echo "Bonjour, vous êtes bizarre!";
}
```


Inclusions

La fonction `require()` se substitue elle-même avec le contenu du fichier spécifié en argument ;

La substitution s'opère avant l'interprétation du code ;

Elle ressemble à peu près au `#include` du C ;

Si on l'utilise au sein d'une boucle, c'est le fichier tel qu'il était à la première itération qui sera réutilisé à chaque itération (même s'il a changé entre temps) ;

De plus, il est déconseillé d'utiliser `require()` avec un nom de fichier dynamique (contenant une variable) ;

La fonction `include()` inclut et évalue le fichier spécifié en argument ;

Ceci est fait à chaque fois que la fonction est rencontrée ;

On peut donc utiliser cette fonction au sein d'une boucle pour inclure un fichier différent à chaque fois par exemple (ex : `include("toto_$i");`).

Fonctions

Objectifs de la section

Présentation des fonctions PHP ;

Les valeurs de retour ;

Les arguments.

Présentation des fonctions PHP

Une fonction peut être définie en utilisant la syntaxe suivante :

```
function ma_fonction ($arg_1, $arg_2, ..., $arg_n)
{
# Corps de ma fonction...
return $valeur_de_retour;
}
```

Tout code PHP correct syntaxiquement peut apparaître dans une fonction et dans une définition de classe ;

En PHP3 toute fonction doit être préalablement définie avant d’être utilisée (idem langage C). Elle peut être définie n’importe où en PHP4.

Les valeurs de retour

Les valeurs sont renvoyées en utilisant une instruction de retour optionnelle ;

Tous les types de variables peuvent être renvoyés (tableaux et objets compris) ;

Ex :

```
function carre($nombre) {
return $nombre * $nombre;
}
echo carre(3); # affiche la valeur 9
```


On ne peut pas renvoyer plusieurs valeurs en même temps mais on peut obtenir un résultat identique en renvoyant un tableau ;

Ex :

```
function couleurs() {
return array("Coeur", "Pique", "Trèfle",
"Carreau");
}
list($c1, $c2, $c3, $c4) = couleurs();
```

PHP4 permet également le retour de fonction par référence ;

Ce mécanisme est utile lorsque l'on cherche à utiliser une fonction pour trouver une variable que l'on veux lier ;

A la différence du passage par référence, il faut utiliser un & à deux endroits :

– pour indiquer que l'on retourne une référence (et non une copie comme à l'habitude), mais aussi
– pour indiquer qu'une liaison à une référence et non un simple assignement de variable doit avoir lieu avec la variable de retour.

Ex :

```
function & trouve_variable ($param) {
# ...recherche de la variable $toto...
return $toto;
}
$titi =  & trouve_variable($tata);
$titi->x = 2;
```

Les propriétés de l'objet retourné par la fonction `trouve_variable()` sont liées à `$titi`. Il ne s'agit pas d'une copie de `$toto`.

Les arguments

Des informations peuvent être passées à une fonction en utilisant un tableau d'arguments dont chaque élément est séparé par une virgule ;

Un élément peut être une variable ou une constante ;

PHP supporte :
– le passage d'arguments par valeur (méthode par défaut),
– le passage d'arguments par référence, et
– les valeurs par défaut des arguments.

Une liste d'arguments dont le nombre est variable ne fonctionne pas en PHP mais un résultat similaire peut être obtenu en utilisant un tableau ;

Passage d'arguments par valeur :

C'est la méthode par défaut ;

Les variables **ne sont pas** affectées par des changements au sein de la fonction ;

On peut donc changer la valeur des arguments au sein de la fonction sans que ceci ait des répercussions à l'extérieur de celle-ci ;

Ex :

```
function ajoute_deux($nombre) {
$nombre += 2;
echo $nombre;
}
$toto = 3;
ajoute_deux($toto); # affiche 5
echo $toto; # affiche 3
```


Passage d’arguments par référence :

Les variables **sont** affectées par des changements au sein de la fonction ;

Deux possibilités :

– de façon permanente en ajoutant un & devant le nom de la variable dans la définition de la fonction, ou
– de façon ponctuelle en ajoutant un & devant le nom de la variable lors de l’appel à la fonction.

Ex :

```
function retranche_deux( & $nombre) {
$nombre -= 2 ;
echo $nombre ;
}
$toto = 3 ;
ajoute_deux( & $toto) ; # affiche 5
echo $toto ; # affiche 5
retranche_deux($toto) ; # affiche 3
echo $toto ; # affiche 3
```


Valeurs par défaut des arguments :

PHP permet, comme en C++, de spécifier des valeurs par défaut pour les arguments de type scalaire ;

Il suffit de spécifier la valeur par défaut avec un = dans la déclaration de la fonction ;

Cette valeur doit être une constante (pas de variables ni d’attributs de classes) et les arguments par défaut doivent être les derniers dans la liste ;

Ex :

```
function annotation($type = "assez bien") {
echo "Vous avez $type travaillé";
}
annotation("mal"); # affiche Vous avez mal travaillé
annotation(); # affiche Vous avez assez bien travaillé
```


Programmation orientée objet

Objectifs de la section

Rappels sur la programmation orientée objet ;

Les classes en PHP ;

Les objets en PHP ;

Particularités de PHP.

Rappels sur la programmation orientée objet

POO -> raisonner en termes de fonctionnalités et de propriétés et non en terme de traitements ;

L' **encapsulation** :

L'objet représente une *abstraction* d'une entité abstraite ou concrète pour laquelle il regroupe en une même structure les données et les traitements ;

Un objet possède des caractéristiques et des fonctionnalités et on l'utilise sans savoir comment elles sont *implémentées* (ex : un téléphone) ;

Les caractéristiques/fonctionnalités non utiles pour la manipulation de l'objet mais utiles à son fonctionnement sont masquées à l'utilisateur (ex : les composants internes du téléphone) ;

L' **héritage** :

Un objet peut *dériver* d'un autre objet et posséder des caractéristiques/fonctionnalités supplémentaires tout en *héritant* des caractéristiques/fonctionalités de son aïeul (ex : un téléphone sans fils est un téléphone un peu particulier) ;

Le **polymorphisme** :

Lorsqu'on invoque une méthode d'un objet, le programme saura retrouver la méthode parmis celles des aïeux de l'objet si celle-ci n'est pas explicitement définie pour lui mais qu'il en a hérité (ex : la méthode "décrocher le téléphone" pour un sans fils) ;

A la création (*instanciation*) de l'objet, une méthode (qui porte généralement le même nom que la classe) est appellée automatiquement : c'est le **constructeur** ;

Parfois, à la destruction de l'objet, une méthode (qui porte généralement le même nom que la classe précédé d'un ~) est appellée automatiquement : c'est le **destructeur** ;


Programmation orientée objet
Alcôve - Programmer en PHP

Une *classe* est une collection de variables (*attributs*) et de fonctions (*méthodes*) qui s'appliquent à ces variables ;

Un *objet* est une instance d'une classe. Son type de données est la classe d'objets à laquelle il appartient (un peu comme `$toto = 123 ;` est une instance du type *entier*).

Les classes en PHP

En PHP, on définit une classe d'objets de la façon suivante :

```
class MaClasse {
# Attributs
var $monAttribut;
# Méthodes
function maMethode($argument) {
# ...traitement de $monAttribut avec $argument...
}
}
```



Programmation orientée objet
Alcôve - Programmer en PHP

Les objets en PHP

Il est possible de spécifier une variable *objet* de la façon suivante :

```
$monObjet = new MaClasse;
```

On accède à un attribut ou à une méthode de la façon suivante :

```
$monObjet->monAttribut = "toto";
$monObjet->maMethode(23);
```

Particularités de PHP

Tous les attributs et méthodes sont publics ;

Les objets sont libérés automatiquement quand ils ne sont plus utilisés (*Garbage Collector*) ;

Pas de destructeur.

Ex : le Caddie

```
class Caddie {
var $elements ; # éléments de notre Caddie
# Constructeur qui ajoute un cadeau à chaque nouveau Caddie
function Caddie() {
$this->ajoute(1, "cadeau");
}
# Ajout de $nombre articles de type $type au Caddie
function ajoute($nombre, $type) {
$this->elements[$type] += $nombre ;
}
```

```
# Retrait de $nombre articles de type $type au Caddie
function enleve($nombre, $type) {
if ($this->elements[$type] > $nombre) {
$this->elements[$type] -= $nombre;
return true;
} else { return false; }
}
}
```


Fonctionnalités essentielles

Objectifs du chapitre

Gestion des connexions et des utilisateurs ;

Gestion des sessions ;

Autres fonctionnalités essentielles.

Connexions et utilisateurs

Objectifs de la section

Gestion des utilisateurs - authentification ;

Gestion des connexions.

Gestion des utilisateurs - authentification

Les fonctions d’authentification HTTP ne sont disponibles que si PHP est exécuté comme module Apache (et non pas comme CGI) ;

On utilise la fonction `header()` pour demander une authentification (`Authentication Required`) au client, générant ainsi l’apparition d’une fenêtre de demande de nom d’utilisateur et de mot de passe ;

Une fois que les champs ont été remplis, l'URL sera de nouveau appelée mais avec les variables suivantes mises à jour :

– `$PHP_AUTH_USER` : utilisateur,
– `$PHP_AUTH_PW` : mot de passe, et
– `$PHP_AUTH_TYPE` : type d'authentification.

Actuellement, seule l'authentification basique (`Basic`) est supportée ;

Ex : page d'authentification

```
<?php if( !isset($PHP_AUTH_USER)) {
Header("WWW-Authenticate : Basic realm='Titre'");
Header("HTTP/1.0 401 Unauthorized");
echo "Texte à envoyer au client en cas
d'annulation";
exit;
} else {
echo "Bonjour $PHP_AUTH_USER.<br>";
echo "Votre mot de passe est $PHP_AUTH_PW.";
}?>
```

Gestion des connexions

Le statut des connexions est conservé en interne par PHP ;

Il y a trois états possibles :
– 0 / `NORMAL` : le script PHP travaille,
– 1 / `ABORTED` : le client distant s'est déconnecté,
– 2 / `TIMEOUT` : la durée maximale d'exécution est dépassée.

Par défaut, le script PHP se termine dès que le client se déconnecte mais on peut changer cette politique de différentes façons :
– continuer le script comme si le client était encore là en activant l'option :
  • `ignore_user_abort` dans `php3.ini/php.ini`, ou
  • `[php3_ignore_user_abort/ignore_user_abort` dans `apache.conf`.
– exécuter une fonction de fermeture préalablement enregistrée grâce à la fonction `register_shutdown_function()`.

La fonction enregistrée avec `register_shutdown_function()` sera aussi appelée à la fin du script quand celui-ci se termine normalement ;

Pour pouvoir avoir un comportement différent suivant l'état du script lors de sa finalisation on peut tester l'état *déconnecté* du script avec la fonction `connection_aborted()` ;

La fonction `connection_status()` permet également de retourner l'état du script (0, 1, 2 ou 3) ;

On peut modifier le délai de *timeout* (30 secondes par défaut) de plusieurs manières :

– fonction `set_time_limit()`,
– `max_execution_time` dans `php3.ini/php.ini`, ou
– `php3_max_execution_time/max_execution_time` dans `apache.conf`.

La fonction enregistrée avec `register_shutdown_function()` sera également appelée lorsqu'un *timeout* intervient ;

On peut tester l'état *timeout* du script avec la fonction `connection_timeout()` ;

Gestion des sessions

Objectifs de la section

Principe ;

Sessions avec PHPlib ;

Sessions avec PHP4.

Principe

La gestion des sessions avec PHP est un moyen de sauver des informations entre deux accès (requêtes HTTP) ;

Cela permet notamment de construire des applications personnalisées, et d'accroître les fonctionnalités de votre site ;

Malheureusement, la gestion des sessions n'était pas inclue dans les fonctionnalités standards de PHP jusqu'à la version PHP4 ;

Une bibliothèque complémentaire a donc été développée pour gérer les sessions avec PHP3, c'est **PHPlib** ;

Sessions avec PHPlib

PHPlib est une bibliothèque PHP libre qui vise à faciliter la création de sites avec PHP ;

Elle est toujours utilisable avec PHP4 et présente d'ailleurs des fonctionnalités que ce dernier ne possède toujours pas en natif ;

Elle se compose de classes et de fonctions qui permettent :
- le suivi des sessions,
- l'utilisation de variables persistantes au niveau utilisateur et au niveau session,
- l'authentification et la gestion des permissions,
- la génération rapide de code HTML...

Installation :

– récupérer l'archive (`http ://phplib.netuse.de`),
– la décompresser dans un répertoire quelconque (ex : `/usr/lib/phplib/`),
– éditer le fichier `php3.ini/php.ini` et fixer les options de la façon suivante :
  - `magic_quotes_gpc = Off`
  - `magic_quotes_runtime = Off`
  - `magic_quotes_sybase = Off`
  - `track_vars = On`
  - `auto_prepend_file =` <répertoire contenant `prepend.php`>
  - `include_path =` <répertoire d'installation>

Gestion des sessions

Alcôve - Programmer en PHP

Pour l'utilisation de PHPlib, il faut créer quelques fichiers en plus des scripts PHP :

– `prepend.php` sert à définir toutes les librairies PHPlib à inclure avant chaque fichier, et
– `local.inc` sert à définir les classes qui vont être utilisées par les scripts.

Pour gérer les sessions et les variables persistantes, la librairie se sert d'une base de données, d'un annuaire LDAP ou encore d'un simple fichier ;

Cette base de données et les différentes tables qui sont utilisées par les scripts utilisant PHPlib doivent être créées par vos soins grâce aux fichiers de configurations fournis (généralement dans `/usr/lib/phplib/stuff/`) ;

Ex : base de données MySQL

```
mysql -h <ma_machine_hote> <ma_base>
<create_database.mysql
```


Pour gérer une session, PHPlib utilise un identifiant qui est transmis de script en script au cours de la navigation ;

Pour les sessions sans reprise, cet identifiant peut être transmis dans l'URL ;

Pour gérer les reprises de sessions, PHPlib peut stocker l'identifiant dans un cookie qu'il transmet au client à la fin de chaque script (fonction `page_close()`) ;

PHPlib définit des classes pour accéder aux bases de données et gérer les sessions ;

Vous devez définir vos propres classes qui dérivent des classes génériques de PHPlib ;

Ces classes sont généralement stockées dans le fichier `local.inc` ;

Toutes les pages gérant la session doivent être encadrées par :
– `page_open(array("sess" => "MaSession"))`, et
– `page_close()` ;

Si on ne souhaite pas utiliser l'option `auto_prepend_file` pour inclure le fichier `prepend.php`, on peut commencer le script par un `include()` de ce fichier ;

Pour rendre une variable persistante, on utilise la méthode `register()` de la classe `MaSession` ;

Ex : session.php

```
<?php
page_open(array("sess" => "MaSession"));
if( !isset($toto)) $toto = 1;
print ++$toto;
$sess->register("toto");
page_close();
?>
```


Ex (suite) : local.php

```
<?php
class MaBaseSql extends DB_Sql {
var $Host = "ma_machine_hote";
var $Database = "ma_base";
var $User = "mon_utilisateur";
var $Password = "mon_password";
}
class MonContainerSql extends CT_Sql {
var $database_class = "MaBaseSql";
var $database_table = "ma_table";
}
```

```
class MaSession extends Session {
var $classname = "MaSession";
var $mode = "cookie";
var $lifetime = 6000;
var $that_class = "MonContainerSql";
}
?>
```


Ex (fin) : `prepend.php`

```
<?php
if (!is_array($_PHPLIB)) $_PHPLIB["libdir"] = "";
require($_PHPLIB["libdir"] . "db_mysql.inc");
require($_PHPLIB["libdir"] . "ct_sql.inc");
require($_PHPLIB["libdir"] . "session.inc");
require($_PHPLIB["libdir"] . "local.inc");
require($_PHPLIB["libdir"] . "page.inc");
?>
```

Sessions avec PHP4

Le principe est le même qu'avec PHPlib : on attribue à un nouvel utilisateur un identificateur de session (`SID`) ;

Ce numéro est transmis soit sous forme de cookie soit dans les URLs ;

Quand un visiteur accède à la page, PHP4 vérifie si un identificateur de session a été transmis avec la requête ;


Gestion des sessions
Alcôve - Programmer en PHP

Cette vérification peut être faite de trois manières différentes :

– automatiquement si l'option `session.auto_start` est activée dans `php.ini`,
– explicitement par l'appel de la fonction `session_start()`, ou
– implicitement par l'appel de la fonction `session_register()`.

Dans le cas où le numéro de session est valide, tout l'environnement de celle-ci est restauré ;

L'option `register_globals` de `php.ini` autorise les variables globales à être sauvegardées dans une session ;

Ex :

```
<?php session_register("toto");
$toto++;?>
```

L'option `track_vars` de `php.ini` (qui est bloquée à `1` pour les versions de PHP4 supérieures à la 4.0.3) autorise les variables de $HTTP_SESSION_VARS à être sauvegardées dans une session ;

Ex :

```
<?php session_register("toto");
$HTTP_SESSION_VARS["toto"]++;?>
```


Il existe deux manières pour transmettre le numéro de session au sein de l'URL :

– manuellement grâce à la constante `SID` qui est une chaîne de caractères de la forme `session_name=session_id` (ou vide),
– automatiquement si PHP4 a été compilé avec l'option `--enable-trans-sid`.

Ex : compteur de visites

```
<?php session_register("compteur");

$compteur++;?>

Vous avez vu cette page <? echo $compteur;?>
fois.

<!-- <?=SID?> sert à conserver le numéro de
session au cas où les cookies sont desactivés -->

Pour continuer cliquez <A
href="compteur.php?<?=SID?>">ici</A>
```


Gestion des sessions

Alcôve - Programmer en PHP

Les fonctions les plus utilisées sont :

- `session_start()` : crée ou restaure une session,
- `session_destroy()` : détruit une session,
- `session_register()`/`session_unregister()` : sauvegarde/oublie une variable,
- `session_set_save_handler()` : permet de définir ses propres fonctions de sauvegarde de sessions (pour stocker les données dans une base de données par exemple).

Autres fonctionnalités

Objectifs de la section

Récupération et traitement des erreurs ;

Les cookies.

Récupération et traitement des erreurs

Il existe 4 types d'erreurs et d'alertes PHP :
- E_ERROR (1) : erreur d'exécution
- E_WARNING (2) : alerte
- E_PARSE (4) : erreur d'analyse
- E_NOTICE (8) : notes (alertes qui peuvent être ignorées)

Pour définir un niveau d'erreur, on ajoute les chiffres des erreurs que l'on souhaite prendre en compte (`bitmask` par défaut 7 = 4 + 2 + 1) ;

Ce niveau peut être changé de trois manières :
- directive `error_reporting` dans `php3.ini/php.ini`,
- directive `php3_error_reporting/error_reporting/error_reporting` dans `httpd.conf`,
- fonction `error_reporting()`.

Toutes les expressions PHP peuvent être appelées avec le préfixe @ qui permet d'ignorer le rapport d'erreur pour cette fonction particulière ;

Si une erreur survient dans une telle expression, et que l'option `track_errors` est activée dans `php3.ini/php.ini`, on retrouve le message d'erreur dans la variable globale `$php_errormsg` ;

La fonction `error_reporting()` établit le niveau d'erreur à prendre en compte et renvoie l'ancien niveau ;

La fonction `error_log()` envoie un message d'erreur :
– dans les logs du serveur,
– à un port TCP, ou
– à un fichier.

Elle prend comme paramètres :
– `message` : message qui doit être envoyé,
– `message_type` : type de transmission choisi :
  - 0 : système standard de log de PHP,
  - 1 : mail,
  - 2 : machine distante (TCP/IP), ou
  - 3 : fichier local.

Ex : `error_log("Alerte rouge", 1, "webmaster@monsite.fr");`

Les cookies

Les cookies sont un mécanisme d'enregistrement d'informations sur le disque du client mais aussi de relecture de ces informations ;

Ce système permet d'authentifier et de suivre les visiteurs ;

PHP supporte les cookies de manière transparente ;

Pour envoyer un cookie, on utilise la commande `setcookie()` ;

Les Cookies font partie de l'entête HTTP, ce qui impose que `setcookie()` soit appelée avant tout affichage sur le client (idem `header()`) ;

Avec PHP4, on peut contourner cette limitation en bufferisant la sortie standard grâce à la fonction `ob_start()` puis en libérant celle-ci grâce à la fonction `ob_end_flush` ;

On peut aussi indiquer à PHP4 de bufferiser automatiquement la sortie grâce à l'option `output_buffering` de `php.ini` mais ceci ralentit sensiblement PHP4 ;

Tous les cookies qui sont envoyés au client sont automatiquement retournés au script PHP et placés dans le tableau associatif `$HTTP_COOKIE_VARS` ;

Seuls les cookies qui correspondent au nom de domaine de la page demandée sont envoyés au serveur par le navigateur ;

Pour affecter plusieurs valeurs à un seul cookie il faut ajouter `[]` au nom du cookie ;

Autres fonctionnalités

Alcôve - Programmer en PHP

Ex : `cookie_ecriture.php`

```
<?php setcookie("MonCookie", "Contenu", time() + 3600);?>
<HTML><BODY><H1>
Maintenant, vous l’avez !<br>
Allez sur la page
<a href = "cookie_lecture.php">précédente</a>.
</H1></BODY></HTML>
```

Ex (suite) : `cookie_lecture.php`

```
<HTML><BODY><H1>
<?php if
(isset($HTTP_COOKIE_VARS["MonCookie"])) :?>
Le cookie MonCookie contient :
<?php print $HTTP_COOKIE_VARS["MonCookie"];
else :?>
Vous n’avez pas le cookie MonCookie.<br>
Allez le chercher <a href =
"cookie_ecriture.php">ici</a>.
<?php endif;?>
</H1></BODY></HTML>
```


Utilisation avancée

Objectifs du chapitre

Interopérabilité avec des applicatifs ;

Traitement des chaînes de caractères ;

Fonctionnalités utiles.

Interopérabilité avec des applicatifs

Objectifs de la section

Utilisation des SGBDs ;

Interfaçage avec l’API Apache ;

Utilisation de LDAP.

Utilisation des SGBDs

Un grand nombre de **S** ystèmes de **G** estion de **B** ases de **D** onnées (**SGBD**) sont supportés par PHP ;

La plupart sont accessibles via le langage **SQL** (**S** tructured **Q** uery **L** angage) ;

Celui-ci permet, comme son nom l’indique, de réaliser des requêtes sur les bases de données et ceci de façon structurée et standardisée ;

L’utilisation des bases des données est la clé de voûte des sites dynamiques et de commerce électronique ;

Bases SQL supportées :

Adabas D	Empress	IBM DB2
Informix	Interbase	mSQL
MySQL	ODBC (Access, MS-SQL, iODBC...)	Oracle
PostgreSQL	Raima Velocis	Solid
Sybase		

Autres SGBD supportés :

– dBase,
– filePro (lecture seule),
– dbm (ndbm, gdbm, Berkeley db).

Interopérabilité avec des applicatifs

Alcôve - Programmer en PHP

Principales commandes SQL :

– `CREATE TABLE` : crée une nouvelle table,
– `DELETE` : supprime des lignes d’une table,
– `INSERT` : insère une nouvelle ligne dans une table,
– `SELECT` : récupère des lignes d’une table ou d’une vue,
– `UPDATE` : modifie des valeurs dans des champs.

Ex : création et alimentation d'une table

```
CREATE TABLE Personne (Secu INT8 NOT NULL PRIMARY
KEY, Nom VARCHAR(20) NOT NULL, Prenom VARCHAR(20)
NOT NULL)

INSERT INTO Personne VALUES ('132456789', 'Smith',
'John')

INSERT INTO Personne VALUES ('987654321', 'Gates',
'Bill')
```


Interopérabilité avec des applicatifs

Alcôve - Programmer en PHP

Ex (suite) : affichage de la table et suppression d'une entrée

```
SELECT * FROM Personne
```

Secu	Nom	Prenom
132456789	Smith	John
987654321	Gates	Bill

```
DELETE FROM Personne WHERE Nom = 'Gates'
```

Chaque base SQL possède sa propre API mais les mécanismes de consultation sont globalement les mêmes pour toutes :

– connexion au serveur de bases de données (machine, utilisateur, mot de passe),
– sélection de la base à utiliser (nom de la base),
– requête SQL,
– exploitation de la requête (affichage...),
– déconnexion.

Bien sûr avant d'utiliser une base de données particulière, il faut s'assurer que la librairie correspondante est chargée dans PHP ;

Pour une base de données MySQL, il faut ajouter la ligne `extension=mysql.so` dans le fichier `php3.ini/php.ini` et vérifier que le fichier `mysql.so` se trouve bien dans le répertoire `extension_dir` ;

Les fonctions MySQL les plus couramment utilisées sont les suivantes :

– `mysql_connect()` : ouvre une connexion à un serveur MySQL,
– `mysql_list_*()` : retournent la liste des bases, tables ou champs disponibles sur le serveur,
– `mysql_select_db()` : selectionne une base de données sur le serveur,

– `mysql_query()` : envoie une requête SQL sur la base de données active sur le serveur et renvoie un identifiant de résultat :
  - `mysql_result()` : exploite le résultat,
  - `mysql_fetch_*()` : exploitent et manipulent le résultat,
  - `mysql_field_*()` : permettent de naviguer au sein du résultat.
– `mysql_close()` : ferme la connexion au serveur MySQL (fait par défaut à la fin du script).

Il est possible de réaliser une connection persistante à un serveur MySQL en utilisant la fonction `mysql_pconnect()` ;

Une telle connexion ne sera pas fermée automatiquement à la fin du script ;

Accélère sensiblement les scripts quand les connexions à la base sont fréquentes ;

La connexion sera refermée automatiquement après un certain temps d’inutilisation ;

Ex :

```
<?php
mysql_pconnect("db.mon-server.com", "username",
"password");
mysql_select_db("ma_base");
$resultat = mysql_query("SELECT * FROM ma_table");
if ($resultat) :?>
<TABLE>
<TR><TH>Nom</TH> <TH>Description</TH></TR>
```



```
<?php
while ($tableau = mysql_fetch_array($resultat))
echo "<TR><TD>", $tableau["nom"], "</TD> <TD>",
$tableau["descr"], "</TD></TR>\n";
?>
</TABLE>
<?php else :
echo "Votre table est vide";
endif;?>
```

Il existe des outils d'administration de bases de données entièrement écrits en PHP ;

– **PHPMyAdmin** pour les bases MySQL (`http ://www.phpwizard.net/phpMyAdmin/`),
– **PHPPgAdmin** pour les bases PostgreSQL (`http ://www.phpwizard.net/phpPgAdmin/`).

Ces outils sont très utiles pour créer les bases de données avant de les utiliser dans des scripts PHP ;

Ils sont également pratique pour maintenir les bases ;

Ces outils étant libres, leur code source peut servir de référence pour accéder aux bases de données avec PHP ;

Interfaçage avec l'API Apache

La fonction `getallheaders()` permet de récupérer tous les en-têtes HTTP correspondant à la requête courante ;

Les en-têtes sont stockés dans un tableau associatif ;

Ex :

```
$headers = getallheaders();
while (list($header, $valeur) = each($headers)) {
echo "$header : $valeur <BR>\n";
}
```

Utilisation de LDAP

L ightweight **D** irectory **A** ccess **P** rotocol ;

Protocole normalisé d’accès aux services d’annuaires ;

Un annuaire est équivalent à une base de données où les informations sont rangées de manière hiérarchique ;

Les caractéristiques de LDAP :
– organisation globale,
– standard ouvert,
– extensible et paramétrable,
– stockages de données hétérogènes,
– sécurisé.

LDAP utilise une syntaxe à nom distincts (dn : *distinguished names*) ;

Ex : dn et signification

```
dn = "cn=Linus Torvalds, ou=Devel, o=Transmeta, c=USA"
country = USA
organization = Transmeta
organizationUnit = Devel
commonName = Linus Torvalds
```

Avant de pouvoir utiliser LDAP avec PHP, il faut ajouter la ligne `extension = ldap.so` dans `php3.ini/php.ini` et vérifier que la librairie `ldap.so` se trouve bien dans le répertoire `extension_dir` ;

Une séquence type de consultation/modification d'une base LDAP avec PHP prendra la forme suivante :

– `ldap_connect()` : établit une connexion avec un serveur LDAP,
– `ldap_bind()` : liaison et authentification sur le serveur,
– `ldap_get_*()`/`ldap_add()`/`ldap_modify()`... : consultation/modification de la base,
– `ldap_close()` : déconnexion.

Ex : recherche de tous les dn commançant par `l`

```
<?php
$ds = ldap_connect("ldap.mon-serveur.com");
if ($ds) {
ldap_bind($ds);
$sr = ldap_search($ds, "ou=Devel, o=Transmeta,
c=USA", "dn=l*");
$info = ldap_get_entries($ds, $sr);
```

```
for ($i=0; $i < $info["count"]; $i++) {
echo "dn vaut : " . $info[$i]["dn"] . "<br>";
echo "première entrée cn vaut : " .
$info[$i]["cn"][0] . "<br>";
echo "première email vaut : " .
$info[$i]["mail"][0] . "<br>";
}
ldap_close($ds);
} else {
echo "<H4>Impossible de se connecter au serveur
LDAP</H4>";
}?>
```


Traitement des chaînes de caractères

Objectifs de la section

Manipulation des chaînes de caractères ;

Expressions régulières.

Manipulation des chaînes de caractères

Affichage :

- `echo()` : affiche une ou plusieurs chaînes séparées par des virgules,
- `print()` : affiche une chaîne,
- `printf()` : affiche une chaîne selon une chaîne de formatage (idem langage C).

Ex :

```
$nom = "toto";

echo "Bonjour", $nom; # affiche Bonjour toto

print "Bonjour $nom"; # affiche Bonjour toto

printf("Bonjour %s", $nom); # affiche Bonjour toto
```

Substitutions :

- `addslashes()` : ajoute un backslash devant tous les caractères spéciaux,
- `stripslashes()` : enlève les backslashs ajoutés par la fonction addslashes,
- `str_replace()` : remplace toutes les occurrences d'une chaîne par une autre,

Ex :

```
print addslashes('\' \ "');# affiche \' \ \"

print str_replace("toto", "titi", "Bonjour
toto !");# affiche Bonjour titi !
```


Découpage :

- `explode()` : scinde une chaîne en morceaux grâce à un délimiteur,
- `implode()/join()` : regroupe tous les éléments d'un tableau dans une chaîne en ajoutant une chaîne de jointure,
- `ltrim()/chop()` : enlève les espaces de début/fin de chaîne.

Ex :

```
$composants = "resistance, condensateur,
transistor ";

$composants = chop($composants);

$tableau = explode(", ", $composants);

print $tableau[2] . ".";# affiche transistor.
```

Web :

– `rawurlencode()`/`rawurldecode()` : encode/décode une chaîne en URL selon la RFC1738,
– `htmlspecialchars()`/`htmlentities()` : converti tous les caractères spéciaux en équivalent HTML.

Ex :

```
$url = rawurlencode('bonjour@toi /toto/');

print $url; # affiche bonjour%40toi%20%2Ftoto%2F

$html = htmlentities("2 < 3");

print $html; # affiche 2 < 3 mais le source HTML contient

2 < 3
```


Comparaison :

– `parse_str()` : analyse une chaîne et en déduit des variables et leur valeur,
– `strcmp()` : comparaison sensible à la casse,
– `strlen()` : retourne la longueur de la chaîne.

Ex :

```
parse_str("toto=23& tab[]=Ceci+fonctionne &
tab[]=aussi");

print "$toto, $tab[0], $tab[1]";
```

Affiche `23, Ceci fonctionne , aussi`

Casse :
- `strtolower()`/`strtoupper()` : met tous les caractères en minuscule/majuscule,
- `ucfirst()` : force le premier caractère d'une chaîne en majuscule,
- `ucwords()` : force le premier caractère de chaque mot d'une chaîne en majuscule.

Ex :

```
$nom = "BiLL GaTeZ";
print $nom = strtolower($nom); # affiche bill gatez
print strtolower($nom); # affiche Bill Gatez
```



Traitement des chaînes de caractères
Alcôve - Programmer en PHP

Expressions régulières

Les expressions régulières (abrv : *regex*) sont utilisées pour faire des manipulations complexes sur les chaînes de caractères (ex : substitutions ou recherches complexes) ;

PHP gère deux types d'expressions régulières :
- les expressions régulières avancées de POSIX, et
- les expressions régulières modifiées du langage Perl.

Principes de base des expressions régulières :

– recherche de séquences de caractères au sein d'une chaîne,
– besoin d'expressions pour décrire ces séquences,
– utilisation d'une syntaxe puissante,
– équivaut à la définition de conditions de recherche.

Signification des principales expressions régulières :

– c : caractère c (sauf caractères spéciaux),
– `\c` : caractère spécial c (sauf chiffres de 1 à 9),
– `^` : début de la ligne,
– $: fin de la ligne,
– `.` : n'importe quel caractère,
– `[s]` : caractère appartenant à l'ensemble s, où s est une suite de caractères et/ou une échelle de caractères (`[c-c]`),

- `[^s]` : caractère n'appartenant pas à l'ensemble `s`,
- `r*` : 0, 1 ou plusieurs occurrences successives de l'expression régulière `r`,
- `rx` : `r`, suivie de l'expression régulière `x` (concaténation),
- `r{m,n}` : un nombre (entre `m` et `n`) d'occurrences successives de `r`,
- `r{m}` : exactement `m` occurrences successives de `r`,
- `r{m,}` : au moins `m` occurrences successives de l'expression régulière `r`.

Utilisation avec PHP :

- `ereg()/eregi()` : expression régulière standard sensible/insensible à la casse,
- `ereg_replace()`/`eregi_replace()` : expression régulière de substitution sensible/insensible à la casse,
- `split()` : découpe une chaîne grâce à un délimiteur défini par une expression régulière,
- `sql_regcase()` : prépare une expression régulière insensible à la casse (utile pour les fonctions ne supportant pas les recherches insensible à la casse).

Ex :

```
<?php $personne = "Nom : Smith, Prénom : John,
Age : 52, Divers : blond";
if (ereg("blond", $personne))
print "Cette personne est blonde.";
# Affiche Cette personne est blonde.
if (eregi("age : *([0-9]+)", $personne,
$resultat))
print "Cette personne a $resultat[1] ans.";
else print "Cette personne n’a pas d’age.";
# Affiche Cette personne a 52 ans.
```



```
print eregi_replace("(age :) *[0-9]+.*", "\1 68",
$personne);
# Affiche Nom : Smith, Prénom : John, Age : 68
print sql_regcase("John Smith");
# Affiche [Jj][Oo][Hh][Nn] [Ss][Mm][Ii][Tt][Hh]
?>
```

Fonctionnalités utiles

Objectifs de la section

Manipulation et traitement des fichiers ;

Arguments et gestion des URLs ;

Création dynamique d'images ;

Fonctions mathématiques ;

PHP et XML ;

Templates avec PHPlib ;

Utilisation de PHP comme langage de script.

Manipulation et traitement des fichiers

Quel que soit le langage de programmation, la gestion des fichiers est toujours importante pour le stockage de données ;

La gestion des fichiers en PHP est issue du langage C mais elle comporte quelques fonctions supplémentaires très utiles (comme la gestion des fichiers distants par exemple) ;

Les principales fonctions de manipulation de fichiers sont les suivantes :

- `fopen()` : ouverture d'un fichier (possibilité de fournir une URL HTTP ou FTP pour les fichiers distants),
- `fclose()` : fermeture du fichier,
- `fpassthru()` : lit un fichier en entier et l'affiche,
- `fread()`/`fgets()` : lit n caractères dans le fichier,
- `fwrite()`/`fputs()` : écrit une chaîne dans un fichier,
- `rewind()`/`fseek()`/`ftell()` : positionnement au sein du fichier.

Ex : affichage de l'image `image.png`

```
<?php
Header("Content-type : image/png");
if (!$fd = fopen("image.png", "rb"))
echo "Impossible d'ouvrir le fichier.";
else fpassthru($fd);
?>
```


Ex2 : fichier de log

```
<?php
...
if (!$fd = fopen("mon_error.log", "a"))
echo "Impossible d'ouvrir le fichier.";
else fputs($fd, "Erreur : $errormsg\n");
# Ecrit le dernier message d'erreur renvoyé dans mon_error.log
...
?>
```

Les principales fonctions de gestion de fichiers sont les suivantes :

– `file_exists()` : test de l'existence d'un fichier,
– `copy()` : copie un fichier,
– `rename()` : renomme un fichier,
– `unlink()` : efface définitivement un fichier.

Ex :

```
<?php
$fichier = "toto.html"
if (file_exists($fichier))
copy($fichier, "/tmp/" . $fichier);
else echo "Impossible d'ouvrir le fichier.";
?>
```


Les principales fonctions de manipulation des répertoires sont les suivantes :

– `chdir()` : changement de répertoire courant,
– `opendir()` : ouverture d'un répertoire,
– `closedir()` : fermeture du répertoire,
– `readdir()` : lit l'entrée suivante dans le répertoire,
– `rewinddir()` : revient au début du répertoire,
– `mkdir()` : crée un nouveau répertoire,
– `rmdir()` : supprime un répertoire,
– `dir()` : instanciation d'un objet répertoire pour une manipulation objet de celui-ci,

Ex : équivalent de la commande `ls`

```
<?php
chdir("/tmp");
$dir = dir(".");
$dir->rewind();
while ($fichier = $dir->read())
echo "$fichier<br>";
$dir->close();
?>
```


Arguments et gestion des URLs

La gestion des URLs est importante pour encoder/décoder des informations au sein même de celles-ci ;

On peut utiliser l'URL pour transmettre des informations (variables...) d'un script à l'autre ;

Ce mécanisme peut constituer une mini-gestion de sessions par exemple ;

Les principales fonctions utilisées pour la gestion des URLs sont les suivantes :

– `parse_url` : analyse une URL et retourne ses composants sous forme d'un tableau associatif,
– `urlencode`/`urldecode` : encode/décode une chaîne en remplaçant les caractères spéciaux par des %xx et les espaces par des + (`application/x-www-form-urlencode`),
– `base64_encode`/`base64_decode` : encode/décode une chaîne en base64 pour permettre à certains systèmes de manipuler les informations binaires sur 8 bits (ex : corps de mail).

Ex : transmission d'une variable par l'URL

```
<?php $chaine = "Bonjour tout le monde?";?>

<A href="mon-cgi?chaine=<?php echo
urlencode($chaine);?>">Envoyer</A>
```

Appelle le script CGI `mon-cgi` avec l'URL
`"mon-cgi?chaine=Bonjour+tout+le+monde+%3F"`

Création dynamique d’images

PHP n’est pas limité à la création de fichiers HTML, il peut aussi servir à générer dynamiquement des images (PNG, JPEG, GIF) ;

Ces images peuvent être émises directement vers le client ou sauvegardées sur le serveur ;

Elles sont très pratiques pour dessiner rapidement des graphiques à partir de données dynamiques (ex : statistiques du site, résultats d’un sondage...) ;

Pour générer des images, PHP utilise la librairie GD ;

Celle-ci se compose d’un ensemble de fonctions qui permettent de créer des images dynamiques de qualité avec très peu de code ;

Historiquement, le format GIF est le premier à avoir été supporté mais aujourd’hui on lui prefère les formats JPEG et surtout PNG, le format le plus ouvert de tous ;

Bien sûr avant d’utiliser cette librairie il faut au préalable la charger dans PHP en ajoutant la ligne `extension=gd.so` dans le fichier `php3.ini/php.ini` et en vérifiant que le fichier `gd.so` se trouve bien dans votre `extension_dir` ;

L'API est très intuitive :

- `ImageCreate*()` : pour créer une image blanche ou à partir d'une image existante,
- `ImageColor*()` : pour définir une couleur et la transparence,
- `ImageString()` : pour écrire du texte,
- `Image<forme>()` : pour dessiner des formes géométrique connue (rectangle, arc, ligne, polygone...),
- `ImageFill*()` : pour colorier des formes,
- `Image<attribut>()` : pour connaître un attribut (largeur, hauteur, taille...) de l'image,
- `Image<type>()` : pour envoyer une image de type PNG, JPEG ou GIF au navigateur,
- `ImageDestroy()` : pour détruire l'image...

Toutes les fonctions de l'API manipulent un pointeur sur l'image courante ;

Ce pointeur est retourné par la fonction `ImageCreate*()` à la création de l'image ;

Typiquement, on crée un fichier PHP dont le seul but est de retourner une image dynamique ;

Les arguments pour créer l'image sont alors passés dans l'URL à la manière des formulaires en mode GET par exemple (ex : `<IMG src = "bouton.php?texte=TITRE+BOUTON">`)

Le fichier doit retourner un en-tête correspondant à l'image retournée (ex : `Header("Content-type : image/gif");`) afin que le navigateur puisse correctement l'afficher ;

On peut aussi créer une fonction qui génère l'image mais cette technique est moins facile à utiliser car il faut s'occuper de stocker l'image puis éventuellement de la détruire par la suite ;

Ex : génération dynamique de boutons, `bouton.php`

```
<?php Header("Content-type : image/gif");

if (!isset($texte)) $texte = "";

$image = ImageCreateFromGif("images/bouton.gif");

$couleur = ImageColorAllocate($image, 255, 255,
255);

$espace = (ImageSX($image) - 7.5 * strlen($texte))
/ 2;

ImageString($image , 5, $espace, 9, $texte,
$couleur);

ImageGif($image);

ImageDestroy($image);?>
```

Avant : 

```
<IMG src = "bouton.php?texte=Mon+titre">
```

Après : 


Fonctionnalités utiles
Alcôve - Programmer en PHP

Fonctions mathématiques

Il existe deux types de fonctions mathématiques dans PHP :
– les fonctions standards, qui s'appliquent sur des entiers long (`double`), et
– les fonctions de précision où les nombres sont représentés par des chaînes de caractères.

La constante `M_PI` est définie pour le nombre Pi ;

Les fonctions standards les plus utilisées sont les suivantes :

– `abs()` : valeur absolue,
– `number_format()` : formate un nombre par groupe de milliers,
– `round()`/`ceil()`/`floor()` : arrondis,
– `max()`/`min()` : encadrements,
– `pow()`/`sqrt()` : puissances,
– `exp()`/`log()`/`log10()` : exponentielles et logarithmes,

– `cos()`/`sin()`/`tan()`/`acos()`/`asin()`/`atan()` : trigonométrie,
– `base_convert()`/`BinDec()`/`DecBin()`/`HexDec()`/ `DecHex()`/`OctDec()`/`DecOct()` : changements de bases,
– `rand()`/`srand()`/`getrandmax()` : nombres pseudo-aléatoires (les fonctions `mt_*` sont meilleures et plus rapides).

Ex :

```
$x = -12;
print sqrt(pow($x, 2)) . " = " . abs($x);
# Affiche 12 = 12
print number_format(round(12550.49999999), 3, ",",
" ");
# Affiche 12 550,000
print cos(M_PI/4) . " = " . 1/sqrt(2);
# Affiche 0.70710678118655 = 0.70710678118655
```


PHP et XML

e **X** tensible **M** arkup **L** anguage ;

Standard ouvert de description de données ;

Utilise des marqueurs qui, à la différence des autres ML, décrivent les données et non pas la manière dont celles-ci doivent être affichées (ex : HTML) ;

L'utilisation de XML avec PHP nécessite l'installation de la librairie **Expat** (`http ://www.jclark.com/xml/`) et la compilation de PHP avec l'option `--with-xml` ;

Depuis la version 1.3.7 d'Apache, la librairie Expat est directement intégrée au serveur web ;

Le module XML permet de créer un parseur XML appelant des fonctions définies par l'utilisateur et permettant de traiter les données incluses dans les tags XML ;

Fonctions principales :

– `xml_parser_create()` : crée un parseur XML et retourne un descripteur de parseur qui sera utiliser par les autres fonctions,
– `xml_set_element_handler()` : modifie les options associées à un parseur (sensibilité à la casse, encodage des caractères...),
– `xml_set_character_data_handler()` : définit les fonctions à appeler à l'ouverture et à la fermeture d'un tag XML spécifique,
– `xml_parse()` : le parseur XML,
– `xml_parser_free()` : élimination d'un parseur XML de la mémoire.

On ne peut pas directement manipuler un parseur XML à l'intérieur d'un objet PHP ;

Il faut pour cela définir un parseur comme variable d'instance de l'objet et passer l'objet par référence à la fonction `xml_set_object()` ;

Ex : `xml_set_object(parser, &object);`

Par la suite, il faut utiliser les fonctions `xml_set_*()` avec `$this->parser` comme identifiant de parseur.

Utilisation des templates PHPlib

PHPlib propose une classe `Template` permettant de séparer le squelette HTML d'une page, du traitement des données permettant de modifier dynamiquement cette page ;

Les parties dynamiques de la page sont représentées dans le code HTML par des identifiants entourés d'accolades ;

Ces identifiants seront remplacées par leur valeur calculée par le script PHP ;

Ex : le template (`expl.ihtml`)

```
<html><head>
<title>{LETITRE}</title>
</head >
<body>
<br>Un exemple de variable :{ICI}
</body></html>
```

Ex (suite) : code PHP utilisant le template `expl.ihtml`

```
include("template.inc")
$tmpl = new Template("rep-template");
$tmpl->set_file("main", "expl.ihtml");
$tmpl->set_var(array(
"LETITRE"=>"Exemple de titre",
"ICI"=>"Un message"));
$tpml->parse("out", "main");
$tmpl->p("out");
```


Fonctions associées aux templates :

- `template()` : retourne un objet Template associé au répertoire où sont stockés les squelettes HTML,
- `set_file()` : associe un fichier template à une variable qui représente celui-ci,
- `set_var()` : affecte une valeur à un nom de variable du template,

Ces deux dernières fonctions peuvent aussi être appelées avec comme unique argument un tableau contenant des paires clefs/valeurs ;

– `parse()` : cette fonction substitue la valeur de toutes les variables définies dans la variable de référence puis stocke (ou ajoute) le résultat dans la variable cible,
– `p()` : cette fonction affiche la valeur de la variable passée en argument dans le navigateur.

Afin de pouvoir traiter des éléments répétitifs (tableaux, listes de sélection...), on peut déclarer des blocs dans les variables manipulées par la classe `Template` ;

La définition utilise les commentaires HTML ;

Ex :

```
<form action = "{PHPSELF}">
<select name = "variable">
<!-- BEGIN monbloc ->
<option>{MUL}
<!-- END monbloc ->
</select>
<br><input type="submit"></form>
```

Ex (suite) :

```
$tmpl = new Template("rep-template");
$tmpl->set_file("main", "expl.ihtml");
$tmpl->set_bloc("main", "monbloc", "toto");
$tmpl->set_var("PHPSELF", $PHP_SELF);
for ($i = 1; $i < 10; $i++) {
$tmpl->set_var("MUL", $i);
$tmpl->parse("toto", "monbloc", true);
}
$tpml->parse("out", "main");
$tmpl->p("out");
```


Une fois un bloc défini, on peut le répéter en spécifiant `true` comme dernier argument de la fonction parse ;

La fonction `set_block($parent, $blocname, $name)` indique que la variable `$parent` contient un bloc nommé `$blocname`. Cette fonction enlève le bloc de `$parent` et le remplace par une variable `$name`.

Utilisation de PHP comme langage de script

On peut utiliser toute la puissance et la richesse des fonctions de PHP pour faire des scripts non orientés web ;

Il faut avoir au préalable compilé PHP comme CGI (la cohabitation entre plusieurs formes de PHP est possible sur une même machine) ;

La commande `php` donne alors accès à un shell PHP où chaque commande PHP tapée est interprétée immédiatement ;

Bien sûr, seules les commandes encadrées par les `<?php ...?>` seront interprétées les autres étant affichées telles-quelles ;

L'option `-q` permet d'éviter d'avoir un en-tête HTTP envoyé avant le premier affichage ;

Pour réaliser des scripts classiques en PHP il suffit de faire commencer le fichier par la ligne :

```
#!/usr/bin/php<version> -q
```

Ex : script d'invite login, `login.php`

```
#!/usr/bin/php<version> -q
Bonjour monsieur <?php echo $USER, "\n"?>,
vous êtes sur la machine <?php echo $HOSTNAME,
"\n"?>
```

En vérifiant que les variables `$USER` et `$HOSTNAME` sont bien exportées par le shell et en rendant le script `login.php` exécutable (`chmod +x login.php`) la commande `./login.php` affiche :

```
Bonjour monsieur nom_login,
vous êtes sur la machine nom_machine.
```


Conclusion

Objectifs du chapitre

Conseils de programmation PHP ;

L'avenir de PHP ;

Références ;

Glossaire.

Conseils de programmation PHP

Penser en matière de réutilisabilité du code ;

Ne pas lésiner sur les commentaires, ils aideront les autres à comprendre votre code (on est rarement seul à travailler sur un site web) ;

Segmenter le code PHP pour le rendre plus clair :
– préférer les fonctions courtes et mono-tâches,
– utiliser des fichiers différents...

Essayer de séparer au maximum les aspects graphiques (charte graphique, feuilles de styles...), contenu (gestion des données dynamiques...) et mise en forme (affichage des données dynamiques...) :

– utilisation de fichiers d'en-têtes, de barres de navigation et de pieds de pages séparés,
– utilisation de fichiers contenant les classes manipulées par les scripts...

L'avenir de PHP

De plus en plus utilisé sur Internet (plusieurs millions de domaines différents) ;

PHP4 depuis le début de l'année 2000 ;

Cette version est néanmoins contestée (ex : FSF) à cause de ses nouvelles licences ;

Elle apporte cependant plus de rapidité (ex : analyseur Zend) et de nouvelles fonctionnalités natives (ex : gestion des sessions...) ;

Références

Livres :

- Professional PHP Programming (Wrox),
- Programmation Web avec PHP (Eyrolles),
- PHP, précis & concis (O'Reilly),
- Programmation PHP (CampusPress)...

URLs :

- Le site officiel de PHP (`http ://www.php.net`),
- Documentation officielle en anglais (`http ://www.php.net/manual`),
- Traduction française de la documentation officielle (`http ://dev.nexen.net/docs`),
- Le site de Zend (`http ://www.zend.com`),
- Développement PHP/MySQL chez Nexen (`http ://dev.nexen.net`)

- Projets libres autour de PHP (`http ://www.phpwizard.net`),
- Articles techniques sur PHP (`http ://www.phpbuilder.com/`),
- Le site officiel de PHPlib (`http ://phplib.netuse.de`),
- Site Francophone pour l'Aide à la Programmation en PHP (`http ://www.phpfr.org`)...

Glossaire

- **Apache** : serveur web (HTTP) le plus utilisé sur Internet,
- **Browser** : navigateur/butineur de page web,
- **CGI** (*Common Gateway Interface*) : interface de communication pour la création de programmes capables d'être exécutés par un serveur HTTP, elle permet la création dynamique de pages du côté serveur,
- **GET** : méthode HTTP de transmission de formulaires via les URLs (données encodées dans l'URL),

- **GIF** (*Graphics Interchange Format*) : format breveté d'images compressés, très utilisé sur le web,
- **HTML** (*Hyper Text Markup Language*) : langage de description structurale de documents, basé sur la notion d'environnement et de délimiteurs, utilisé pour créer les pages web,
- **HTTP** (*HyperText Transfer Protocol*) : protocole de transfert de fichiers utilisé sur le web,
- **HTTPD** (*HTTP Daemon*) : partie principale d'un serveur web qui échange, via HTTP, des fichiers avec les clients web (navigateurs),

- **JPEG** : format d'images compressés très utilisé sur le web,
- **LDAP** (*Lightweight Directory Access Protocol*) : protocole normalisé d'accès aux services d'annuaires,
- **PHP** (*PHP : Hypertext PreProcessor*) : langage de script côté serveur, embarqué dans les pages HTML,
- **PNG** (*Portable Network Graphics*) : format libre d'images compressés créé pour fournir une alternative libre au format GIF,

- **POST** : méthode HTTP de transmission de formulaires dans le corps de la requête (alternative à la méthode GET),
- **Regex** : raccourci pour *Regular Expressions* , syntaxe pour réaliser des manipulations complexes sur les chaînes de caractères,
- **Session PHP** : mécanisme de sauvegarde d'informations entre deux accès (requêtes HTTP),
- **SGBD** : Système de Gestion de Bases de Données,

- **SQL** (*Structured Query Langage*) : langage structuré de requête pour interroger des SGBDs,
- **URL** (*Uniform Resource Locator*) : syntaxe utilisée pour localiser une ressource (fichier) sur le web (ex : `http ://www.php.net/index.html`)
- **WWW** (*World Wide Web*) : aussi appellé web, c'est le nom donné au réseau Internet,
- **XML** (*eXtensible Markup Language*) : standard ouvert de description de données basé sur l'utilisation de marqueurs decrivant les données qu'ils encapsulent.