L'Altruiste : Le guide des langages Web

Le langage PHP

Sommaire

1 / Introduction

Le langage PHP (Hypertext Preprocessor) est devenu en quelques années d'existence, le langage de programmation de sites web dynamiques le plus populaire.

De même que le serveur web Apache, la plateforme Linux et le gestionnaire de base de données MySQL, **le langage PHP s'appuie sur un logiciel Open Source, libre et gratuit**, et partant, demeure la solution la plus économique pour développer des applications internet à un coût minimum.

PHP est un langage de script fonctionnant essentiellement côté serveur. Les scripts PHP, à l'instar de l'ASP (Active Server Pages) de Microsoft, sont **incorporés directement avec le balisage HTML**, au sein d'une page Web.

Associé à MySQL, le langage PHP permet de développer des applications web puissantes reliées si nécessaire à des bases de données.

Empruntant des concepts à des langages tels que Perl ou C, **PHP est un langage de script capable de fonctionner sur n'importe quelle plateforme**. En effet,chacun des systèmes d'exploitation de Microsoft, Unix, Linux ou encore Mac OS X, peut devenir **un support pour des applications Internet écrite en PHP en accueillant un module spécifique s'intégrant au serveur web en place**. Ainsi, la plupart des serveurs web à l'image de Microsoft IIS (Internet Information Server), Netscape Enterprise Server ou encore Apache, supportent parfaitement la technologie PHP.

En outre, **le langage PHP possède de nombreux outils facilitant sa connectivité à des bases de données** comme non seulement son SGBDR (Système de Gestion de Bases de Données Relationnelles) de prédilection, MySQL, mais aussi quasiment tous les autres tels que Sybase, Oracle, SQL Server ou DBase.

PHP Solutions Magazine n°9 :

- AddWeb 7 - optimisation et promotion des sites Web,
- Maguma Workbench 2.2.0 - un bon IDE pour PHP,
- Cute PHP Library - bibliothèque pour tous les bricoleurs,
- OpenOffice et PHP, comment créer des rapports efficaces pour le Web ?,
- Test des bases de données open Source,
- Search Engine Optimization,
- Flash en PHP : Ming en action,
- Créer une galerie de photos en utilisant EXIF,
- Interface XUL pour PHP,
- Zend PHP Certification,
- Google Adsense - des revenus pour webmaster.

- PHP 5 disponible sur PHP.net.
- MySQL 5.0 disponible sur MySQL.com.
- Apache 2 disponible sur Apache.org.
- EasyPHP 1.8 est disponible sur Easy PHP
- PHP Edit 1.2, un éditeur PHP est disponible sur le site PHPEdit Network

2 / La syntaxe

Comme pour tous les langages de programmation, PHP obéit à certaines règles syntaxiques sous peine d'engendrer un dysfonctionnement des pages web.

2.1 / Le séparateur d'instructions

Les instructions PHP doivent être terminées par un point-virgule, à l'instar du Javascript.

```
instruction_PHP;

echo "Bienvenue dans le cours PHP !";
```

Si une instruction n'est pas terminée dans un script, alors **cette erreur risque de perturber le fonctionnement correct de la page**, voire de déclencher une erreur d'exception.

Si les marqueurs sont placés de part et d'autre de l'instruction, alors il est possible d'omettre le séparateur puisque la balise de fermeture marque implicitement la fin de l'instruction.

```
<?php echo "Instruction correcte" ?>
```

Exemple [voir]

```
<?php
 $i = 10;

 echo "Compte à rebours :";

 while ( $i >= 0 )
 {
  echo "$i ";
  $i--;
 }
?>
```

2.2 / Les commentaires

A l'instar de la plupart des langages, PHP accepte au sein de ses lignes de codes des commentaires permettant de rendre le script facilement compréhensible.

Les commentaires peuvent être insérés dans un script selon plusieurs méthodes différentes.

```
// Commentaire sur une ligne

/* Commentaire sur
   plusieurs lignes */

# Commentaire Unix
```

Exemple

```
<?php
  // Définition d'une constante
  define("MESSAGE", "Attention, vous n'avez pas correctement rempli un champ !");

  echo MESSAGE; # Instruction d'affichage

/* Ces lignes de code permettent simplement d'écrire
dans le document résultant le message suivant :
"Attention, vous n'avez pas correctement rempli un champ !" */
?>
```

3 / L'insertion

Le code PHP peut être directement intégré dans un document HTML ou compris dans un fichier externe qui sera par la suite inséré dans une page web.

3.1 / Insertion des scripts

Les scripts écrits en PHP doivent être intégrés dans une page HTML par le biais d'un balisage spécifique.

```
<? Script PHP... ?>
```

Dans le style des instructions de traîtement XML, les scripts PHP sont encadrés par une balise d'ouverture *<?* et une autre de fermeture *?>*. Cela n'est possible que si l'option d'exécution *short_open_tag* ou l'option de complation *--enable-short-tags.* est activée.

Il est également possible d'**intégrer le mot clé *php* après la première balise** de telle sorte à adopter le style de balisage spécifique à PHP, lequel est évidemment le plus couramment utilisé dans les applications PHP.

```
<?php Script PHP... ?>
```

Par ailleurs, **l'utilisation des marqueurs HTML de scripts est permise** avec l'attribut *language* paramétré pour le langage PHP.

```
<script language="PHP">
  Script PHP...
</script>
```

Enfin, **le code PHP peut être encadré par le balisage réservé à ASP**, si l'option d'exécution *asp_tags* ou l'option de compilation *--enable-asp-tags* est activée.

```
<% Script PHP... %>
```

Exemple [voir]

```
<html>
  <head>
    <script language="PHP">
      $texte = "Le marqueur PHP (&lt;?php...>) demeure le plus courant.";
    </script>
  </head>
  <body>
    <h1>
      <?php echo "$texte " ?>
    </h1>
  </body>
</html>
```

3.2 / Insertion complexe

Les scripts PHP peuvent être insérés à n'importe quel endroit d'une page HTML.

Exemple [voir]

```
<html>
 <body>
  <form method="post" action="formulaire.php">
   <table border="0">
    <tr>
     <td>Nom</td>
     <td><input type="text" name="nom" size="20"></td>
    </tr>
    <tr>
     <td>Adresse</td>
     <td><textarea name="adresse" cols="20" rows="3"></textarea></td>
    </tr>
    <tr>
     <td> </td>
     <td><input type="submit" name="soumettre" value="Envoyer"></td>
    </tr>
   </table>
  </form>
 </body>
</html>
<!-- Fichier de traîtement : formulaire.php -->
<html>
 <body>
  <?php
   if($nom)
   {
    if($adresse)
    {
 ?>
   <table border="0">
    <tr>
     <td>Nom</td>
     <td><?php echo $nom ?></td>
    </tr>
    <tr>
     <td>Adresse</td>
     <td><?php echo $adresse ?></td>
    </tr>
   </table>
  <?php
      }
      else
      {
       echo "<h1>Le champ Adresse n'a pas été rempli !</h1>";
      }
    }
    else
    {
     echo "<h1>Le champ Nom n'a pas été rempli !</h1>";
    }
   ?>
 </body>
</html>
```

3.3 / Les fichiers externes

Les scripts PHP peuvent être contenus dans des fichiers externes afin d'être réutilisés par plusieurs documents HTML différents.

Ainsi, **un mécanisme d'inclusion permet d'insérer puis d'évaluer le contenu PHP** d'un fichier externe dans des pages HTML.

```
<?php include("nom_fichier.inc"); ?>
```

Une autre instruction contribue à **garantir que le fichier externe ne sera inclus et évalué qu'une seule et unique fois** dans le document hôte.

```
<?php include_once("nom_fichier.inc"); ?>
```

L'instruction *require* effectue son propre remplacement par le contenu du fichier externe au sein du document HTML.

```
<?php require("nom_fichier.inc"); ?>
```

A l'instar de l'instruction *include*, ***require* possède son pendant permettant une unique substitution** dans le document hôte.

```
<?php require_once("nom_fichier.inc"); ?>
```

Exemple [voir]

```
<?php
// Fichier : variable.inc
  $operande = 100;
  $diviseur = 5;
?>

<?php
// Fichier : division.inc
  if(empty($diviseur))
  {
    echo "Attention le diviseur ne peut être nul !";
  }
  else
  {
    echo "$operande / $diviseur = ", $operande / $diviseur;
  }
?>

<?php
// Fichier : date.inc
  $date_jour = date("d-m-Y");
  echo("Nous sommes le $date_jour ");
?>
<!-- Fichier : inclusion.php -->
<html>
  <head>
    <?php include_once("variable.inc");?>
  </head>
  <body>
    <h2>
      <?php include("division.inc"); ?>
    </h2>
    <br>
    <h3>
      <?php require("date.inc"); ?>
    <h4>
  </body>
</html>
```

4 / Les types de données

Il n'existe que quatre types de données dans le langage PHP.

Les variables comme dans de nombreux langages web ne possèdent pas de types de données prédéfinis. Seul le contenu de la variable en cours de traîtement définira son type de données. Les variables peuvent donc changer de type à n'importe quelle étape du script.

Type	Description
entier	représente un entier positif compris entre -2 147 483 647 et 2 147 483 647.
flottant	représente un nombre à virgule flottante compris entre -1.78 10^{308} et 1.78 10^{308}.
chaîne de caractères	représente une chaîne de caractères.
booléen	représente une valeur booléenne possédant deux valeurs possibles, soit *true* ou *false*.
tableau	constitue une collection d'éléments de divers types (objets, chaînes de caractères, valeurs numériques et booléennes).
objet	représente des instances de classe opérée avec l'opérateur *new*.
ressource	représente une référence vers une ressource externe, à l'instar d'un fichier avec *fopen()* ou d'une base de données avec *mysql_connect()*.
NULL	représente l'inexistence d'une valeur pour une variable.

Les pseudo-types sont utilisés dans la documentation officielle PHP, pour distinguer les types de données attendus pour les paramètres des fonctions.

Type	Description
mixed	indique que des paramètres sont susceptibles d'accepter plusieurs types de données distincts.
number	indique que des paramètres sont susceptibles de n'accepter que des types numériques.
callback	indique que des paramètres sont des noms de fonctions, à l'instar des fonctions *call_user_func()* et *xml_set_default_handler()*.

5 / Les constantes

Les constantes sont des noms chargés de représenter des valeurs invariables et inaltérables.

```
NOM_CONSTANTE = valeur
```

Conventionnellement, les constantes sont **toujours écrites en majuscules** pour des raisons de clarté du code.

Les constantes acceptent uniquement des valeurs parmi les types suivants :

- chaîne de caractères,
- entier,
- double,
- et booléen.

La création d'une constante est réalisée par l'intermédiaire de la fonction *define*.

```
define(nom_constante, valeur_constante);
```

Une constante est simplement **appelée par son nom avec une casse correcte et sans le signe *$*** dévolu aux variables.

```
echo NOM_CONSTANTE;
```

Les constantes sont **utilisables partout dans le code** puisqu'elles sont par définition, globales.

5.1 / Les constantes prédéfinies

Plusieurs constantes prédéfinies existent dans le langage PHP dans le but de rendre disponible certaines valeurs utiles dans une application web.

Constante	valeur	Description
__FILE__		contient le nom du fichier en cours d'exécution.
__LINE__		contient le numéro de la ligne en cours d'exécution.
PHP_VERSION		contient la chaîne de caractères indiquant la version du PHP en cours d'utilisation.
PHP_OS		contient le nom du système d'exploitation en cours d'utilisation sur le serveur.
TRUE		représente la valeur logique TRUE (vrai).
FALSE		représente la valeur logique FALSE (faux).
E_ERROR	1	représente une erreur impossible à corriger. Elle est différente d'une erreur d'analyse.
E_WARNING	2	représente un message d'alerte provoquée par une erreur qui n'interrompt pas le script.
E_PARSE	4	représente une erreur d'analyse dans le domaine syntaxique dont la correction est impossible.
E_NOTICE	8	représente un avertissement ou une erreur n'ayant pas provoqué un arrêt du script.
E_CORE_ERROR	16	représente un avertissement ou une erreur n'ayant pas provoqué un arrêt du script.
E_CORE_WARNING	32	représente un avertissement ou une erreur n'ayant pas provoqué un arrêt du script.
E_COMPILE_ERROR	64	représente une erreur de compilation provoquant l'interruption du script.
E_COMPILE_WARNING	128	représente un message d'avertissement provenant du compilateur sans interrompre le script.
E_USER_ERROR	256	représente une erreur dûe à l'utilisateur provoquant l'interruption du script.
E_USER_WARNING	512	représente un message d'avertissement dû à l'utilisateur ne provoquant pas l'arrêt du script.
E_USER_NOTICE	1024	représente un message d'avertissement ou une erreur n'ayant pas provoqué l'arrêt du script dû à l'utilisateur.
E_ALL		représente toutes les constantes E_....

La fonction *get_defined_constants* permet de retourner la liste de toutes les constantes prédéfinies et créées par *define* ainsi que leur valeur associée.

```
echo get_defined_constants();
// affiche la liste : constante => valeur
```

6 / Les variables

Les variables dans le langage PHP se distinguent par le caractère *dollar* (*$*) placé devant leur nom.

```
$nom_variable
```

Les noms de variables doivent être composés de caractères valides [a-zA-Z_] et de plus sont sensibles à la casse.

Les variables peuvent **contenir n'importe quel type de donnée** supporté par le langage PHP, soit :

- une chaîne de caractères,
- un entier,
- un nombre réel,
- un booléen,
- un tableau,
- un objet,
- une ressource,
- un null.

L'assignation d'une valeur à une variable s'effectue par l'intermédiaire du signe égal (=).

```
$variable = valeur;
```

```
$i = 0;
$nb = 3.14;
$hex = 0x24;
$chaine = "Un texte court...";
$paragraphe = $paragraphe . $chaine;
$tableau = array(1,2,3,4,5);
```

En ajoutant une esperluette (*&*) devant la variable, il devient possible d'**assigner une référence à une variable**, c'est-à-dire que la variable réceptrice en changeant de valeur affectera celle de la variable référencée.

```
$variable = &$variable_réferencée;
```

```
$prenom = 'Pierre';
$prenom_temp = &$prenom; # $prenom_temp = 'Pierre'
$prenom_temp = 'Jean-$prenom_temp';
// $prenom_temp = $prenom = 'Jean-pierre'
```

6.1 / La portée des variables

La portée des variables diffère selon son emplacement au sein du code PHP dans une page web.

Les variables peuvent avoir une portée globale si elles sont définies en dehors d'une fonction. Dans le cas contraire, elles seront locales.

```
<?php
 $variable_globale = valeur;

 function Fonction()
 {
  $variable_locale = valeur;
 }
?>
```

Les fichiers inclus dans un autre peuvent non seulement, **accéder aux variables globales** définies dans la page hôte, mais également **les modifier**.

```
<?php
 $variable_globale = valeur;
 include "fichier.inc";
?>
<!-- fichier.inc -->
<?php
 $variable_globale = nouvelle_valeur;
?>
```

Les variables globales peuvent être utilisées dans une fonction à condition de déclarer à nouveau les variables dans la fonction avec le mot-clé *global* ou en utilisant le tableau associatif prédéfini *$GLOBALS*.

```
$var_glo = valeur;
function Fonction()
{
 global $var_glo;
 $var_glo++;
}

function Autre_Fonction()
{
 $GLOBALS["var_glo"]++;
}
```

Par ailleurs, **des variables peuvent être déclarées statiques dans une fonction** afin de les utiliser récursivement dans la fonction elle-même.

```
function Fonction()
{
 static $variable = 0;
 if($variable <= 0)
 {
  Fonction();
 }
 $variable++
}
```

Dans cet exemple, si la variable n'était pas statique, sa valeur resterait toujours nulle puisque son initialisation à zéro annulerait son incrémentation.

Exemple [voir]

```php
<?php
 $personne = '';
 $sexe = array('Monsieur','Madame','Mademoiselle');

 function fiche($nom, $prenom)
 {
  global $sexe;
  global $personne;
  $personne = $sexe[0] . ' ' . $prenom . ' ' . $nom;
 }

 $champ1 = 'Lemestre';
 $champ2 = 'Jean-Christophe';

 fiche($champ1, $champ2);

 echo $personne;
?>
```

6.2 / Les variables dynamiques

Les variables dynamiques permettent d'affecter un nom différent à une autre variable.

```
$nom_variable = 'nom_var';

$$nom_variable = valeur;
// équivaut à
$nom_var = valeur;
```

Cette technique facilite, donc, la modification dynamique du nom d'une variable.

Les variables tableaux sont également capables de supporter les noms dynamiques, cependant il est nécessaire de faire attention à la syntaxe à utiliser afin d'éviter toutes équivoques lors d'un appel à *$$variable*. C'est pourquoi, l'adjonction d'accolades rend l'expression plus sûre.

```
$nom_variable = array("val0", "val1", ...,"valN");
${$nom_variable[0]} = valeur;
$val0 = valeur;

$nom_variable = "nom_var";
${$nom_variable}[0] = valeur;
$nom_var[0] = valeur;
```

Les accolades servent aussi à **éviter toutes confusions lors du rendu d'une variable dynamique**.

```
echo "Nom : $nom_variable - Valeur : ${$nom_variable}";
// équivaut à
echo "Nom : $nom_variable - Valeur : $nom_var";
```

6.3 / Le transtypage

Le transtypage permet de convertir le type d'une variable dans un autre type explicite.

```
$variable = (operateur_typage) valeur;
```

Opérateur	Description
(int) (integer)	convertit une variable en un nombre entier.
(real) (double) (float)	convertit une variable en un nombre décimal.
(string)	convertit une variable en une chaîne de caractères.
(array)	convertit une variable en un tableau.
(object)	convertit une variable en un objet.

Une variable peut délivrer son type de donnée en utilisant une fonction PHP spécifique.

```
$chaine_type = gettype($variable);
```

Fonction	Description
gettype()	retourne le type d'une variable (integer, double, string, boolean, NULL, array, object, resource, user function, unknown type).
is_numeric()	indique si la variable est de type numérique.
is_int() is_integer() is_long()	indique si la variable est un entier.
is_double() is_real() is_float()	indique si la variable est un nombre décimal.
is_string()	indique si la variable est un chaîne de caractères.
is_array()	indique si la variable est un tableau.
is_object()	indique si la variable est un objet.
is_bool()	indique si la variable est booléenne.
is_null()	indique si la variable est nulle.
is_resource()	indique si la variable est une ressource.
is_scalar()	indique si la variable est scalaire, si elle contient des entiers, des nombres décimaux, des chaînes de caractères ou des booléens.

Les fonctions *is_ retournent une valeur booléenne**, *true* si la comparaison est vraie, sinon *false*.

```
true | false = is_integer($variable);
```

6.4 / Les variables prédéfinies

De nombreuses variables prédéfinies sont disponibles à partir de la configuration courante de PHP.

Evidemment, le langage PHP pouvant fonctionner sur de multiples plateformes et serveurs web, **les variables peuvent fortement différer d'une installation à une autre**.

C'est pourquoi, avant d'utiliser une de ces variables, il sera nécessaire de **vérifier son existence à l'aide de la fonction *phpinfo***.

Les informations fournies par cette commande peuvent être les options de configuration principales et standards ou encore celles d'extensions comme pour MySQL, ODBC ou XML, des informations sur le serveur, mais aussi les variables d'environnement et les variables PHP prédéfinies, les en-têtes HTTP, ainsi que la licence GNU Public.

```
echo phpinfo([constante]);
```

Les informations indiquées par *phpinfo* peuvent être **sélectionnées par catégorie à l'aide de l'argument *constante***.

Constante	Description
INFO_ALL	affiche toutes les informations.
INFO_CONFIGURATION	affiche les informations de configuration.
INFO_CREDITS	affiche les informations sur les auteurs du module PHP.
INFO_ENVIRONMENT	affiche les variables d'environnement.
INFO_GENERAL	affiche les informations sur la version de PHP.
INFO_LICENSE	affiche la licence GNU Public
INFO_MODULES	affiche les informations sur les modules associés à PHP.
INFO_VARIABLES	affiche les variables PHP prédéfinies.

Voir un exemple de la fonction *phpinfo()* : cliquez ici !

Ces variables peuvent être utilisées n'importe où dans un script PHP à condition de placer devant leur nom le fameux signe dollar (*$*).

```
echo $HTTP_USER_AGENT;

nom_ordinateur = $_ENV["COMPUTERNAME"];

$tableau = explode("/", $PHP_SELF);
end($tableau);
$fichier = current($tableau);
```

6.5 / Les fonctions de variables

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les variables.

Fonction
Description
$nombre_double = **doubleval**($variable);
retourne une valeur double à partir de la variable.
true | false = **empty**($variable);
indique si la variable est vide.
$chaine_type = **gettype**($variable);
retourne une chaîne de caractères représentant le type de la variable (integer, double, string, boolean, NULL, array, object, resource, user function, unknown type).
$tableau = **get_defined_vars**();
retourne une liste de toutes les variables définies.
$chaine = **get_resource_type**($variable);
retourne le type de ressource d'une variable.
$entier = **intval**($variable);
retourne le nombre entier de la variable.
true | false = **is_array**($variable);
indique si la variable est un tableau.
true | false = **is_bool**($variable);
indique si la variable est un booléen.
true | false = **is_double**($variable);
indique si la variable est un nombre de type double.
true | false = **is_float**($variable);
indique si la variable est un nombre à virgule flottante.
true | false = **is_int**($variable);
indique si la variable est un nombre entier.
true | false = **is_integer**($variable);
indique si la variable est un nombre entier.
true | false = **is_long**($variable);
indique si la variable est un nombre entier long.
true | false = **is_null**($variable);
indique si la variable est un NULL.
true | false = **is_numeric**($variable);
indique si la variable est un type numérique
true | false = **is_object**($variable);
indique si la variable est un objet.
true | false = **is_real**($variable);
indique si la variable est un nombre réel.

true | false = **is_resource**($variable);
indique si la variable est une ressource.
true | false = **is_scalar**($variable);
indique si la variable est de type scalaire.
true | false = **is_string**($variable);
indique si la variable est une chaîne de caractères.
true | false = **isset**($variable);
indique si la variable est affectée.
print_r($variable);
affiche les informations lisibles de la variable.
chaîne = **serialize**($variable);
retourne une chaîne de caractères contenant une représentation sérialisée de la variable afin d'être stockée.
settype($variable, $type);
affecte un type à une variable (boolean, integer, double, string, array, object).
chaîne = **strval**($variable);
retourne une chaîne de caractères représentant la valeur de la variable.
$var = **unserialize**($variable);
désérialise une variable contenant une représentation sérialisée afin de créer une autre variable.
unset($variable [, $variableN]);
détruit une ou plusieurs variables séparées par une virgule.
var_dump($variable)
affiche des informations à propos d'une variable.

7 / Les expressions

Une expression peut être n'importe quel ensemble valide de littéraux, de variables, d'opérateurs, d'instructions PHP et d'autres expressions, qui correspond à une valeur simple.

```
$i += 5;

$texte = "L'agneau sortit de l'étable alors
            que le loup embusqué l'attendait.";

$nombre = sizeof($tableau) - 1;

$result = in_array("loup", $tableau);
//retourne true ou false
```

Cette valeur peut être un nombre (42), une chaîne de caractères (olivier), ou encore une valeur logique (*true* ou *false*).

Conceptuellement, il y a deux types d'expressions : **celles qui assignent une valeur à une variable**, et **celles qui ont simplement une valeur**.

Par exemple, l'expression *$i = 0* est une expression qui affecte à *$i* la valeur zéro. Cette expression elle-même correspond à zéro.

La plupart des expressions font appel à divers opérateurs pour effectuer des affectations, des calculs arithmétiques ou encore des concaténations.

Ainsi, à partir d'expressions basiques, il est possible de **construire des expressions extrêmenent complexes** retournant toujours une valeur.

```
echo $i . " - " . $titre
      . "<a href="../$para[$i - 1]/texte.html">lire</a>";
/*bien que contenant une expression complexe, echo n'est
pas une expression puisqu'elle ne retourne aucun résultat.*/
$date = sprintf("%02d/%02d/%04d", $jour, $mois, $annee);
/* L'ensemble est une expression complexe,
y compris sprintf qui retourne un résultat */
```

8 / Les opérateurs

Les expressions permettent d'évaluer, par l'intermédiaire d'opérateurs spécifiques, différentes valeurs numériques littérales.

Le langage PHP possède un jeu complet d'opérateurs permettant de multiples combinaisons d'expressions.

8.1 / Les opérateurs d'affectation

Un opérateur d'affectation assigne la valeur de l'opérande gauche basé sur la valeur de l'opérande droite.

L'opérateur d'affectation de base est le signe d'égalité (=), qui assigne la valeur de son opérande droite à son opérande gauche. C'est-à-dire, *droit = gauche* assigne la valeur de *gauche* à *droit*.

Les autres opérateurs d'affectation sont sténographiés pour des exécutions standard, comme montré dans la table suivante.

Opérateur	Equivalent	Description
$x = $y		$y est affecté à $x
$x += $y	$x = $x + $y	$y est additionné à $x
$x -= $y	$x = $x - $y	$y est soustrait de $x
$x *= $y	$x = $x * $y	$x est multiplié par $y
$x /= $y	$x = $x / $y	$x est divisé par $y
$x %= $y	$x = $x % $y	le reste de $x/$y est affecté à $x

8.2 / Les opérateurs d'incrémentation et de décrémentation

Les opérateurs d'incrémentation et de décrémentation permettent respectivement d'augmenter de un la variable concernée.

Opérateur	Description	Exemple
x **++** *	Cet opérateur unaire permet l'incrémentation de la valeur *x*	i++ //équivaut à i=i+1
x **--** *	Cet opérateur unaire permet la décrémentation de la valeur *x*	i-- //équivaut à i=i-1

* Si l'un de ses opérateurs est placé avant la variable, alors la valeur de la variable sera incrémentée (++) ou décrémentée (--) avant son utilisation. Par exemple pour *i=0*, *i++* donnera 0 et *++i* donnera 1 de même que *i--* donnera 0 et *--i* donnera -1.

8.3 / Les opérateurs arithmétiques

Les opérateurs arithmétiques prennent des valeurs numériques (des littéraux ou des variables) comme leurs opérandes et renvoient une valeur numérique.

Les opérateurs arithmétiques standards sont l'addition (*+*), la soustraction (-), la multiplication (*), et la division (*/*).

Opérateur	Description	Exemple
x **%** y	L'opérateur modulo retourne le reste de la division *x/y*.	20 Mod 3 'retourne 2
x **+** y	L'opérateur permet d'additionner la valeur *x* à la valeur *y*.	5 + 6'retourne 11
x **-** y	L'opérateur permet de soustraire la valeur *y* de la valeur *x*.	8 - 10'retourne -2
x * y	L'opérateur permet de multiplier la valeur *x* par la valeur *y*.	4 * 9 'retourne 36
x / y	L'opérateur permet de diviser la valeur *x* par la valeur *y* en retournant un nombre à virgule flottante.	4 / 16 'retourne 0.25

8.4 / Les opérateurs de comparaisons

Ce type d'opérateur compare ses opérandes et renvoie une valeur logique en fonction du résultat. Si la comparaison est vraie, la valeur logique *true* est retournée sinon *false*.

Les opérandes peuvent être des valeurs numériques ou des chaînes de caractères.

Les chaînes de caractères sont comparées sur la base du standard lexicographique, en utilisant des valeurs d'Unicode.

Opérateur	Description	Exemples
$x == $y	Si la valeur *y* est égale à *x*, l'opérateur retourne *true*. Dans ce cas, si le type de donnée ne correspond pas alors Javascript tentera de convertir les opérandes dans le type approprié afin d'effectuer la comparaison.	if ($choix == 1)
$x != $y	Si la valeur *y* est différente de *x*, l'opérateur retourne *true*	if ($valeur != $prix)
$x <> $y	Si la valeur *y* est différente de *x*, l'opérateur retourne *true*	if ($valeur != $prix)
$x === $y	Si la valeur de *$y* est strictement identique (valeur et type) à *$x*, alors l'opérateur retourne *true*	if ($paragraphe = $texte)
$x !== $y	Si la valeur de *$y* est strictement différente à *$x*, l'opérateur retourne *true*	if ($ref !== "A000000")
$x > $y	Si la valeur de *$y* est supérieure à *$x*, l'opérateur retourne *true*	if ($montant > 1500)
$x >= $y	Si la valeur de *$y* est supérieure ou égale à *$x*, l'opérateur retourne *true*	if ($hab >= $pop)
$x < $y	Si la valeur de *$y* est inférieure à *$x*, l'opérateur retourne *true*	if ($numero < $page)
$x <= $y	Si la valeur de *$y* est inférieure ou égale à *$x*, l'opérateur retourne	if ($fin <= $premier)

8.5 / Les opérateurs au niveau du bit

Au niveau du bit les opérateurs traitent leurs opérandes comme ensemble de 32 bits (des zéros et ceux), plutôt qu'en tant que nombres décimaux, hexadécimaux, ou octals.

Par exemple, le numéro décimal neuf a une représentation binaire de 1001. Au niveau du bit les opérateurs exécutent leurs exécutions sur de telles représentations binaires, mais ils renvoient des valeurs numériques standard de JavaScript.

Opérateur	Usage	Description	Exemple

AND	*$x* **&** *$y*	Chaque position binaire de l'opérande *$x* est comparée avec la position correspondante de l'opérande *$y*. Dans ce cas, si un *1* est trouvé dans l'une ou l'autre des opérandes, alors l'opérateur retourne un *1*, sinon il renvoie un *0*.	1101 & 0111
			//retourne 0101
			0001 & 1000
			//retourne 0000
			1001 & 0001
			//retourne 0001

OR	*$x* \| *$y*	Chaque position binaire de l'opérande *$x* est comparée avec la position correspondante de l'opérande *$y*. Dans ce cas, si un *1* est trouvé dans les deux opérandes soit dans l'une, soit dans l'autre, alors l'opérateur retourne un *1*, sinon il renvoie un *0*.	1101 | 0111
			//retourne 1111
			0001 | 1000
			//retourne 1001
			1001 | 0001
			//retourne 1001

XOR	*$x* **^** *$y*	Cette opérateur *OU EXCLUSIF* fonctionne comme le précédent à la seule exception que le *1* ne doit se trouver que dans l'une ou l'autre des opérandes pour produire le résultat *1* sinon il renvoie un *0*.	1101 ^ 0111
			//retourne 1010
			0001 ^ 1000
			//retourne 1001
			1001 ^ 0001
			//retourne 1000

NOT	~ *$a*	L'opérateur unaire *NOT* retourne un *0* lorsque l'opérande contient un *1* et un *1* pour un *0*.	~1101
			//retourne 0010
			~0001
			//retourne 1110
			~1001
			//retourne 0110

Décalage à gauche	*$x* << *n*	Cet opérateur déplace la représentation binaire de *$x* de *n* bits à gauche. La valeur de ces *n* bits est de zéro.	15 << 2 /* 0000 1111 décalé de 2 bits produit 0011 1100 et donc retourne 60
Décalage à droite avec préservation du signe	*$x* >> *n*	Cet opérateur déplace la représentation binaire de *$x* de *n* bits à droite et conserve le signe.	15 >> 2 /* 0000 1111 décalé de 2 bits produit 0000 0011 et donc retourne 3 */ -15 >> 2 /* 11111 0001 décalé de 2 bits produit 1111 1100 t donc retourne -4 */

8.6 / Les opérateurs logiques

Les opérateurs logiques sont typiquement utilisés pour comparer des expressions ou des nombres et retournent en résultat des valeurs booléennes.

Opérateur	Usage	Description	Exemple
and ou **&&**	$x **and** $y	L'opérateur renvoie *True* si les deux opérandes sont vraies; autrement, il retourne *False*.	($a < 10) and ($b < 10) *Si *$a* et *$b* sont inférieures à 10, l'opérateur retourne *true*.*/
orou ||	$x **or** $y	L'opérateur renvoie *True* si l'une ou l'autre des opérandes est vraie ou si toutes les deux sont fausses, sinon il retourne *False*.	($a >= 1) or ($b == "fin") *Si *$a* est supérieur ou égal à 1 ou/et si *$b* est égal à *fin* alors l'opérateur renvoie *true*, sinon il retourne *false*.*/
xor	$x **xor** $y	L'opérateur renvoie *True* si les des opérandes sont vraies ou fausses, sinon il retourne *False*.	($a >= 1) xor ($b == "fin") *Si *$a* est supérieur ou égal à 1 et si *b* est égal à *fin* ou si les deux expressions sont fausses alors l'opérateur renvoie *true*, sinon il retourne *false*.*/
!	**!** $x	L'opérateur renvoie *false* si son unique opérande peut être convertie en *true*, sinon il retourne *false*.	!($a <= 100) *Si *$a* est inférieur ou égal à 100 alors l'opérateur retourne *false*, sinon il renvoie *true*.*/

8.7 / Les opérateurs de tableaux

Ce type d'opérateur compare ses opérandes et renvoie une valeur logique en fonction du résultat. Si la comparaison est vraie, la valeur logique *true* est retournée sinon *false*.

Les opérandes peuvent être des valeurs numériques ou des chaînes de caractères.

Les chaînes de caractères sont comparées sur la base du standard lexicographique, en utilisant des valeurs d'Unicode.

Opérateur	Description
$x **+** $y	Une union du tableau $x vers $y est opérée..
$x **==** $y	Les tableaux $x et $y contiennent les mêmes paires clé/valeur.
$x **===** $y	Les paires clé/valeur des tableaux $x et $y sont parfaitement identiques, y compris dans leur ordre et leur type.
$x **!=** $y	Les tableaux $x et $y contiennent des paires clé/valeur différentes.
$x **<>** $y	Les tableaux $x et $y contiennent des paires clé/valeur différentes.
$x **!==** $y	Les paires clé/valeur des tableaux $x et $y ne sont pas parfaitement identiques, y compris dans leur ordre et leur type.

```
<?php
$x = array('coursphp' => 'Le langage PHP',
        'coursjava' => 'Le langage Java',
        'coursxsl' => 'Le langage XSL',
        'coursxml' => 'Le langage XML');
$y = array('coursphp' => 'PHP',
        'coursxml' => 'XML',
        'courscss' => 'CSS',
        'coursjava' => 'Java',
        'courshtml' => 'HTML');
//Union +
$z = $x + $y;
print_r($z);
echo '\n';
$z = $y + $x;
print_r($z);

/*Affiche : Array ( [coursphp] => Le langage PHP [coursjava] => Le langage Java [coursxsl]
=> Le langage XSL [coursxml] => Le langage XML [courscss] => CSS [courshtml] => HTML )
Array ( [coursphp] => PHP [coursxml] => XML [courscss] => CSS [coursjava] => Java
[courshtml] => HTML [coursxsl] => Le langage XSL ) */

//Comparaisons
comparer($x, $y);

$y = array('coursphp' => 'Le langage PHP',
        'coursjava' => 'Le langage Java',
        'coursxsl' => 'Le langage XSL',
        'coursxml' => 'Le langage XML');

comparer($x, $y);

$y = array('coursphp' => 'Le langage PHP',
        'coursxsl' => 'Le langage XSL',
        'coursjava' => 'Le langage Java',
        'coursxml' => 'Le langage XML');

comparer($x, $y);

/* Affiche : $x == $y : false $x != $y : true $x === $y : false $x !== $y : true $x == $y : true $x
!= $y : false $x === $y : true $x !== $y : false $x == $y : true $x != $y : false $x === $y : false
$x !== $y : true */

function comparer($x, $y){
```

```
  echo '$x == $y : ' . ($x == $y ? 'true ' : 'false');
  echo ' $x != $y : ' . ($x != $y ? 'true ' : 'false');
  echo ' $x === $y : ' . ($x === $y ? 'true ' : 'false');
  echo ' $x !== $y : ' . ($x !== $y ? 'true ' : 'false');
  echo '\n';
}
?>
```

8.8 / Les opérateurs de concaténations

En plus des opérateurs de comparaison, qui peuvent être utilisés sur des valeurs de chaîne de caractères, l'opérateur d'enchaînement . permet de concaténer deux chaînes de caractères, en retournant une autre qui est le résultat de la concaténation des deux opérandes.

L'opérateur d'affectation .= peut également être utilisé pour enchaîner des chaînes de caractères en boucle.

```
echo 'mon' .' ' . 'programme';
# retourne "mon programme"

$texte = 'Un programme';
$texte_2 = 'Hypertext Preprocessor';
resultat = $texte . ' ' . $texte_2;
# retourne 'Un programme Hypertext Preprocessor'

<?php
  $tableau = array('un','deux','trois');
  $resultat = '';
  for($i = 0; $i <= 3; $i++)
  {
  resultat .= resultat . $i . ' - ' . tableau[$i];
  }
  echo resultat;
?>
```

8.9 / L'opérateur d'exécution

L'opérateur d'exécution, soit deux guillemets obliques entourant une expression (`expression`), invite PHP à exécuter une commande shell.

```
//Affiche le type de fichiers de configuration //Bash de l'utilisateur...
echo `file ~/.bash*`;

/*Affiche : .bash_history: ASCII text .bash_logout: ASCII text .bash_profile: ASCII text */
```

En fait, le fonctionnement de l'opérateur d'exécution correspond à celui de la fonction *shell_exec()*.

```
echo shell_exec('file ~/.bash*');
```

Si le mode de sécurité de PHP (*safe mode*) est activé, alors cet opérateur sera désactivé. L'option de configuration *safe_mode* se trouve dans le fichier de configuration *php.ini*. Par défaut, il est désactivé (*safe_mode = Off*).

8.10 / La priorité des opérateurs

La priorité des opérateurs détermine l'ordre qui leur est appliqué en évaluant une expression. Vous pouvez ignorer la priorité d'opérateur en utilisant des parenthèses.

La table suivante décrit la priorité des opérateurs, du plus haut vers le plus bas, 1 ayant la priorité la plus élevé, et 20 la plus faible.

Priorité	Opérateur	Associativité
1	,	gauche
2	or	gauche
3	xor	gauche
4	and	gauche
5	print	droite
6	= += -= *= /= .= %= &= |= ^= ~= <<=>>=	gauche
7	? :	gauche
8	||	gauche
9	&&	gauche
10	|	gauche
11	^	gauche
12	&	gauche
13	== != ===	non-associative
14	< <= > >=	non-associative
15	<< >>	gauche
16	+ - .	gauche
17	* / %	gauche
18	! ~ ++ -- (int) (double) (string) (array) (object) @	droite
19	[ ]	droite
20	new	non-associative

9 / Les instructions conditionnelles

Les instructions conditionnelles permettent de poser une condition avant d'exécuter un bloc d'instructions.

L'instruction conditionnelle la plus répandue dans les scripts demeure la fameuse structure *if*.

```
if (condition) { bloc d'instructions... }
if ($y != 0) { echo "$x / $y = " . $x / $y; }
```

L'instruction *if* peut être accompagnée de *else*, voire d'une combinaison *else if* afin de formuler une alternative par défaut (*sinon*) ou une ou plusieurs conditions différentes (*sinon si*).

```
if (Première condition) { Premier bloc d'instructions... }
else if (Seconde condition) { Second bloc d'instructions... }
...
else if (Nième condition) { Nième bloc d'instructions... }
else { bloc d'instructions par défaut... }

if ($journee == 'matin') { echo "C'est le matin !"; }
else if ($journee == 'après-midi') { echo "C'est l'après-midi !"; }
else if ($journee == 'soirée') ( echo "C'est la soirée !"; )
else { echo "C'est la nuit !"; }
```

Néanmoins, une telle structure conditionnelle peut être **remplacée par une instruction *switch*** dont le fonctionnement est similaire et en outre la syntaxe est bien plus compacte.

```
switch ($variable)
{
  case valeur_1 :
      Premier bloc d'instructions...
      break;
  case valeur_2 :
      Second bloc d'instructions...
      break;
  ...
  case valeur_N :
      Nième bloc d'instructions...
      break;
  default :
      bloc d'instructions par défaut...
      break;
}

switch ($journee)
{
  case 'matin' :
    echo "C'est le matin !";
    break;
  case 'après-midi' :
    echo "C'est l'après-midi !";
    break;
  case 'soirée' :
    echo "C'est la soirée !";
    break;
  default :
    echo "C'est la nuit !";
    break;
}
```

La variable ***$variable*** **est comparée successivement à chacune des valeurs explicitée par l'instruction *case***.
C'est pourquoi, **l'instruction *break* est nécessaire afin de quitter la structure conditionnelle** après que la condition adéquate eut été trouvée.
Par ailleurs, correspondant à *else*, **l'instruction *default* fournit une alternative par défaut à *switch***.

L'instruction ternaire offre une structure plus compacte sous réserve de ne proposer que

deux possiblités selon une condition spécifiée.

```
$resultat = condition ? première_alternative : seconde_alternative;
```

Si la condition est vraie alors la première alternative est retournée, sinon c'est la seconde.

10 / Les boucles

Les boucles permettent de répéter des opérations jusqu'à ce qu'une ou plusieurs conditions soient réunies.

Le langage PHP possède plusieurs instructions itératives, certaines sont plus orientées vers des opérations spécifiques d'autres sont utilisables plus généralement.

10.1 / Les boucles *for* et *foreach*

La boucle *for* se démarque par sa compacité puisqu'elle contient entre parenthèses trois expressions permettant de l'initialiser (ex : *i = 0;*) de poser une condition à son terme (ex : *i > 10*) et de l'incrémenter ou décrémenter automatiquement (ex : *i++*).

```
for (initialisation; condition; incrémentation)
{
  bloc d'instructions...
}

for($i = 0; $i < 10; $i++)
{
   echo $i.' ';
}
//Affiche : 0 1 2 3 4 5 6 7 8 9
```

Le bloc d'instructions à exécuter dans une boucle *for* peut être **contenu soit à l'intérieur d'accolades**, soit **une instruction de terminaison spéciale *endfor*** indique sa fin.

```
for (initialisation; condition; incrémentation):
  bloc d'instructions...
endfor;

for($i = 0, $j = 10; $i + $j < 10000; $i+=2, $j--):
  echo 'La somme de '.$i.' et '.$j.' est égale à '.($i + $j).'&ltbr/>';
endfor;
/* Affiche :
La somme de 0 et 10 est égale à 10
La somme de 2 et 9 est égale à 11
La somme de 4 et 8 est égale à 12
...
La somme de 19976 et -9978 est égale à 9998
La somme de 19978 et -9979 est égale à 9999*/
```

Littéralement, cette boucle **accomplit des opérations à partir d'un état initial et jusqu'à une certaine condition par pas de tant**.

La boucle *foreach* est utilisée exclusivement pour les tableaux ou toutes autres structures tabulaires.

A chaque itération, l'instruction *foreach* **extrait chaque valeur d'un tableau** en la rendant disponible dans le bloc d'instructions.

```
foreach ($tableau as $valeur)
{
  bloc d'instructions...
}

$tableau = array("January" => "Janvier", "February" => "Février",
                 "March" => "Mars", "April" => "Avril", "May" => "Mai",
                 "June" => "Juin", "July" => "Juillet", "August" => "Août",
                 "September" => "Septembre", "October" => "Octobre",
                 "November" => "Novembre", "December" => "Décembre");
foreach ($tableau as $valeur => $cle)
{
  echo $cle.' est traduit en fançais par '.$valeur.'<br>';
}
```

Littéralement, cette boucle **accomplit des opérations pour chaque valeur d'un tableau**.

Exemple [voir]

```
<?php
 $nb = 0;
 $page = 8;
 $total = 256;

 if ($total % $page == 0) $inc = $total / $page;

 $resultat = '<table><tr><th style="font-size:14pt" colspan="' . $page . '">'
           . 'Tableau de caractères ISO-8859-1</th></tr><tr>';

 for ($i = 0; $i < $page; $i++)
 {
  $limite = $nb + $inc;
  $resultat .= '<td><table border = "1">'
             . '<tr><th>Index</th><th>caractère</th><th>Entité</th></tr>';
  for ($j = $nb; $j <= $limite - 1; $j++)
  {
   $resultat .= '<tr><th>' . $j . '</th><td>' . chr($j)
           . '</td><td>&#' . $j . ';</td></tr>';
  }
  $resultat .= '</table></td>';
  $nb = $limite;
 }

 $resultat .= '</tr></table>';

 echo $resultat;
?>
```

10.2 / Les boucles *while* et *do...while*

La boucle *while* contient essentiellement comme argument une condition d'arrêt toujours évaluée avant chaque itération. L'initialisation et l'incrémentation devront être explicitées dans le code.

```
while (condition)
{
  //bloc d'instructions...
}

i = 0;
while(i < 10)
{
  echo 'valeur de i : '.$i;
  $i++;
}
```

A l'instar de la boucle for, ***while* possède une instruction de fin *endwhile***.

```
while (condition):
  bloc d'instructions...
endwhile;

$fin = true;
$i = 100;
while($fin):
  if($i % 5 && $i <=50) $fin = false;
  $i--;
endwhile;
```

la boucle *do...while* contrairement à *while...endwhile* évalue la condition après chaque itération.

```
do
{
  //bloc d'instructions...
}
while (condition);
```

Littéralement, ces boucles accomplissent des opérations tant qu'une condition est vraie (*faire... tant que...*).

Exemple [voir]

```php
<?php
 $nb = 0;
 $page = 8;
 $total = 256;

 if ($total % $page == 0) $inc = $total / $page;

 $resultat = '<table><tr><th style="font-size:14pt" colspan="' . $page . '">'
        . 'Tableau de caractères ISO-8859-1</th></tr><tr>';

 $i = 0;

 while($i < $page)
 {
  $limite = $nb + $inc;
  $resultat .= '<td><table border = "1">'
             . '<tr><th>Index</th><th>caractère</th><th>Entité</th></tr>';
  $j = $nb;

  do
  {
   $resultat .= '<tr><th>' . $j . '</th><td>' . chr($j)
              . '</td><td>&#' . $j . ';</td></tr>';
   $j++;
  }
  while($j <= $limite - 1);

  $resultat .= '</table></td>';
  $nb = $limite;
  $i++;
 }

 $resultat .= '</tr></table>';

 echo $resultat;
?>
```

10.3 / Les instructions *break* et *continue*

Les instructions *break* et *continue* facilitent respectivement la sortie immédiate et l'arrêt pour l'itération en cours d'une boucle suite à une condition particulière nécessitant une action précise.

```
while(condition)
{
  Bloc d'instructions...
  break;
  // sortie de la boucle avant la prochaîne itération
  Bloc d'instructions...
}

while(condition)
{
  Bloc d'instructions...
  continue;
  /* arrêt de la boucle à l'itération en cours mais passage à l'itération suivante */
  Bloc d'instructions...
}
```

Dans le cas d'une imbrication de plusieurs boucles, les instructions *break* et *continue* peuvent **utiliser un argument indiquant le niveau de la ou des boucles à quitter**.

```
for(initialisation; condition; incrémentation)
  while(condition)
    foreach($tableau as $valeur)
    {
      Bloc d'instructions...
      break n;
      Bloc d'instructions...
      continue m;
    }
  endwhile;
endfor;
```

Si la valeur ***n* est égale 1**, alors la sortie de la boucle foreach se produit.
Si la valeur ***n* est égale 2**, alors la sortie des boucles foreach et while se produit.
Si la valeur ***n* est égale 3**, alors la sortie des trois boucles se produit.

Le fonctionnement est identique pour l'instruction *continue m*.

Exemple [voir]

```php
<?php
 $tableau = array("5" => "Janvier", "6" => "Février",
                  "7" => "Mars", "8" => "Avril", "9" => "Mai",
                  "10" => "Juin", "11" => "Juillet", "12" => "Août",
                  "1" => "Septembre", "2" => "Octobre",
                  "3" => "Novembre", "4" => "Décembre");
 // Exemple break
 while(list($cle, $valeur) = each($tableau)){
  if($valeur == 'Juillet'){
   echo "Fin de l'année scolaire !<br>";
   break;
  }
  echo $valeur."<br>";
 }

 // Exemple continue
 while(list($cle, $valeur) = each($tableau)){
  if($valeur == 'Juillet' || $valeur == 'Août') {
   continue;
  }
   echo "Mois d'une année scolaire : ".$valeur."<br>";
 }

 // Exemple break avec paramètre
 $nouveau = true;
 for(;;){
  while(list($cle, $valeur) = each($tableau)){
   if($nouveau && $cle >= 5) continue;
   $nouveau = false;
   echo "Mois N°".$cle." : ".$valeur."<br>";
   if($cle == 12) break 2;
  }
  reset($tableau);
 }
?>
```

11 / Les tableaux

Le langage PHP dispose de tableaux permettant le regroupement et la gestion de valeurs. Les éléments d'un tableau sont ordonnés selon un couple clé/valeur. La clé peut correspondre soit à **un nombre entier (tableau indicé)**, soit à **une chaîne de caractères (tableau associatif)**. Il est même possible de panacher les types de clé, mais cela peut poser problème lors de l'exploitation du tableau.

En fait, **les tableaux peuvent être assimilés à des collections**. Par exemple en Java, un tableau indicé correspondrait à un vecteur (*Vector*) et un tableau associatif à une table de hachage (*Hashtable*).

Contrairement à d'autres langages, **les tableaux en PHP ne sont pas figés** après leur déclaration. Il est tout à fait possible d'**étendre un tableau**, c'est à dire d'ajouter de nouveaux éléments à ce dernier sans qu'il soit nécessaire de recalculer sa taille.

11.1 / Les tableaux indicés

Les tableaux indicés sont composés de plusieurs éléments dont chacun est repéré par une valeur numérique unique.

La première cellule d'un tableau commence toujours par l'indice zéro (0).

Les tableaux indicés peuvent être **remplis par l'intermédiaire d'affectations de valeurs à une variable suivie d'un indice entre crochets**.

```
$tableau[indice] = valeur;

$jour[3] = "Mercredi";

$note[0] = 20;
```

PHP autorise l'**affectation de valeurs sans qu'il soit nécessaire de spécifier des indices**.

```
$tableau[] = 'v1';
$tableau[] = 'v2';
$tableau[] = 'v3';
$tableau[] = 'v4';
$tableau[] = 'v5';

echo $tableau[0]; //Affiche v1
echo $tableau[1]; //Affiche v2
echo $tableau[2]; //Affiche v3
echo $tableau[3]; //Affiche v4
echo $tableau[4]; //Affiche v5
```

Dans ce cas, les indices sont calculés automatiquement par PHP.

Les valeurs numériques des indices dans un tableau peuvent ne pas être consécutifs.

```
$tableau[10] = 'Webmestre';
$tableau[100] = 'Analyste/Programmer';
$tableau[1000] = 'Chef de projet';
$tableau[10000] = 'Directeur de projet';

print_r($tableau);
/*Affiche Array ( [10] => Webmestre [100] => Analyste/Programmer [1000] => Chef de projet
[10000] => Directeur de projet ) */
```

Il est également possible d'**utiliser la fonction *array* afin de créer et de remplir un tableau**.

```
$tableau = array(valeur0, valeur1,..., valeurN);

$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi",
                    "Jeudi", "Vendredi", "Samedi");

$note = array(20, 15, 12.6, 17, 10, 20, 11, 18, 19);
```

L'accès aux valeurs contenues dans un tableau indicé s'effectue par l'intermédiaire de son indice numérique.

```
$variable = $tableau[indice];

$JJ = $jour[6]; # affecte "Samedi" à $JJ

echo $note[1] + $note[5];
```

Exemple [voir]

```
<html>
  <body>
    <?php
      $i = 0;
      $annee_modif = "";
      $mois_modif = array("Janvier", "Février", "Mars",
                          "Avril", "Mai", "Juin",
                          "Juillet", "Août", "Septembre",
                          "Octobre", "Novembre", "Décembre");
      $mois_anglais = array("January", "February", "March",
                          "April", "May", "June",
                          "July", "August", "September",
                          "October", "November", "December");

      $date_modif = date( "d F Y", getlastmod());

      list($jour, $mois, $annee) = split( '[ ]', $date_modif);

      foreach($mois_anglais as $MM)
      {
        if($MM == $mois)
        {
          $mois = $mois_modif[$i];
        }
        $i++;
      }

      $date_modif = "Dernière modification : $jour $mois $annee";

      echo $date_modif;
    ?>
  </body>
</html>
```

11.2 / Les tableaux associatifs

Les tableaux associatifs permettent de ranger des données par rapport à une valeur quelconque, telle qu'une chaîne de caractères ou un nombre.

```
$tableau["indice"] = valeur;

$jour["Dimanche"] = 7

$jour["Mercredi"] = "Le jour des enfants"
```

Il est également possible d'**utiliser la fonction *array* afin de créer et de remplir un tableau**.

```
$tableau = array(ind0 => val0, ind1 => val1,..., indN = valN);

$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" => 3,
              "Mercredi" => 4, "Jeudi" => 5,
              "Vendredi" => 6, "Samedi" => 7);
```

L'accès aux données des tableaux associatifs s'effectue par l'intermédiaire de la valeur indiciaire.

```
$variable = $tableau["indice"];

$JJ = $jour["Vendredi"]; # affecte 6 à $JJ

echo $jour["Lundi"]; # retourne la valeur 2
```

Exemple [voir]

```
<html>
 <body>
  <?php
   $annee_modif = "";
   $mois_modif = array("January" => "Janvier", "February" => "Février",
                       "March" => "Mars", "April" => "Avril", "May" => "Mai",
                       "June" => "Juin", "July" => "Juillet", "August" => "Août",
                       "September" => "Septembre", "October" => "Octobre",
                       "November" => "Novembre", "December" => "Décembre");

   $date_modif = date( "d F Y", getlastmod());

   list($jour, $mois, $annee) = split( '[ ]', $date_modif);

   foreach($mois_modif as $cMM => $MM)
   {
    if($cMM == $mois)
    {
     $mois = $MM;
    }
   }

   $date_modif = "$jour $mois $annee";

   echo "Dernière modification : " . $date_modif;
  ?>
 </body>
</html>
```

11.3 / Les tableaux multidimensionnels

Dans le langage PHP, il n'existe pas d'outils pour créer directement des tableaux multidimensionnels. Néanmoins, l'imbrication des tableaux étant possible, l'ajout de plusieurs dimensions devient tout à fait réalisable.

```
$tab1 = array(Val0, Val1,..., ValN);
$tab2 = array(Val0, Val1,..., ValN);
// Création d'un tableau à deux dimensions
$tableau = array($tab1, $tab2);

$mois = array("Janvier", "Février", "Mars", "Avril",
              "Mai", "Juin", "Juillet", "Août",
              "Septembre", "Octobre", "Novembre", "Décembre");
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi",
                    "Jeudi", "Vendredi", "Samedi");

$element_date = array($mois, $jour);
```

L'accès aux valeurs contenues dans un tableau indicé s'effectue par l'intermédiaire de l'indice numérique du tableau conteneur suivi de celui du second tableau.

```
$variable = $tableau[indice][indice];

$MM = $element_date[0][0]; # affecte "Janvier" à $MM

echo $element_date[1][5] . " 7 " . $element_date[0][2] . "2002";
// retourne "Jeudi 7 Mars 2002"
```

Bien qu'il soit possible de créer des tableaux multidimensionnels avec des valeurs panachées, il est préférable d'éviter ce genre de structure dont l'exploitation se révélera vite fastidieuse.

```
&tableau = array("chaine", $tab1, Nombre, $tab2);
```

En général, le parcours des éléments d'un tableau s'effectue au moyen de boucles.

Exemple [voir]

```
<html>
 <body>
  <?php
   $i = 0;
   $annee_modif = "";
   $mois_modif = array("Janvier", "Février", "Mars",
                       "Avril", "Mai", "Juin",
                       "Juillet", "Août", "Septembre",
                       "Octobre", "Novembre", "Décembre");
   $mois_anglais = array("January", "February", "March",
                       "April", "May", "June",
                       "July", "August", "September",
                       "October", "November", "December");

   $mois_tab = array($mois_anglais, $mois_modif);

   $date_modif = date( "d F Y", getlastmod());

   list($jour, $mois, $annee) = split( '[ ]', $date_modif);

   foreach($mois_tab as $sMM)
   {
    $i++;
    foreach($sMM as $MM)
    {
     if($MM == $mois)
     {
      $mois = $mois_tab[1][$i];
      break;
     }
    }
   }

   $date_modif = "$jour $mois $annee";

   echo "Dernière modification : $date_modif.";
  ?>
 </body>
</html>
```

11.4 / Le parcours des tableaux

Généralement, le parcours des éléments d'un tableau s'effectue par le truchement d'une boucle *while* et plus spécifiquement *foreach*.

La boucle *while* nécessite l'utilisation de trois fonctions supplémentaires :

- **la fonction *reset*** permettant de réinitialiser le pointeur du tableau,
- **la fonction *list*** contient une liste de variables correspondant aux éléments d'un tableau,
- **la fonction *each*** retourne le couple clé/valeur en cours et pointe l'élément suivant du tableau.

La condition de la boucle *while* consiste à comparaître la liste de variables au couple clé/valeur en cours.

```
$tableau = array(val1, val2, ..., valN);

reset($tableau);
while (list(, $valeur) = each($tableau)){
  echo "Valeur: $valeur";
}

$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi",
                "Jeudi", "Vendredi", "Samedi");

$i = 0;
reset($jour);
while (list(, $JJ) = each ($jour)){
  echo "La cellule n°". $i . " : " . $JJ . "<br>";
  $i++;
}
```

La boucle *foreach* permet de parcourir tous les éléments d'un tableau à partir d'une structure de boucle simple.

```
$tableau = array(val1, val2, ..., valN);

foreach($tableau as $valeur){
  echo "Valeur: $valeur";
}

$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi",
                "Jeudi", "Vendredi", "Samedi");
$i = 0;
foreach($jour as $JJ){
  echo "La cellule n°". $i . " : " . $JJ . "<br>";
  $i++;
}
```

Le parcours d'un tableau associatif est réalisable en ajoutant avant la variable *$valeur*, la clé associée.

```
$tableau = array(clé1 => val1, clé2 => val2, ..., cléN => valN);

foreach($tableau as $clé => $valeur){
  echo "Valeur ($clé): $valeur";
}

$jour = array("Dimanche" => 7, "Lundi" => 1, "Mardi" => 2,
              "Mercredi" => 3, "Jeudi" => 4,
              "Vendredi" => 5, "Samedi" => 6);

foreach($jour as $sJJ => $nJJ){
  echo "Le jour de la semaine n°". $nJJ . " : " . $sJJ . "<br>";
}
```

La boucle *for* parcourt essentiellement les tableaux indicés, en commençant l'itération à un indice précis (ex.: 0), jusqu'à la taille du tableau moins un. Le parcours s'effectue en incrémentant une variable qui permettra de désigner un élément précis du tableau.

```
$fichier = fopen('assurances.csv', 'r');

echo '<table border="1">\n';
while(!feof($fichier)){
  $enregistrement = fgetcsv($fichier, 1024, ';');
  if(substr($enregistrement[0], 0, 1) == '#') continue;
  echo '<tr>\n';
  $taille = sizeof($enregistrement);
  for ($i = 0; $i < $taille; $i++) {
    echo '\t<td>' . $enregistrement[$i] . '</td>\n';
  }
  echo '</tr>\n';
}
echo '</table>";
```

La fonction *list()* affecte une série de variables en fonction des éléments d'un tableau et de leur séquence.

```
$fichier = fopen('assurances.csv', 'r');

echo '<table border="1">\n';
while(!feof($fichier)){
  $enregistrement = fgetcsv($fichier, 1024, ';');
  if(substr($enregistrement[0], 0, 1) == '#') continue;
  list($nom,
      $adresse,
      $telephone,
      $fax) = $enregistrement;
  echo '\t<tr>\n';
  echo '\t\t<td>' . $nom . '</td>\n';
  echo '\t\t<td>' . $adresse . '</td>\n';
  echo '\t\t<td>' . $telephone . '</td>\n';
  echo '\t\t<td>' . $fax . '</td>\n';
  echo '\t</tr>\n';
}
echo '</table>";
```

L'affectation d'un tableau en utilisant une boucle est tout à fait envisageable. Il suffit seulement d'obtenir les valeurs à affecter au tableau, à partir d'un autre tableau, d'une source de données quelconque (fichier csv, base de données, xml, etc.) ou bien calculer à l'aide variables existantes dans le programme.

```
for($i = 0; $i < 10; $i++){
  $puissance[] = pow(2, $i); //Indiçage automatique
  $racine[$i] = sqrt($i); //Indiçage explicite
}
```

11.5 / Les fonctions de tableaux

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les tableaux.

Fonction
Description
$tableau_indice = **array**($val_0, $val_1, ..., $val_N); $tab_associatif = **array**(cle_0 => val_0, cle_1 => val_1, ..., cle_N => val_N);
crée un tableau indicé ou un tableau associatif.
$tableau = **array_count_values**($variable);
retourne un tableau comptant le nombre d'occurrences des valeurs d'un tableau.
$tableau = **array_diff**($var_1, $var_2, ..., $var_N);
retourne dans un tableau contenant les valeurs différentes entre deux ou plusieurs tableaux.
$tableau = **array_filter**($variable, "fonction")
retourne un tableau contenant les enregistrements filtrés d'un tableau à partir d'une fonction.
$tableau = **array_flip**($variable);
intervertit les paires clé/valeur dans un tableau.
$tableau = **array_intersect**($var_1, $var_2, ..., $var_N);
retourne un tableau contenant les enregistrements communs aux tableaux entrés en argument.
$tableau = **array_keys**($variable [, $valeur]);
retourne toutes les clés d'un tableau ou les emplacements d'une valeur dans un tableau.
$tableau = **array_map**($var_1 [, $var_2, ..., $var_N], 'fonction');
applique une fonction à un ou plusieurs tableaux.
$tableau = **array_merge**($var_1, $var_2, ..., $var_N);
enchaîne des tableaux entrés en argument afin d'en retourner un unique.
$tableau = **array_merge_recursive**($var_1, $var_2, ..., $var_N);
enchaîne des tableaux en conservant l'ordre des élément dans le tableau résultant. Dans le cas de de clés communes, les valeurs sont placées dans un tableau.
true | false = **array_multisort**($var, $critere1, $critere2 [, ..., $var_N, $critere1, $critere2])
trie un ou plusieurs tableaux selon un ordre croissant ou décroissant (*SORT_ASC* ou *SORT_DESC*) et selon une comparaison alphabétique, numérique ou de chaîne de caractères (*SORT_REGULAR*, *SORT_NUMERIC* ou *SORT_STRING*).
$tableau = **array_pad**($variable, $taille, $valeur);
recopie tout un tableau en ajustant sa taille à l'argument correspondant et en bourrant d'une valeur spécifiée les éléments vides.
$dernier_element = **array_pop**($variable);
supprime le dernier élément d'un tableau tout en le retournant.
$tableau = **array_push**($variable, $val_1 [, ..., $val_N]);
ajoute une ou plusieurs valeurs à la fin un tableau.
$tableau = **array_reverse**($variable, true | false);
inverse l'ordre des valeurs d'un tableau tout en conservant l'ordre des clés si le second argument est *true*.
valeur = **array_reduce**($variable, "fonction", $valeur_init);
réduit itérativement à une valeur simple, un tableau par rapport à une fonction et éventuellement d'une valeur initiale.
$tableau = **array_rand**($variable, $nombre);
extrait d'une façon aléatoire un certain nombre de valeurs d'un tableau.

$première_valeur = **array_shift**($variable);
supprime la première valeur d'un tableau tout en la retournant.
$tableau = **array_slice**($variable, $position, $nombre);
retourne un tableau contenant un certain nombre d'élément d'un tableau à partir de la position spécifiée.
$tableau = **array_splice**($variable, $position, $nombre, $tab);
supprime un certain nombre d'éléments d'un tableau à partir d'une position et les remplacent éventuellement par le même nombre d'éléments du tableau *$tab*. Si *position* est négatif, les éléments seront supprimés à partir de la fin du tableau. Si *nombre* est négatif, les éléments seront ajoutés à partir de la fin du tableau *$tab*.
valeur = **array_sum**($variable);
retourne la somme des valeurs d'un tableau sous forme d'un nombre entier ou à virgule flottante.
$tableau = **array_unique**($variable);
supprime les doublons d'un tableau.
$tableau = **array_unshift**($variable, $val [, ..., $val_N]);
ajoute des valeurs au début d'un tableau.
$tableau = **array_values**($variable);
retourne les valeurs d'un tableau.
nombre = **array_walk**($variable, "fonction", "argument")
applique une fonction sur chaque membre d'un tableau en utilisant si spécifié un argument utilisateur.
arsort($tableau);
trie un tableau dans un ordre inverse tout en conservant l'ordre des clés.
asort($variable);
trie les valeurs d'un tableau en conservant le couple clé/valeur.
$tableau = **compact**($variable [, ..., $variableN]);
crée un tableau associatif dont les variables sont les clés et leurs contenus, les valeurs.
valeur = **count**($variable);
retourne le nombre le nombre d'éléments d'un tableau.
valeur = **current**($variable);
retourne l'élément courant désigné par le pointeur d'un tableau.
$tableau = **each**($variable);
retourne chacune des paires clé/valeur dans un tableau à quatre éléments (0 => clé, 1 => 'valeur', key => clé, value => 'valeur').
end($variable)
place le pointeur sur le dernier élément du tableau.
nombre = **extract**($variable, $action, $prefixe);
crée des variables dont les noms sont les clés et les contenus sont les valeurs d'un tableau. En cas d'existence des variables, l'argument *action* peut avoir différentes valeurs pour réécrire la variable existante (*EXTR_OVERWRITE*), ne pas réécrire la variable existante (*EXTR_SKIP*), ajouter un préfixe *prefixe* et créer une nouvelle variable (*EXTR_PREFIX_SAME*), ajouter le préfixe *prefixe*, et créer une nouvelle variable (*EXTR_PREFIX_ALL*) ou enfin ajouter un préfixe *prefixe* uniquement sur les variables aux noms invalides ou de type numérique.
true | false = **in_array**(valeur, $variable [, true | false]);

recherche la valeur spécifiée dans un tableau en vérifiant en même temps la concordance du type si *true* est indiqué en argument.

clé | false = **array_search**(valeur, $variable [, true | false]);

retourne la clé associée en cas de réussite de la recherche de la valeur spécifiée dans un tableau en vérifiant en même temps la concordance du type, si *true* est indiqué en argument.

clé = **key**($variable);

retourne la clé en cours dans le tableau.

nombre = **krsort**($variable);

trie les clés d'un tableau dans l'ordre inverse en conservant les paires clé/valeur.

nombre = **ksort**($tableau);

trie les clés d'un tableau en conservant les paires clé/valeur.

list($variable , ..., $variableN);

affecte une série de variables en un seule opération.

natsort($variable);

trie un tableau dans l'ordre naturel en respectant la casse.

natcasesort($tableau);

trie un tableau dans l'ordre naturel sans respecter la casse.

valeur | false = **next**($variable);

avance le pointeur au prochain élément et retourne la valeur correspondante.

valeur | false = **pos**($variable);

retourne la valeur de l'élément courant désigné par le pointeur.

valeur | false = **prev**($variable);

déplace le pointeur sur l'élément précédent et retourne la valeur.

$tableau = **range**($nb_max, $nb_min);

retourne un intervalle de tous les éléments de type entier compris entre un minimum et un maximum sous forme d'un tableau.

valeur = **reset**($variable);

déplace le pointeur sur le premier élément d'un tableau et retourne sa valeur.

rsort($variable);

trie un tableau dans un ordre décroissant.

shuffle($variable);

mélange au hasard tous les éléments d'un tableau après un appel de la fonction *range*.

nombre = **sizeof**($variable);

retourne le nombre d'éléments d'un tableau.

sort($variable);

trie un tableau dans un ordre croissant.

uasort($variable, fonction);

trie les éléments d'un tableau par rapport à une fonction.

uksort($variable, fonction);

trie les clés d'un tableau par rapport à une fonction.

usort($variable, fonction);

trie les valeurs d'un tableau par rapport à une fonction.

12 / Les chaînes de caractères

Les chaînes de caractères peuvent être constituées de n'importe quel caractère alphanumérique et de ponctuation, y compris les caractères spéciaux.

```
\tLa nouvelle monnaie unique, l' €uro, est enfin là...\n\r
```

Une chaîne de caractères **doit être toujours entourée par des guillemets** simples (')ou doubles (").

```
"Ceci est une chaîne de caractères valide."
'Ceci est une chaîne de caractères valide.'
"Ceci est une chaîne de caractères invalide.'
```

Il existe **une différence entre une chaîne de caractères entourée par des guillemets simples et doubles**. En effet, les **guillemets doubles** (*"..."*) indique à l'interpréteur PHP d'**analyser la chaîne de caractères** afin de vérifier si cette dernière contient des variables.

```
<?php
$var = 'bonjour';
echo "$var tout le monde !";
//Affiche : bonjour tout le monde !
?>
```

Tandis que des **guillemets simples** assure que la chaîne de caractères doit être **traitée comme une simple séquence de caractères**.

```
<?php
$var = 'bonjour';
echo '$var tout le monde !';
//Affiche : $var tout le monde !
?>
```

Ainsi, **afin d'optimiser l'exécution du code, il est préférable d'utiliser les guillemets simples** lorsqu'une chaîne de caractères ne contient pas de variables.

Des caractères spéciaux à insérer directement dans le texte, permettent de créer directement certains effets comme des césures de lignes.

Car	Code ASCII	Code hex	Description
\car			échappe un caractère spécifique.
" "	32	0x20	un espace simple.
\t	9	0x09	tabulation horizontale
\n	13	0x0D	nouvelle ligne
\r	10	0x0A	retour à chariot
\0	0	0x00	caractère NUL
\v	11	0x0B	tabulation verticale

Le fait de **placer le caractère d'échappement anti-slash (\)** devant un guillemet évite d'une part que le moteur de script interpréte le caractère et d'autre part l'insére dans le texte.

```
echo "Il l'avait poutant \"prévenu maintes fois\" !"

echo 'Il l\'avait poutant "prévenu maintes fois" !'
/* Dans les deux cas, l'affichage est :
Il l'avait pourtant "prévenu maintes fois" */
```

Il existe de nombreuses instructions permettant de travailler les chaînes de caractères. Ainsi, des chaînes de caractères peuvent être concaténer via l'opérateur point (.). Des recherches, des comparaisons, des extractions, des remplacements, des suppressions ou des ajouts peuvent être effectuer par l'intermédiaire d'une vaste palette de fonctions PHP.

```
//recherche le texte dans le paragraphe
$txt = stristr($para, $texte);

//compare le texte au paragraphe
$nombre = strcasecmp($para, $texte);

//remplace l'ancien texte par le nouveau dans le paragraphe
$para_temp = str_replace($texte_nouveau, $texte_ancien, $para);
```

12.1 / Les fonctions de chaînes de caractères

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les chaines de caractères.

Fonction
Description
$chaine_result = **addCSlashes**($chaine, $liste_caracteres);
ajoute des slashs dans une chaîne.
$chaine_result = **addSlashes**($chaine);
ajoute un slash devant tous les caractères spéciaux.
$chaine_result = **bin2hex**($chaine);
convertit une valeur binaire en hexadécimale
$chaine_result = **chop**($chaine);
supprime les espaces blancs en fin de chaîne.
$caractere = **chr**($nombre);
retourne un caractère en mode ASCII.
$chaine_result = **chunk_split**($chaine, $nb, $caractere);
retourne un chaîne de caractères composée de sous-chaînes de *chaîne* d'un certain nombre de caractères et séparées par le caractère spécifiée.
$chaine_result = **convert_cyr_string**($chaine, $alphabet, $alphabet2);
convertit la chaîne d'un alphabet cyrillique vers un autre. Les valeurs posibles des deux derniers paramètres sont : **k** - koi8-r, **w** - windows-1251, **i** - iso8859-5, **a** - x-cp866, **d** - x-cp866, **m** - x-mac-cyrillic.
$tableau | $chaine = **count_chars**($chaine, $nb);
retourne des informations sur les caractères utilisés dans une chaîne. L'argument *nb* permet de générer soit un tableau avec l'octet comme clé, et la fréquence comme valeur (*0*) ou seules les fréquences non nulles sont listées (*1*) ou seules les fréquences nulles sont listées (*2*), soit chaîne de caractères contenant tous les octets utilisés (*3*) ou tous les octets non utilisés (*4*).
$nombre = **crc32**($chaine);
calcule le polynôme crc32 d'une chaîne.
$chaine_result = **crypt**($chaine [, $chaine_code])
code une chaîne avec une base de codage.
echo $expression_chaîne;
affiche à l'écran une ou plusieurs chaînes de caractères.
$tableau = **explode**($delimiteur, $chaine);
scinde une chaîne de caractères en fragments à l'aide d'un délimiteur et retourne un tableau.
$tableau = **get_html_translation_table**(nombre);
retourne la table de traduction des balises et des entités HTML.
$tableau = **get_meta_tags**($chaine, $nombre);
extrait toutes les balises meta d'un fichier dans un tableau associatif.
$chaine_result = **hebrev**($chaine, $nombre);
convertit un texte hébreux logique en texte visuel avec un certain nombre de caractères par ligne.
$chaine_result = **hebrevc**($chaine, $nombre);
convertit un texte hébreux logique en texte visuel avec les nouvelles lignes de conversion.
$chaine_result = **htmlentities**($chaine, $mode_guillemet);
convertit tous les caractères spéciaux en entité HTML avec un argument pour l'affichage des guillemets pouvant être *ENT_COMPAT* (conversion uniquement des guillemets doubles), *ENT_QUOTES* (conversion de tous les guillemets) et *ENT_NOQUOTES* (aucune conversion).

$chaine_result = **htmlspecialchars**($chaine, $mode_guillemet);
convertit tous les caractères spéciaux en entité HTML. Le traîtement des guillemets est identique à celui de la fontion *htmlentities*.
$chaine_result = **implode**($delimiteur, $tableau);
concatène tous les éléments d'un tableau dans une chaîne séparés par une chaîne de caractères délimitrice.
$chaine_result = **join**($delimiteur, $tableau);
concatène tous les éléments d'un tableau dans une chaîne séparés par une chaîne de caractères délimitrice.
$nombre = **levenshtein**($chaine_1, $chaine_2);
retourne la distance *Levenshtein* entre deux chaînes de caractères.
$tableau = **localeconv**(void);
retourne un tableau contenant le formatage numérique et monétaire.
$chaine_result = **ltrim**($chaine);
supprime les espaces blancs en début de chaîne de caractères.
$chaine_result = **md5**($chaine);
code la chaîne de caractères avec la méthode md5.
$chaine_result = **metaphone**($chaine);
retourne la clé phonétique d'une chaîne de caractères.
$chaine_result = **nl2br**($chaine);
convertit les sauts de ligne dans une chaîne de caractères en balises HTML * *.
$nombre = **ord**($chaine);
retourne la valeur ASCII du premier caractère de la chaîne.
parse_str($chaine, $tableau);
analyse une chaîne de caractères contenant des paires "*nom=valeur*" et en extrait des variables et leur valeur.
print($expression_chaîne);
affiche à l'écran une chaîne de caractères.
$nombre = **printf**($format, $variable, ..., $variableN);
affiche à l'écran une ou plusieurs chaînes de caractères formatée.
$chaine_result = **quoted_printable_decode**($chaine);
décode une chaîne de caractères codée en 8 bits.
$chaine_result = **quoteMeta**($chaine);
ajoute un antislash devant tous les caractères méta.
$chaine_result = **rtrim**($chaine);
supprime les espaces blancs en fin de chaîne de caractères.
$tableau = **sscanf**($chaine, format, &$var, ..., &$varN);
analyse les valeurs contenues dans une chaîne de caractères en fonction d'un format et les passe aux arguments si spécifiés, sinon les retourne dans un tableau.
$chaine_result = **setlocale**(fonction, $code_local);
définit les paramètres locaux. L'argument fonction peut être *LC_ALL* pour toutes les fonctions, *LC_COLLATE* pour les comparaisons de chaînes, *LC_CTYPE* pour la classification de caractères et les conversions, *LC_MONETARY* pour les formats monétaires, *LC_NUMERIC* pour les séparateurs décimaux et *LC_TIME* pour le format des dates et heures date.

$nombre = **similar_text**($chaine_1, $chaine_2 [, $pourcentage]);
retourne le nombre de caractères identiques entre deux chaînes de caractères.
$chaine_result = **soundex**($chaine);
retourne la valeur *soundex* (prononciation des mots) d'une chaîne de caractères.
$chaine_result = **sprintf**($format, $variable, ..., $variableN);
retourne une chaîne de caractères formatée.
$nombre = **strncasecmp**($chaine_1, $chaine_2, $nb_limite);
compare en binaire deux chaînes de caractères avec une longueur limite.
$nombre= **strcasecmp**($chaine_1, $chaine_2);
compare en binaire des chaînes de caractères insensiblement à la casse.
$chaine_result = **strchr**($chaine, $occurrence);
retourne une chaîne à partir de la première occurrence d'un caractère d'une chaîne spécifiée.
$nombre = **strcmp**($chaine_1, $chaine_2);
compare en binaire deux chaînes de caractères et retourne un nombre négatif si la première chaîne et plus petite que la seconde, un positif dans le cas contraire ou un nul en cas d'égalité.
$nombre = **strcoll**($chaine_1, $chaine_2);
compare deux chaînes de caractères localisées et retourne un nombre négatif si la première chaîne et plus petite que la seconde, un positif dans le cas contraire ou un nul en cas d'égalité.
$nombre = **strcspn**($chaine_1, $chaine_2);
retourne la longueur du premier fragment de la chaîne de caractères qui ne contient aucun caractère de la seconde chaîne.
$chaine_result = **strip_tags**($chaine [, $balise]);
supprime les balises HTML et PHP d'une chaîne de caractères hormis les balises à conserver.
$chaine_result = **stripCSlashes**($chaine);
retourne une chaîne sans les caractères d'échappement à partir d'une chaîne de caractères passée par *addcslashes*.
$chaine_result = **stripSlashes**($chaine);
supprime les slashs ajoutés par la fonction *addslashes*.
$chaine_result = **stristr**($chaine_1, $chaine_2);
recherche la seconde chaîne de caractères dans la première et retourne les caractères suivants.
$nombre = **strlen**($chaine);
retourne la longueur de la chaîne.
$nombre = **strnatcmp**($chaine_1, $chaine_2);
compare avec une sensibilité à la casse, deux chaînes de caractères dans l'ordre naturel et retourne un nombre négatif si la première chaîne et plus petite que la seconde, un positif dans le cas contraire ou un nul en cas d'égalité.
$nombre = **strnatcasecmp**($chaine_1, $chaine_2);
compare insensiblement à la casse, deux chaînes de caractères dans l'ordre naturel et retourne un nombre négatif si la première chaîne et plus petite que la seconde, un positif dans le cas contraire ou un nul en cas d'égalité.
$nombre = **strncmp**($chaine_1, $chaine_2, $nombre);
compare en binaire deux chaînes de caractères sur un certain nombre de caractères et retourne un nombre négatif si la première chaîne et plus petite que la seconde, un positif dans le cas contraire ou un nul en cas d'égalité.

$chaine_result = **str_pad**($chaine_1, nb, $chaine_2, $emplacement);
concatène la seconde chaîne de caractères à la seconde selon l'emplacement spécifié, en l'occurrence soit *STR_PAD_RIGHT* (à droite), *STR_PAD_LEFT* (à gauche) ou *STR_PAD_BOTH* (des deux côtés).
$nombre = **strpos**($chaine, $caractere [, $position_recherche]);
retourne la position de la première occurrence d'un caractère dans une chaîne à partir de la position spécifiée.
$chaine_result = **strrchr**($chaine, $caractere);
recherche la partie terminale d'une chaîne à partir d'un caractère spécifié.
$chaine_result = **str_repeat**($chaine, $nombre);
retourne une chaîne de caractères concaténéé sur elle-même un certain nombre de fois.
$chaine_result = **strrev**($chaine);
inverse l'ordre des caractères d'une chaîne.
$nombre = **strrpos**($chaine, $caractere);
retourne la position de la dernière occurrence d'un caractère dans une chaîne.
$nombre = **strspn**($chaine_1, $chaine_2);
retourne la longueur du premier fragment qui correspond à la seconde chaîne.
$chaine_result = **strstr**($chaine, $caractère);
retourne la sous-chaîne à partir de la première occurrence du caractère dans la chaîne de caractères spécifiée.
$chaine_result = **strtok**($chaine, $delimiteur);
découpe une chaîne de caractères selon un délimiteur.
$chaine_result = **strtolower**($chaine);
transforme tous les caractères d'une chaîne en minuscules.
$chaine_result = **strtoupper**($chaine);
transforme tous les caractères d'une chaîne en majuscules.
$chaine_result = **str_replace**($recherche, $remplacement, $chaine);
remplace toutes les occurrences de la première chaîne par la seconde dans la dernière chaîne de caractères passée en argument.
$chaine_result = **strtr**($chaine, $recherche);
retourne une sous-chaîne du premier argument, allant de la première occurrence du second argument jusqu'à la fin de la chaîne.
$chaine_result = **substr**($chaine, $position, $longueur);
retourne un fragment de la chaîne de la chaîne de caractères à partir de la position spécifée et jusqu'à une certaine longueur.
$nombre = **substr_count**($chaine, $sous-chaîne);
retourne le nombre de sous-chaînes trouvées dans la chaîne de caractères.
$chaine_result = **substr_replace**($chaine, $chaine_remplacement, $position[, $longueur]);
remplace les occurences de *$chaine_remplacement* dans une fragment de la chaîne de caractères, délimité par les positions *$position* et la *$position + $longueur*.
$chaine_result = **trim**($chaine);
supprime les espaces blancs au début et à la fin d'une chaîne de caractères.
$chaine_result = **ucfirst**($chaine);
retourne la chaîne de caractères avec son premier caractère en majuscule.

$chaine_result = **ucwords**($chaine);
retourne la chaîne de caractères avec chaque premier caractère de mot en majuscule.
$chaine_result = **wordwrap**($chaine [, $nombre, $cesure, $coupure])
ajoute une césure à une chaîne tous les n (*bombre*) caractères. Si l'argument *coupure* vaut 1, alors les mots ne seront pas découpés.

12.2 / La chaîne de format

La chaîne de format est utilisée par certaines fonctions (*sscanf*, *printf*, *sprintf*) en tant qu'argument afin de convertir précisément une chaîne de caractères par rapport à un masque.

```
sprintf($format, $texte);

$format = "%[-]['{caractère}{taille}][nombre].[précision]type";
// justifie une chaîne de caractères à droite
$format = "%-'<30s";

// formate une date 25/12/2001
$format= "%02d/%02d/%04d";
```

Les chaînes de format peuvent être constituées de plusieurs parties comme des chaînes de caractères associées à des expressions de formatage précédées du signe de pourcentage (%).

```
$format = "Le %02d/%02d/%04d, à %s, monsieur %s %s déclare...";
```

Les expressions de formatage comprennent un signe pourcentage suivi d'un ou plusieurs arguments optionnels et du type de donnée.

Un argument optionnel d'alignement précise la justification de la chaîne résultante, à gauche (par défaut) ou à droite. Le tiret (-) permet de justifier la chaîne de caractères à droite.

```
// aligne la chaîne à droite '.............chaîne'
$format = "%-'.20s";

// aligne la chaîne à gauche 'chaine.............'
$format = "%'.20s";
```

Une option de remplissage spécifie quel caractère sera utilisé pour complèter une chaîne de caractère jusqu'à une taille définie.

Le caractère de remplissage peut être :

- soit un **espace simple** (valeur par défaut),
- soit le **caractère zéro** (0),
- soit une **valeur définie** par l'intermédiaire d'un guillemet simple (') précédant la définition.

```
<?php
  $taille_champ = 30;
  $caractere = "_";

  // $format = "%'_30s";
  $titre = "PHP 4 et MySQL";
  $champ = sprintf ("%'{$taille_champ}{$caractere}s", $livre);
  echo $champ . "<br>";

  $livre = "PHP Précis et concis";
  $champ =  sprintf ("%'{$taille_champ}{$caractere}s", $livre);
  echo $champ . "<br>";
?>
```

L'argument optionnel *nombre* indique le nombre minimum de caractères à afficher dans la chaîne résultante. Pour des chiffres, ce paramètre précisera le remplissage à appliquer.

```
$chiffre = 2;
$format = "%d";
echo printf($format, $chiffre);
// affiche 2

$format = "%04d";
// affiche 0002
```

L'argument optionnel de précision indique le nombre de chiffres utilisé pour afficher un nombre à virgule flottante. Ce pramètre doit être séparé du paramètre *nombre* par un point (.).

```
$chiffre = 1520.3697;
$format = "%01.2f";

echo printf($format, $chiffre);
// affiche 1520.36

$format = "%09.3f";
// affiche 000001520.369
```

L'argument *type* est obligatoire et spécifie le type de données de la valeur sujette au formatage.

```
$format = "%type";

// chaîne de format pour les chaîne de caractères.
$format = "%s";

// chaîne de format pour les nombres à virgule flottante.
$format = "%12.6f";
```

Format	Description
%	le signe pourcentage ne nécessite aucun argument.
b	la valeur attendue est un entier représenté comme un nombre binaire.
c	la valeur attendue est un entier représenté comme un nombre ASCII.
d	la valeur attendue est un entier représenté comme un nombre décimal.
u	la valeur attendue est un entier représenté comme un nombre décimal non signé.
f	la valeur attendue est un double représenté comme un nombre à virgule flottante.
o	la valeur attendue est un entier représenté comme un nombre octal.
s	la valeur est une chaîne de caractères représentée comme telle.
x	la valeur attendue est un entier représenté en minuscules comme un nombre hexadécimal.
X	la valeur attendue est un entier représenté en majuscules comme un nombre hexadécimal.

Une séquence de caractères spécifique ('*nombre*\$') permet de numéroter les expressions de formatage par rapport aux variables passées en argument dans la fonction.

```
<?php
 $numero = 12;
 $titre = "Les fonctions ODBC";

 $format = "Chapître N° %d - %s";
 printf($format, $numero, $titre);
 // retourne 'Chapître N° 12 - Les fonctions ODBC'

 $format = "¤ %-'.80s %02d";
 /* passage d'arguments erroné, le type de donnée ne correspondant pas. */

 $format = "¤ %2\$-'.80s %1\$02d";
 // retourne 'Les fonctions ODBC.....12'
?>
```

12.3 / Les fonctions de caractères

Le langage PHP dispose de nombreuses fonctions permettant de vérifier les types de caractères.

Fonction
Description
true | false = **ctype_alnum**($caractere);
vérifie si un caractère est alpha-numérique.
true | false = **ctype_alpha**($caractere);
vérifie si un caractère est alphabétique.
true | false = **ctype_cntrl**($caractere);
vérifie si un caractère est un caractère de contrôle.
true | false = **ctype_digit**($caractere);
vérifie si un caractère est un nombre.
true | false = **ctype_lower**($caractere);
vérifie si un caractère est en casse minuscule.
true | false = **ctype_graph**($caractere);
vérifie si un caractère est imprimable hormis le caractère d'espace blanc " ".
true | false= **ctype_print**($caractere);
vérifie si un caractère est imprimable.
true | false = **ctype_punct**($caractere);
vérifie si un caractère est imprimable sans être ni un espace, ni un caractère alpha-numérique.
true | false = **ctype_space**($caractere);
vérifie si un caractère est caractère blanc (espace, tabulation...).
true | false = **ctype_upper**($caractere);
vérifie si un caractère est en majuscule.
true | false = **ctype_xdigit**($caractere);
vérifie si un caractère représente un nombre héxadécimal.

13 / Les expressions régulières

Les expressions régulières sont des modèles utilisés pour vérifier des combinaisons spécifiques de caractère dans les chaînes.

Plusieurs fonctions et instructions utilisent de tels modèles afin d'accomplir diverses opérations sur des chaînes de caractères.

```
$tableau = split("\ws", chaîne [, nombre]);
```

13.1 / Les symboles de délimitation

Les expressions régulières permettent d'effectuer des manipulations complexes de chaînes de caractères en utilisant des modèles (*pattern*) répondant à des règles de syntaxes précises.

Tout d'abord, **plusieurs symboles spéciaux permettent de délimiter précisément un modèle** :

Symbole	Description
^	indiquant le début d'une chaîne de caractères,
$	indiquant la fin d'une chaîne de caractères.
\b	indiquant une limite de mot d'une chaîne de caractères, soit la position entre un mot (*\w*) et un espace ou un caractère de ponctuation (*\W*).
\B	indiquant ce qui n'est pas une limite de mot dans une chaîne de caractères.
\d	indique un chiffre comprenant les caractères suivants [0-9].
\D	indique ce qui n'est pas un chiffre [^0-9].
\n	indique un nouvelle ligne.
\r	indique un retour charriot.
\s	indique un espace blanc comme \t, \n, \r, ou \f.
\S	indique ce qui n'est pas un espace blanc (\t|\n|\r|\f).
\t	indique une tabulation verticale.
\w	indique un mot comprenant le caractères suivants [0-9a-zA-Z_].
\W	indique ce qui n'est pas un mot [^0-9a-zA-Z_]

```
$chaine = "Le loup est dans la bergerie.";

// recherche 'Le' en début de chaîne
$regexp = "^Le";

// recherche 'rie.' en fin de chaîne
$regexp = "rie\.$";

// recherche 'ber' au début d'un mot ex: 'bergerie'
$regexp = "\bber";
$regexp = "\sber";
$regexp = "\Wber";

// recherche 'rie' à la fin d'un mot ex: 'bergerie'
$regexp = "rie\b";
$regexp = "rie\W";

/* recherche 'rie' à la fin d'un mot mais ne le trouve pas
car 'bergerie' se termine par un point et pas par un espace blanc */
$regexp = "rie\s";

// recherche 'but' à la fin d'un mot ex: 'début'
$regexp = "but\s"

// recherche 'ger' dans un mot ex: 'bergerrie'
$regexp = "\Bger";
$regexp = "\Sger";
$regexp = "\wger";
```

Si une expression régulière ne contient aucun des deux symboles '^' ou '*$*', alors le domaine de recherche peut être située n'importe où dans la chaîne de caractères.

```
$regexp = "dans";

$regexp = "loup";
```

13.2 / Nombre d'occurrences

Trois symboles spéciaux, soit les signes : étoile (*), plus (+) et point d'interrogation (?), fournissent des solutions pour **signifier un certain nombre d'occurrences possible d'un caractère** dans une expression régulière.

Indicateur	Description
*	indique zéro ou plusieurs occurrences du caractère précédent.
+	indique une ou plusieurs occurrences du caractère précédent.
?	indique zéro ou une occurrence du caractère précédent.

```
// recherche '21', 210', '2100', '21000', etc..
$regexp = "210*";

// recherche 'cré', 'créé'
$regexp = "cré+";

// recherche 'pa' ou 'pas'.
$regexp = "pas$";

// recherche 'occurrence', 'occurrences' mais aussi malgré
une orthographe erronée 'ocurence' ou 'ocurences', etc..
$regexp = "oc+ur+ences?";
```

Afin de **limiter ou de préciser le nombre d'occurrences**, il suffit d'utiliser ce nombre à l'intérieur d'accolades placées avant le caractère.

```
// recherche 'créé'
$regexp = 'cré{2}';

// recherche de '1000' jusqu'à '1000000000'.
$regexp = '10{3, 9}';

// recherche 'as' ou 'ass'.
$regexp = 'as{1, 2}';
```

13.3 / Regroupement et alternative

Il est possible de regrouper des caractères dans le but de créer soit une séquence à laquelle sera appliquée un symbole spéciale d'occurençage soit une structure alternative.

Pour **appliquer les différents symboles d'occurrences à une séquence de caractères**, il faut placer cette dernière entre des parenthèses.

```
$regexp = "1( 000)*";

$regexp = "1( 000){1, 3}";

$regexp = "(-/-)+";
```

A l'intérieur des parenthèses, **un symbole barre verticale (|) entre deux séquences de caractères pose une opération logique *OU***.

```
// recherche 'assu' ou 'accu'.
$regexp = "a(s{2}|c{2})u";

// recherche 'A' ou 'B' au début de la chaîne de caractères.
$regexp = "^(A|B)";
```

13.4 / Mise en correspondance des caractères

Le point (.) dans une expression régulière représente n'importe quelle caractère.

```
/* recherche toute séquence commençant par un 'a'
et terminant par un 'e' comme 'ate' ou 'aie'. */
$regexp = "a.e";

// recherche par exemple 'occurrence' ou 'occurrences'.
$regexp = "oc{2}.*r{2}.*es?";
```

Les crochets ([])dans une expression régulière servent à mettre en place une structure alternative ou chacun des caractères qu'elle comprend, pourra se trouver dans une chaîne de caractères. **Un signe moins (-) entre deux caractères désigne une plage de lettres ou de chiffres.**

```
// correspond à 'A | B'.
$regexp = "[AB]";

// recherche n'importe quelle lettre comprise entre a, z, A et Z.
$regexp = "[a-zA-Z]";

/* recherche toutes chaînes de caractères
comprenant des caractères alphanumériques spécifiés. */
$regexp = "^[a-zA-Zàäâéèëêîïôôùüûç0-9]+";
```

Il est également possible d'**exclure des caractères par l'intermédiaire de l'accent circonflexe (^) à l'intérieur des crochets**.

```
// exclut les chiffres de la recherche.
&regexp = "[^0-9]";

/* exclut les lettres alphabétiques minuscules
et les chiffres en fin de chaîne. */
$regexp = "[^a-b0-9]$";
```

Le caractère d'échappement anti-slash (\) permet d'inclure des caractères spéciaux dans une expression régulière.

```
// recherche le signe euro, dollar ou livre.
$regexp = "(€|\$|£)";

/* recherche un point d'interrogation
à la fin d'une chaîne de caractères. */
$regexp = "\?$";

// recherche le crochet ouvrant
$regexp = "\["
```

Les crochets évitent l'utilisation du caractère d'échappement pour les caractères spéciaux.

```
// recherche les caractères indiqués.
$regexp = "[(){}.*+?$\]";
```

13.5 / Les fonctions d'expressions régulières

Le langages PHP dispose de plusieurs fonctions permettant de travailler sur les expressions régulières (*regexp*).

Fonction
Description
$nombre = **ereg**($modele, $chaine [, $tableau]);
recherche un modèle dans une chaîne de caractères et enregistre les résultats dans le tableau.
$chaine_result = **ereg_replace**($modele, $chaine_remplacement, $chaine);
remplace le fragment de chaîne correspondant au modèle par la chaîne de substitution.
$nombre = **eregi**($modele, $chaine [, $tableau]);
recherche en ignorant la casse, un modèle dans la chaîne de caractères puis enregistre les résultats dans le tableau.
$chaine_result = **eregi_replace**($modele, $chaine_remplacement, $chaine);
remplace le fragment de chaîne correspondant au modèle par la chaîne de substitution en ignorant la casse.
$tableau = **split**($modele, $chaine [, $nombre]);
scinde une chaîne de caractères selon un modèle et éventuellement une longueur limite.
$chaine_result = **spliti**($modele, $chaine [, $nombre]);
scinde en ignorant la casse, une chaîne de caractères selon un modèle et éventuellement une longueur limite.
$chaine_result = **sqlrecase**($chaine);
crée une expression régulière insensible à la casse à partir d'une chaîne de caractères.

14 / Les fonctions

Les fonctions sont des modules où un ensemble d'instructions accomplit une tâche spécifique dans le but de retourner une ou plusieurs valeurs au programme appelant.

Les fonctions peuvent être **créées par l'intermédiaire de l'instruction *function***.

```
<?php
 function nom_fonction([$param_1 [= val_déf],
                                    ..., $param_N [= val_déf]])
 {
  Bloc d'instructions...
  [return valeurs...;]
 }

 $resultat = nom_fonction([arg_1, ..., arg_N]);
?>
```

L'utilisation d'une fonction nécessite la **définition d'un nom**, puis éventuellement d'**un ou plusieurs paramètres avec le cas échéant leur valeur par défaut** qui seront traités par **un bloc d'instructions**.

Enfin, **le nom de la fonction et des arguments permettront l'appel de la fonction à un endroit quelconque d'une application PHP**.

Les paramètres dans une fonction peuvent être falcutatifs, car la valeur de retour peut être indépendante de toutes données extérieures.

```
// fonction sans argument
 function RepeterCesure()
 {
  for($i = 0; $i < 5; $i++)
  {
   $cesure .= '<br>';
  }
  return $cesure;
 }

 echo RepeterCesure();

// fonction avec argument
 function RepeterCesure($nombre)
 {
  for($i = 0; $i < $nombre; $i++)
  {
   $cesure .= '<br>';
  }
  return $cesure;
 }

 echo RepeterCesure(5);
```

Les fonctions retournent un résultat au programme appelant par le truchement de l'instruction *return*. Les valeurs de retour peuvent être de n'importe quel type comme des nombres, des chaînes de caractères, des tableaux ou des objets.

Les fonctions variables consistent à affecter à des variables, des fonctions. Subséquemment, ces variables particulières utilisent les instructions de la fonction à leur profit.

```
function multiplication($nombre, $multiplicateur)
{
 echo '$nombre * $multiplicateur = ' . $nombre * $multiplicateur;
}

$fois = 'multiplication';
$fois(100, 12);
// retourne '100 * 12 = 1200'
```

14.1 / Les arguments

Il est possible de passer des arguments à une fonction soit par valeur, soit par référence.

Les valeurs passées puis traitées par la fonction peuvent être des valeurs simples de tout type, y compris des tableaux.

```
function fonction($tableau)
{
  foreach($tableau as $valeur)
  {
    $resultat *= $valeur;
  }
  return $resultat;
}

$valeurs = array(1, 2, 3, 4, 5);
echo fonction($valeurs); # retourne 120
```

Dans le premier cas, les arguments sont des valeurs passées à la fonction lors de l'appel de cette dernière. Ensuite, ces valeurs, affectées à leur variable respective dans la fonction, sont utilisées dans le bloc d'instructions.

```
<?php
  function prefixe($pfx, $balise)
  {
    $balise = $pfx . '.' . $balise;
    return $balise;
  }

  $balise = 'element_racine';
  echo '<' . $balise . '>';
# retourne '<element_racine>'
  echo '<' . prefixe('rso', $balise) . '>';
# retourne '<rso.element_racine>'
  echo '<' . $balise . '>';
# retourne '<element_racine>'
?>
```

Dans cet exemple, la variable *$variable* définie dans la fonction ne modifie absolument pas la variable de même nom déclarée dans le programme appelant.

Ainsi, dans le cas d'un passage d'argument par valeur, la variable contenant cette valeur dans le programme appelant ne sera pas affectée par des opérations dans la fonction.

Néanmoins, **il est possible de passer des arguments par référence.** Cela entraîne la **possiblité de modification des variables** du programme appelant impliquées par la fonction.

```
// Passage d'arguments par référence sur la fonction
function incrementation(&$inc)
{
  return $inc += 100;
}
$i = 0;
echo 'i = ' . $i; # affiche 0;
incrementation($i);
echo 'i = ' . $i; # affiche 100;
// Passage d'arguments par référence lors de l'appel de la fonction
function incrementation($inc)
{
  return $inc += 100;
}
$i = 0;
echo 'i = ' . $i; # affiche 0;
incrementation(&$i);
echo 'i = ' . $i; # affiche 100;
incrementation($i);
echo 'i = ' . $i; # affiche 100 et non 200;
```

Le passage d'arguments par référence **défini au niveau de la fonction implique une**

modification constante de la variable spécifiée, alors que si le passage par référence **s'effectue lors de l'appel de la fonction**, le résultat demeure le même tout en restant **limité à cet appel**.

14.2 / Les variables des fonctions

Les variables présentes au sein d'une fonction sont par définition locales, mais sous réserve d'une déclaration spéciale, leur portée peut être globale ou par ailleurs d'un type statique.

Toutes les variables déclarées dans une fonction sont utilisables localement. Elles ne peuvent être appelées et utilisées dans le corps du programme.

```
function SurfaceTriangle($largeur, $hauteur)
{
  $resultat = ($largeur * $hauteur) / 2;
  return $resultat;
}

echo SurfaceTriangle(20, 10) . " cm<sup>2</sup>;
// retourne 100 cm2

echo "($largeur * $hauteur) / 2 = " . $resultat;
// instruction erronée : variables indéfinies
```

Les variables globales peuvent être utilisées dans une fonction à condition de déclarer à nouveau les variables dans la fonction avec **le mot-clé global** ou en utilisant **le tableau associatif prédéfini *$GLOBALS***.

```
$var_glo = valeur;
function Fonction()
{
  global $var_glo;
  $var_glo++;
}

function Autre_Fonction()
{
  $GLOBALS["var_glo"]++;
}
```

Par ailleurs, **des variables peuvent être déclarées statiques dans une fonction** afin de les utiliser récursivement dans la fonction elle-même.

```
function Fonction()
{
  static $variable = 0;
  if($variable <= 0)
  {
    Fonction();
  }
  $variable++
}
```

Dans cet exemple, si la variable n'est pas statique, sa valeur restera toujours nulle puisque son initialisation à zéro anulera son incrémentation.

14.3 / Les fonctions des fonctions

Le langages PHP dispose de plusieurs fonctions permettant de travailler sur les fonctions.

Fonction
Description
$valeur = **call_user_func_array**(nom_fontion [, $tableau_paramètres]);
appelle une fonction utilisateur avec des paramètres rassemblés en tableau.
$valeur = **call_user_func**(nom_fonction [, $param_1, ..., $param_N]);
appelle une fonction utilisateur avec zéro ou plusieurs paramètres.
nom_fonction = **create_function**('$param_1 [, ..., $param_N]', 'return instruction...');
crée une fonction avec un ou plusieurs paramètres et un code simple.
$valeur = **func_get_arg**($numero_argument);
retourne un élément de la liste des arguments.
$tableau = **func_get_args**();
retourne la liste des arguments sous forme de tableau.
$nombre = **func_num_args**();
retourne le nombre d'arguments dans une fonction.
true | false = **function_exists**(nom_fonction);
vérifie si la fonction existe.
$tableau = **get_defined_functions**();
retourne un tableau multidimensionnel contenant la liste de toutes les fonctions définies dans le programme.
$entier = **register_shutdown_function**(nom_fonction);
enregistre une fonction pour une exécution à l'extinction du script.
register_tick_function(nom_fonction [, $argument]);
enregistre une fonction à chaque événement intervenant à chaque commande de bas niveau exécutées par l'analyseur dans le bloc de directive *declare*.
unregister_tick_function();
annule la fonction à chaque événement intervenant à chaque commande de bas niveau exécutées par l'analyseur dans le bloc de directive *declare*.

15 / Les objets

Le langage PHP supporte certaines notions de la programmation orientée objet et partant permet de développer ses propres objets possédant des caractéristiques spécifiques pour des applications internet ambitieuses.

Les objets sont issus d'instanciations de classes qui leurs fournissent des attributs, des propriétés et des méthodes permettant d'accomplir des tâches prédéfinies dans un script.

15.1 / Les classes

Une classe est une structure de référence à partir de laquelle se concrétise un objet.

Le code au sein d'un module de classe, décrit :

- **les attributs** (ou champs), soit les variables d'instance déclarées au sein de la classe,
- **les champs statiques**, soit les variables de classe déclarées au sein de la classe,
- **les constantes**, soit des champs contenant des valeurs invariables déclarés au sein de la classe,
- **les constructeurs**, soit des fonctions spécialisées dans l'initialisation de l'état de l'instance de classe,
- **les méthodes**, soit des fonctions définies au sein de la classe.

Les composants d'une classe, ceux précités, sont appelés **les membres d'une classe**.

D'autre part, les caractéristiques des attributs telles que leur type de données et leur valeur sont appelées les propriétés que possédera l'objet suite à sa création et à son exploitation.

Une classe est conçue à partir de l'instruction *class* qui contiendra la déclaration des attributs ainsi que les fonctions et leurs instructions.

```
[final|abstract] class nom_classe
{
  // Déclarations des attributs
  $attribut_1;
  //...
  $attribut_N;

  // Déclarations des méthodes
  function nom_fonction_1 (param_1, ..., param_N)
  {
    [$this->attribut_1 = valeur;]
  //bloc d'instructions...
  }
  //...
  function nom_fonction_N (param_1, ..., param_N)
  {
    [$this->attribut_1 = valeur;]
  //bloc d'instructions...
  }
}
```

Une classe possède donc **des attributs et des méthodes**, lesquels fournissent à l'objet, instance de cette classe, **un état et un comportement**.

- **L'état correspond au contenu de l'objet**, c'est à dire, aux valeurs de ses attributs.
- **Le comportement est quant à lui régit par les méthodes agissant sur l'objet.**

Dans une classe, **la pseudo-variable *$this* constitue une référence vers l'objet courant, qui permet à une fonction d'accéder aux méthodes et attributs déclarés dans la classe courante**.

Ensuite, **un objet est créé au moyen d'une instanciation de la classe** par l'intermédiaire de l'instruction *new*.

```
$objet = new nom_classe();
```

A partir de l'objet, il devient possible d'**accéder à ses attributs et méthodes en utilisant un tiret '-' suivi d'un signe *supérieur à* '>'** entre le nom de l'objet et respectivement le nom de l'attribut ou de la fonction déclarées dans la classe.

```
// Accès aux propriétés de l'objet
$nom_objet->nom_attribut = valeur;

// Accès aux méthodes de l'objet
$nom_objet->nom_fonction(arg_1, ..., arg_N);
```

Exemple [voir]

```
<?php
 class surface
 {
  var $hauteur;
  var $longueur;
  var $rayon;
  var $diametre;

  function triangle($H, $L)
  {
   $this->hauteur = $H;
   $this->longueur = $L;
   return $H * $L / 2;
  }

  function rectangle($H, $L)
  {
   $this->hauteur = $H;
   $this->longueur = $L;
   return $H * $L;
  }

  function carre($L)
  {
   $this->longueur = $L;
   return $L * $L;
  }

  function cercle($T, $type)
  {
   if ($type = 'd')
   {
    $this->diametre = $T;
    return (pow(2, $T) * M_PI) / 4;
   }
   else
   {
    $this->rayon = $T;
    return pow(2, $T) * PI;
   }
  }
 }

 $d = 12;
 $choix = array('d' => 'diamètre', 'r' => 'rayon');

 $calculSurface = new surface();
 $resultat = $calculSurface->cercle($d, $choix['d']);

 echo 'La surface du cercle avec un <b>' . $choix['d'] . '</b> de <b>'
      . $calculSurface->diametre . ' cm</b> est <b>'
      . $resultat . ' cm<sup>2</sup></b>.';
?>
```

15.1.1 / La déclaration des classes

La déclaration d'une classe constitue une étape essentielle de la programmation orientée objet.

La déclaration d'une classe s'effectue par l'intermédiaire du mot clé *class*.

```
[abstract|final] class Identificateur {
  //Bloc d'instructions...
}
```

Les modificateurs possibles pour une classe de haut niveau ne peuvent être que abstraites (*abstract*), finales (*final*).

Le modificateur *final* indique que la classe ne peut être étendue.

```
final class Classe {
  //...
}
//Impossible car Classe ne peut être étendue
class SousClasse extends Classe {
  //...
}
```

Le modificateur *abstract* indique que la classe est abstraite.

Le mot-clé *class* permet de commencer la définition d'une classe.

Une classe est accessible à travers tout le programme à condition d'inclure le fichier source contenant la définition de cette classe.

```
include(source_classe.php);
```

Un nom permet d'identifier sans ambiguités la classe dans un programme.

Les accolades délimitent le bloc d'instructions définissant les attributs, les méthodes, les constructeurs, les champs statiques et les constantes ainsi que les éventuelles classes imbriquées.

Exemple [voir]

```
<html>
<body>
<?php
class Societe {
  //Attributs
  private $nom;
  private $telephone;
  private $fax;
  private $adresse;

  //Constructeur
  public function Societe($nom,
                          $telephone,
                          $fax,
                          $adresse){
    $this->nom = $nom;
    $this->telephone = $telephone;
    $this->fax = $fax;
    $this->adresse = $adresse;
  }

  //Méthodes
  public function getNom() {
    return $this->nom;
  }
  public function getTelephone() {
    return $this->telephone;
  }
  public function getFax() {
    return $this->fax;
  }
  public function getAdresse() {
    return $this->adresse;
  }
  public function __toString() {
    return '[' . $this->nom . ', '
               . $this->telephone . ', '
               . $this->fax . ', '
               . $this->adresse . ']';
  }
}

$fic = fopen('assurances.csv', 'r');

echo '<table border="1">';
while(!feof($fic)){
  $ligne = fgetcsv($fic, 1024, ';');
  if(substr($ligne[0], 0, 1) == '#') continue;
  list($nom,
     $adresse,
     $telephone,
     $fax) = $ligne;
     $societe = new Societe($nom,
                            $adresse,
                            $telephone,
                            $fax);
  echo '<tr>';
  echo '<td>';
  echo $societe;
  echo '</td>';
  echo '</tr>';
}
echo '</table>';
?>
</body>
</html>
```

15.1.2 / L'instanciation des classes

Les objets sont obtenus à partir d'une instanciation de classe.

Tous **les objets sont issus de classes** créées par l'intermédiaire de l'instruction *class* suivi d'un identificateur.

```
public class nom_classe{
  //instructions...
}
```

Un objet est caractérisé par des attributs le décrivant et des comportements représentant ces actions.

Les attributs sont en fait les variables contenues déclarées dans la classe de l'objet.

Les comportements sont les méthodes figurant dans cette même classe.

```
class calcul{
  pivate $i = 10;
  private $j = 12;
  public function multiplication(){
    echo $this->i . ' * ' . $this->j . ' = ' . $this->i * $this->j;
  }
}

/* Deux attributs i et j Une méthode multiplication */
```

L'utilisation d'un objet dans un programme est possible à condition de déclarer une instance d'un type de classe. Cette opération s'effectue à partir de **l'opérateur *new*** affectant une instance de classe à une variable.

```
$nom_objet = new nom_classe();

$operation = new calcul();
```

A partir de là, il devient possible d'**utiliser les variables et les méthodes de la classe à partir de l'objet**.

```
//premiere_operande = 10
$premiere_operande = $operation->i;
//seconde_operande = 12
$seconde_operande = $operation->j;
//affiche 10 * 12 = 120
$operation->multiplication();

//seconde_operande = 12
$operation->j = 24;
//affiche 10 * 24 = 240
$operation->multiplication();
```

Chaque classe peut contenir une méthode spécialisée dénommée *__construct()*, laquelle permet d'initialiser les variables membres de la classe. Cette méthode ne peut renvoyer une valeur de retour.

Lors de l'utilisation de l'opérateur *new*, le constructeur est automatiquement appelé afin d'affecter des ressources telles que la définition et l'initialisation de variables, nécessitées par l'objet.

```
class calcul{
  public $i;
  public $j;
  public function __construct($val1, $val2){
    $this->i = $val1;
    $this->j = $val2;
  }
  public multiplication(){
    return $this->i * $this->j;
  }
```

```
}

$operation = new calcul(250, 100);
//affiche 250 * 100 = 25000
$resultat = $operation->multiplication();

echo $operation->i . ' * ' . $operation->j . ' = ' . $resultat;
```

15.1.3 / Les classes abstraites

L'utilisation du modificateur *abstract* dans une déclaration de classe signifie qu'une ou plusieurs des méthodes à l'intérieur du corps de la classe sont déclarées comme abstraites.

```
abstract class NomClasse {
  modificateur Type NomMethode();
  ...
}
```

Si une méthode à l'intérieur de la classe est déclarée avec le modificateur *abstract*, alors cette classe doit être définie explicitement comme abstraite.

```
abstract class NomClasse {
  abstract modificateur nomMethode([$Parametres, ...]);
  ...
}
```

Les classes déclarées avec le modificateur *abstract* ne peuvent être instanciées. Les classes abstraites contraignent les classes qui les étendent, à un comportement imposé. Les classes dérivées doivent fournir un bloc d'instructions pour chacune des méthodes abstraites héritées de la classe portant le modificateur *abstract*. La nouvelle classe dérivée pourra alors être instanciée.

```
abstract class NomClasse {
  modificateur NomMethode();
  ...
}

class ClasseDerivee extends NomClasse {
  modificateur_acces nomMethode() {
    // Bloc d'instructions...
  }
  ...
}
```

Une classe dérivée d'une classe abstraite **doit impérativement implémenter toutes les méthodes abstraites** de cette dernière, sinon elle doit porter le modificateur *abstract* dans sa déclaration.

Une interface pouvant être considérée comme étant une classe abstraite, une classe qui en implémente une doit être **déclarée abstraite si toutes les méthodes de l'interface n'y sont pas implémentées**.

Les classes abstraites sont utilisées pour fournir des méthodes et des variables communes à l'ensemble de leurs sous-classes.

Les classes abstraites, au contraire des interfaces, peuvent contenir des méthodes qui possèdent une implémentation.

```
abstract class Produit {
  public function getNom();
  public function getReference();
  public function getPrix();
  public function __toString() {
    return '[' . $this->getNom() .', '
             . $this->getReference() . ', '
             . $this->getPrix() . ']';
  }
}

class Livre extends Produit {
  private $titre;
  private $isbn;
  private $prix;
  private $description;
  $private $nb_pages;
  //...
```

```
  public function getNom(){
    return $this->titre;
  }
  public function getReference(){
    return $this->isbn;
  }
  public function getPrix(){
    return $this->prix;
  }
  public function getDecription(){
    return $this->description;
  }
  public function getNbPages(){
    return $this->nb_pages;
  }
  //...
}

class DVDVideo extends Produit {
  private $titre;
  private $code_barre;
  private $prix;
  private $description;
  $private $duree;
  //...

  public function getNom(){
    return $this->titre;
  }
  public function getReference(){
    return $this->code_barre;
  }
  public function getPrix(){
    return $this->prix;
  }
  public function getDecription(){
    return $this->description;
  }
  public function getDuree(){
    return $this->duree;
  }
  //...
}
```

15.1.4 / L'héritage de classe

L'héritage est un principe fondamental de la programmation orientée objet, elle permet à une nouvelle classe de posséder les caractéristiques d'une autre appelée *super-classe* ou *classe générique*. La classe ayant héritée d'une autre, est quant à elle dénommée sous-classe, classe étendue ou classe dérivée

L'héritage de classe se réalise par l'intermédiaire de l'**instruction *extends***.

```
class super_classe
{
  // Déclarations des attributs
  var $attribut_1;
  var $...
  var $attribut_N;

  // Déclarations des fonctions
  function nom_fonction_1 (param_1, ..., param_N)
  {
  //bloc d'instructions...
  }
  ...
  function nom_fonction_N (param_1, ..., param_N)
  {
  //bloc d'instructions...
  }
}

class sous_classe extends super_classe
{
  // Déclarations des nouveaux attributs
  var $attribut_1;
  var $...
  var $attribut_N;

  // Déclarations des nouvelles fonctions
  function nom_fonction_1 (param_1, ..., param_N)
  {
  //bloc d'instructions...
  }
  ...
  function nom_fonction_N (param_1, ..., param_N)
  {
  //bloc d'instructions...
  }
}
```

Une classe dérivée hérite de l'ensemble des membres de sa super-classe. La sous-classe peut ainsi définir de nouveaux membres qui ajouteront de nouvelles fonctionnalités à celles existantes, mais aussi de redéfinir des membres hérités à l'aide de la surcharge.

Le langage PHP ne supporte pas l'héritage multiple, c'est-à-dire qu'une classe ne peut hériter que d'une seule et unique classe.

Exemple [voir]

```
<html>
<body>
<?php
class Chaine {
   public $valeur;
   public $date;
   public function __construct(){
      $this->valeur = func_get_arg(0);
      $this->date = date(Personne::MASQUE_DATE);
   }
   public function set($valeur){
      $this->valeur = $valeur;
   }
   public function get(){
      return $this->valeur;
   }
   public function __toString(){
     return '[' . $this->valeur . ']';
   }
}
class Personne {
   const MASQUE_DATE = 'd M Y H:i:s';
   static $compteur;
   private $date_inscription;
   private $date_modification;
   private $id;
   private $civilite;
   private $nom;
   private $prenom;
   private $adresse;
   private $telephone;
   public function __construct(
                        $civilite,
                        $nom,
                        $prenom,
                        $adresse,
                        $telephone){
      $this->date_inscription = new Chaine(date(self::MASQUE_DATE));
      $this->civilite = new Chaine($civilite);
      $this->nom = new Chaine($nom);
      $this->prenom = new Chaine($prenom);
      $this->adresse = new Chaine($adresse);
      $this->telephone = new Chaine($telephone);
      $this->date_modification = new Chaine($this->date_inscription->get());
      $this->idModif();
   }

   public function __toString(){
    return '[' . $this->civilite->get() . ', '
              . $this->nom->get() . ', '
              . $this->prenom->get() . ', '
              . $this->adresse->get() . ', '
              . $this->telephone->get() . ', '
              . $this->date_inscription->get() . ', '
              . $this->date_modification->get() . ', '
              . $this->id->get() . ']';
   }
   public function setNom($nom){
      $this->nom->set($nom);
      $this->dateModif();
      $this->idModif();
   }
   public function setPrenom($prenom){
      $this->prenom->set($prenom);
      $this->dateModif();
      $this->idModif();
   }
   public function setAdresse($adresse){
      $this->adresse->set($adresse);
      $this->dateModif();
   }
   public function setTelephone($telephone){
      $this->telephone->set($telephone);
      $this->dateModif();
```

```
  }
  private function dateModif(){
    $this->date_modification->set(date(self::MASQUE_DATE));
  }
  private function idModif(){
    $this->id = new Chaine(self::$compteur++ . date('dmy')
                . '.' . $this->nom->get()
                . '_' . substr($this->prenom, 0, 1));
  }

  public function __clone(){
    $this->date_inscription = new Chaine(date(self::MASQUE_DATE));
    $this->civilite = new Chaine($this->civilite->get());
    $this->nom = new Chaine($this->nom->get());
    $this->prenom = new Chaine($this->prenom->get());
    $this->adresse = new Chaine($this->adresse->get());
    $this->telephone = new Chaine($this->telephone->get());
    $this->date_modification = new Chaine($this->date_inscription->get());
    $this->idModif();
  }
}
class Employe extends Personne {
  public $specialite;
  public $service;
  public $date_embauche;
  public function __construct($civilite,
                $nom,
                $prenom,
                $adresse,
                $telephone,
                $specialite,
                $service){
    //Appel du constructeur de la super-classe à l'aide de parent::
    parent::__construct($civilite,
                $nom,
                $prenom,
                $adresse,
                $telephone);
    $this->specialite = new Chaine($specialite);
    $this->service = new Chaine($service);
    $this->date_embauche = new Chaine(date(parent::MASQUE_DATE));
  }
  //Surcharge
  public function __toString(){
    //Appel d'une méthode de la super-classe à l'aide de parent::
    return '[' . parent::__toString() .', '
            . $this->specialite->get() . ', '
            . $this->service->get() . ', '
            . $this->date_embauche->get() . ']';
  }
}
$employe = new Employe(
                'Madame',
                'Waldec',
                'Anne',
                '75 rue Belleville 75010 Paris',
                '0142152432',
                'Analyste/programmeur',
                'Recherche et développement');

echo $employe;
?>
</body>
</html>
```

15.1.5 / La fonction *__toString*

La fonction *__toString()* est chargée de retourner une représentation textuelle d'un objet.

De cette manière un objet peut fournir un aperçu de son état, sous la forme d'une chaîne de caractères.

```
class Personne {
  private $civilite;
  private $nom;
  private $prenom;
  private $adresse;
  private $telephone;
  public function __construct(
                    $civilite,
                    $nom,
                    $prenom,
                    $adresse,
                    $telephone){
    $this->civilite = $civilite;
    $this->nom = $nom;
    $this->prenom = $prenom;
    $this->adresse = $adresse;
    $this->telephone = $telephone;
  }

  public function __toString(){
   return '[' . $this->civilite . ', '
            . $this->nom . ', '
            . $this->prenom . ', '
            . $this->adresse . ', '
            . $this->telephone . ']';
  }
}

$personne = new Personne(
                'Madame',
                'Waldec',
                'Anne',
                '75 rue Belleville 75010 Paris',
                0142152432);
//Appel implicite de la méthode __toString()
echo $personne;
```

La fonction *__toString()* est applelée automatiquement lorsque l'objet est appelé automatiquement à partir d'une commande *echo* ou *print*.

Néanmoins dans certaines conditions, la fonction *toString()* ne pourra être invoquées implicitement à l'exécution du programme.

- Lorsque l'objet est associé à un opérateur de concaténation.

  ```
  echo 'Une chaine ' . $objet;
  ```

- Lorsque l'objet se trouve au sein d'une chaine de caractères.

  ```
  echo "Une chaine $objet";
  ```

- Lorsque l'objet subi une opération de conversion.

  ```
  echo (string)$objet;
  ```

  Dans le premier cas, il est possible de pallier à ce problème en passant une liste de paramètres à la fonction *echo*.

  ```
  echo 'Une chaine ', $objet, 'Une autre chaine';
  ```

15.1.6 / La fonction *__autoload*

La fonction magique *__autoload()* est chargée d'inclure les classes que le programme aurait besoin pour son exécution.

```
function __autoload($nom_classe) {
  require_once $nom_classe . '.php';
}
```

Lors de l'exécution d'un programme, PHP appelera automatiquement la fonction *__autoload()* dès qu'il rencontrera une classe inconnue. La fonction *__autoload()* tentera alors de trouver la classe adéquate.

```
//Fichier Classe.php
class Classe {
 public function __autoload($nom){
  if(include_exists($nom . '.inc')) {
   include_once($nom . '.inc');
  }
 }
}
$obj = new Classe(); //Appel automatique de __autoload()

//Fichier Classe1.inc
class Classe1 {
 //...
}
```

L'utilisation efficace de cette fonction demande de se conformer à certaines règles.

Premièrement, il faut que chaque classe possède son propre fichier et que ce dernier porte le même nom que la classe, avec exactement la même orthographe et la même casse de caractères (majuscule et minuscule).

Deuxiémement, que tous les fichiers portent la même extension (ex.: *.php* ou *.inc*).

Troisièmement, dans le cas, d'un programme complexe faisant appel à des classes concrètes, abstraites et des interfaces, il peut être souhaitable de préfixer chaque type de fichier par une expression afin d'éviter toutes ambiguités.

```
cls_UneClasseConcrete.php
abs_UneClasseAbstraite.php
int_UneInterface.php
```

Le non-respect des deux premières règles provoquera certainement des difficultés dans l'élaboration du code de la fonction *__autoload()*, puisqu'il faudra composer avec divers extensions et des noms de fichiers qui ne correspondront pas forcément aux noms de classes.

```
<?php
 function __autoload($name) {
    $class = $interface = false;

    //Interface
    if (include_exists("interface.$name.inc")) {
       include_once("interface.$name.inc");
       $interface    = true;
    }
    //Classe
    if (include_exists("class.$name.inc")) {
       include_once("class.$name.inc");
       $class    = true;
    }

    //Si l'interface ou la classe n'a pas été trouvée
    if ($class === false && $interface === false) {
       trigger_error("Impossible d'inclure l'interface "
                . "ou la classe $name", E_USER_WARNING);
       return false;
    }
```

```
        return true;
    }

    function include_exists($file)
    {
        static $include_dirs = null;
        static $include_path = null;

        // set include_dirs
        if (is_null($include_dirs)
        || get_include_path() !== $include_path) {
            $include_path    = get_include_path();
            foreach (split(PATH_SEPARATOR, $include_path) as $include_dir) {
                if (substr($include_dir, -1) != '/') {
                    $include_dir .= '/';
                }
                $include_dirs[]    = $include_dir;
            }
        }

        if (substr($file, 0, 1) == '/') {
            //absolute filepath - what about file:///?
            return (file_exists($file));
        }
        if ((substr($file, 0, 7) == 'http://'
        || substr($file, 0, 6) == 'ftp://')
          && ini_get('allow_url_fopen')) {
            return true;
        }
        foreach ($include_dirs as $include_dir) {
            if (file_exists($include_dir.$file)) {
                return true;
            }
        }
        return false;
    }
    // voir http://www.php.net/manual/en/language.oop5.autoload.php
?>
```

15.2 / Les interfaces

Une interface est un modèle d'implémentation, permettant de mettre en relation des entités sans rapports.

Une interface, définissant un protocole de comportement, est **un élément que des objets sans liens utilisent pour interagir entre eux**.

La déclaration d'une interface s'effectue par l'intermédiaire de **l'instruction *interface***.

```
interface NomInterface {
  //...
}
```

De plus, PHP ne permettant pas l'héritage multiple, l'utilisation des interfaces permet de palier à cette carence dans la mesure où **une classe peut désormais implémenter plusieurs interfaces**.

L'instruction *implements* permet d'indiquer que la classe fournit une implémentation pour une ou plusieurs interfaces. Ce mot clé doit toujours être placé après la clause d'extension introduite par *extends*.

```
class NomClasse implements NomInterface, ..., NomInterfaceN {
   //...
}
```

Les interfaces possèdent en général dans leur bloc d'instructions, des déclarations de méthodes sans corps ainsi que des déclarations de variables.

```
// Déclaration d'une méthode
modificateur_acces function nomMethode();
// Déclaration d'une variable
const nomVariable = valeurConstante;
```

Toutes les méthodes contenues dans une interface sont **implicitement déclarées publiques et abstraites**.

Les méthodes ne peuvent être déclarées avec le modificateur *static*.

Toutes les variables, en fait des constantes, contenues dans une interface doivent être déclarées avec le modificateur *const*. Elles doivent avoir une valeur constante d'affectation.

Toutes les classes implémentant une interface doivent possèder les méthodes déclarées dans cette dernière en y ajoutant un bloc d'instructions.

```
interface ValidationEnvoi {
  const PRIX_TIMBRE = 0.46;
  public function Poste(float timbre, String adr_exp);
  public function PossedeUnTimbre();
  public function PossedeUneAdressedExpedition();
  public function Envoi();
}
class Lettre implements ValidationEnvoi {
  $nb_timbre
  $adr_exp = null;
  $valide_timbre = false;
  $valide_adresse = false;

  public function Poste($timbre, $adr_exp) {
    $this->nb_timbre = timbre;
    $this->adr_exp = adr_exp;
  }

  public function PossedeUnTimbre() {
    if ($this->nb_timbre > 0)
      return $this->valide_timbre = true;
    else
      return $this->valide_timbre = false;
  }

  public function PossedeUneAdressedExpedition() {
    if (is_null($this->adr_exp))
      return $this->valide_adresse = true;
    else
      return $this->valide_adresse = false;
  }

  public function Envoi() {
    if ($this->valide_timbre && $this->valide_adresse)
      return true;
    else
      return false;
  }
  public function PrixTimbres() {
    return $this->nb_timbre * self::PRIX_TIMBRE;
  }
}
$adresse = "Mlle Julie LAMYrn"
           . "10 Allée d'Artoisrn"
           . "93330 Neuilly Sur Marne";
$affranchissement = 1;

$envoiLettre = new Lettre();

$envoiLettre->Poste(affranchissement, adresse);

echo "Adresse d'expedition : " .
   ($envoiLettre->PossedeUneAdressedExpedition() ? "Oui" : "Non");

echo "Affranchissement : " .
   ($envoiLettre->PossedeUnTimbre() ? "Oui" : "Non");

echo "Procedure d'envoi validee : " .
   ($envoiLettre->Envoi() ? "Oui" : "Non");
```

Les interfaces **ne peuvent implémenter d'autres interfaces**. Par contre, elles **peuvent hériter de plusieurs autres interfaces**.

```
public interface NomInterface
        extends AutreInterface, ..., AutreInterface {
  //...
}
```

Le bloc d'instructions des interfaces contient essentiellement des déclarations de constantes et de méthodes abstraites.

Les interfaces constituent **une catégorie particulière des classes abstraites**.

15.3 / Le constructeur

Le constructeur est une fonction qui est appelée automatiquement par la classe lors de son instanciation avec l'opérateur *new*.

```
class super_classe
{
  function super_classe()
  {
    //bloc d'instructions...
  }
}

class nouvelle_classe extends super_classe
{
  function nouvelle_classe()
  {
    //bloc d'instructions...
  }
}
```

La fonction constructeur doit posséder un nom identique à celle de la classe.

Une différence importante existe entre la version 3 et 4 du langage PHP quant à la gestion des constructeurs. Avec PHP 3, une fonction définie dans une classe héritée devient un constructeur si son nom est similaire à celle de la nouvelle classe, tandis qu'avec la quatrième version, une fonction constructeur ne peut être définie que dans sa propre classe évitant ainsi toutes ambguités.

```
class classe_X
{
  function classe_X()
  {
    echo "Bloc d'instructions de la fonction classe_X...";
  }

  function classe_Y()
  {
    echo "Bloc d'instructions de la fonction classe_Y...";
  }
}

class classe_Y extends classe_X
{
  function fonction()
  {
    echo "Bloc d'instructions de la fonction fonction...";
  }
}
/* Quelque soit la version, l'expression retourne :
'Bloc d'instructions de la fonction classe_X...' */
$objet = new classe_X();

/* PHP 3 : l'expression retourne :
'Bloc d'instructions de la fonction classe_Y...' */
/* PHP 4 : l'expression retourne :
'Bloc d'instructions de la fonction classe_X...' */
$snd_objet = new classe_Y();
```

Avec la version 4, lorsqu'une classe héritant d'une autre est instanciée et **si aucun constructeur n'est défini dans cette classe**, alors la fonction constructeur sollicitée sera celle de la super-classe, si elle même existe.

Si la classe étendue possède son propre constructeur, alors ce dernier devra appeler le constructeur de la superclasse dans le cas où celui-ci serait nécessaire à l'initialisation de l'objet.

```
class BaseDonnees {
  protected $id;
  function BaseDonnees(){
    $hote = 'hote.mysql.com';
```

```
    $utilisateur = 'laltruiste';
    $motpasse = 'PaSsWORd';
    $this->id = mysql_connect($hote, $utilisateur, $motpasse);
    if(!$this->id & !mysql_select_db("base", $this->id))
      echo "<h3>Une erreur est survenue :</h3>"
         . "<b><u>Erreur numéro " . mysql_errno()
         . " :</u> " . mysql_error() . "</b>";
  }
  function extraction($tables, $donnees, $conditions){
    //sélection dans une table
    if($conditions)
      $conditions = " WHERE ".$conditions;
    $requete = "SELECT ".$donnees." FROM ".$tables.$conditions.";";
    $id_res = mysql_query($requete, $this->id);
    if($id_res)
      return $id_res;
    else {
      echo '<h3>Une erreur est survenue :</h3>'
       . '<b><u>Erreur numéro ' . mysql_errno()
       . ' :</u> ' . mysql_error() . '</b>'
       . '<p style="color:red">Tentative de connexion
                                à la base de données</p>';
      return false;
    }
    mysql_free_result($id_res);
  }
  //...
}

class document extends BaseDonnees {
  private $cle;
  function document($valeur){
    $super->BaseDonnees();
    $id_res = $this->extraction('table', 'champs', 'id_table = '.$valeur);
    $this->cle = mysql_result($id_res, 0, 0);
  }
  //...
}
```

15.3.1 / Les fonctions *__construct()* et *__destruct()*

Le constructeur est une méthode spéciale d'une classe, permettant d'initialiser les membres de cette classe et également, d'exécuter différentes opérations initiales définies par le programmeur.

L'instanciation d'une classe (ou la création d'un objet) nécessite l'utilisation de l'opérateur *new* appelant un constructeur.

```
$ref_objet = new Constructeur([$arg1, ..., $argN]);
```

Le nom du constructeur porte exactement le même nom que la classe courante.

```
class NomClasse {
  public function NomClasse([$arg1, ..., $argN]){
    //...
  }
}
```

Le langage PHP utilise également une fonction spécifique, à déclarer au sein de la classe destinée à l'instanciation. Il s'agît de la méthode *__construct()*.

```
ref_objet = new IdentificateurClasse();
```

En l'absence d'une déclaration de constructeur dans une classe, le compilateur Java en génère automatiquement un par défaut, lequel n'aura pas d'argument et aucune autre fonction que de créer un objet.

```
class UneClasse {
  // Aucun constructeur défini
  ...
}
// est équivalent à
class UneClasse {
  // Constructeur de la classe UneClasse
  public function __construct(){
    //Initialisation des membres de la classe
  }
  ...
}
```

Afin d'éviter cette création automatique, il peut être parfois préférable de **déclarer un constructeur mieux adapté aux besoins de la classe**.

```
class NomClasse {
  // méthode constructeur
  modificateur __construct([Paramètre, ...]) {
  // Bloc d'instructions...
  }
}
...
//Instanciation de la classe
$ref_objet = new NomClasse([Arguments]);
```

Le constructeur doit avoir **un identificateur en tout point semblable à celui de sa classe**.

Les constructeurs peuvent être déclarés avec un modificateur d'accès *public*, *protected*, *par défaut* ou *private*.
Les constructeurs sont par défaut uniquement accessibles dans le paquetage de la classe parente.
Le modificateur *private* entraîne l'impossibilité d'instancier la classe du constructeur et également de l'étendre.

Un constructeur ne peut être déclaré avec les modificateurs suivants : *abstract*, *static*, *final*.

Le constructeur **ne peut retourner une valeur**, et donc, sa déclaration **ne contient aucun type de retour**. En fait, **l'instance de l'objet est toujours le type retourné** par un constructeur.

Le constructeur **peut posséder un à plusieurs paramètres** permettant notamment l'affectation

des champs de l'objet avec des valeurs fournies par l'utilisateur.

```
class Article {
  private $idProduit;
  private $prix;
  private $reduction;

  public function __construct($idProduit, $prix, $reduction) {
    $this->idProduit = $idProduit;
    $this->prix = $prix;
    $this->reduction = $reduction;
  }
}

$livre = new Article(9782212075021, 55.0, 5.0);
```

La clause *throws* optionnelle permet d'indiquer que le constructeur peut déclencher les exceptions spécifiées.

La définition d'un constructeur doit apparaître au moins une fois dans une classe, mais n'est en aucun cas une nécessité absolue, dans la mesure où aucune initialisation particulière ne serait nécessaire au démarrage de l'objet.

Parfois, il est utile de **définir dans une classe, plusieurs constructeurs avec des signatures distinctes** (voir La surcharge des méthodes). Cela permet d'initialiser un objet de plusieurs façons différentes.

```
class Article {
  private $nomProduit;
  private $idProduit;
  private $prix;
  private $reduction;

  public Article($idProduit, $prix, $reduction) {
    $this->idProduit = $idProduit;
    $this->prix = $prix;
    $this->reduction = $reduction;
  }

  public Article($idProduit, , $nomProduit, $prix, $reduction) {
    $this->idProduit = $idProduit;
    $this->nomProduit = $nomProduit;
    $this->prix = $prix;
    $this->reduction = $reduction;
  }
}

$livre1 = new Article(9782744090004, 49.0, 0.0);

$livre2 = new Article(9782744090004, "Maîtrisez Java 2", 49.0, 0.0);
```

Il est possible également d'utiliser les fonctions *func_num_args()*, *func_get_arg(0)* et *func_get_args()* pour récupérer les arguments passés au constructeur et réagir en fonction des arguments passés.

```
function __construct() {
  $nb_args = func_num_args();
  if($nb_args == 3){
    $this->idProduit = func_get_arg(0);
    $this->prix = func_get_arg(1);
    $this->reduction = func_get_arg(2);
  }
  else if($nb_args == 4){
    $this->idProduit = func_get_arg(0);
    $this->nomProduit = func_get_arg(1);
    $this->prix = func_get_arg(2);
    $this->reduction = func_get_arg(3);
  }
  else {
    echo 'Le nombre d'arguments est incorrect !';
  }
}
```

Afin de s'assurer du type des arguments, il peut être souhaitable d'utiliser les fonctions de tests de type, telles que *is_array()*, *is_bool()*, *is_int()*, *is_float()*, *is_string()*, etc..

```
if(is_long(func_get_arg($indice))
   $this->idProduit = func_get_arg($indice);
else if(is_float(func_get_arg($indice))
   $this->prix = func_get_arg($indice);
else if(is_string(func_get_arg($indice))
   $this->nomProduit = func_get_arg($indice);
//...
```

Utilisé avec des constructeurs multiples, **l'opérateur de résolution de portée *::* permet d'appeler le constructeur possèdant les arguments correspondants au sein de la classe courante (*self::*) ou de la classe parente (*parent::*)**. Ceci permet d'abréger la rédaction des affectations dans les définitions de constructeurs.

```
class Produit {
  private $nomProduit;
  private $idProduit;
  private $prix;
  private $reduction;

  function __construct($nomProduit,
                       $idProduit,
                       $prix,
                       $reduction) {
    $this->idProduit = $idProduit;
    $this->nomProduit = $nomProduit;
    $this->prix = $prix;
    $this->reduction = $reduction;
  }
}

class Livre extends Produit {
  private $auteur;
  private $editeur;
  private $nb_pages;

  function _construct($auteur,
                      $editeur,
                      $nb_pages) {
    $this->auteur = $auteur;
    $this->editeur = $editeur;
    $this->nb_pages = $nb_pages;
  }
  function __construct($nomProduit,
                       $idProduit,
                       $prix,
                       $reduction,
                       $auteur,
                       $editeur,
                       $nb_pages) {
    parent::__construct($nomProduit,
                       $idProduit,
                       $prix,
                       $reduction);
    self::__construct($auteur,
                       $editeur,
                       $nb_pages);
  }
}

$livre = new Article(9782746024588,
                     'PHP 5',
                     27.14,
                     0.0,
                     'Olivier Heurtel',
                     'Eni Editions',
                     504);
```

A l'opposé des constructeurs, PHP 5 introduit un concept de destructeur. La méthode destructeur, dénommée *__destruct()*, est destinée à décharger les ressources utilisées par l'objet. cette méthode est appelée automatiquement après que la dernière référence à un objet est supprimée ou lorsque cet objet est explicitement détruit.

```
class UneClasse {
  private $var;
  function __construct($var) {
    $this->var = $var;
  }

  function __destruct() {
    $this->name = Null;
  }
}
//Appel du constructeur
$objet = new UneClasse();

//Appel du destructeur
$objet = Null;
```

15.4 / Les méthodes

Une méthode est un sous-programme destiné à exécuter un bloc d'instructions suite à son appel dans un programme principal.

Les méthodes sont destinées à fournir un comportement à l'instance de classe ou à la classe elle même , c'est-à-dire, de définir diverses opérations à accomplir, à l'instar d'un calcul mathématique, d'une affectation d'attribut ou l'initialisation d'un objet.

Les méthodes se déclarent de la même façon qu'une fonction PHP, **en utilisant l'instruction *function***.

```
[modificateur] function nomMethode([liste_params]){
  //Instructions...
}
```

La partie déclarative d'une méthode, soit l'ensemble des éléments précédant le bloc d'instructions délimités par des accolades, s'appelle **la signature de la méthode**. Cette dernière contient toutes les caractéristiques permettant d'utiliser convenablement les méthodes. Ainsi, on distingue :

- les modificateurs d'accès et spéciaux,
- le type de valeur retourné,
- le nom de la méthode,
- la liste des paramètres.

Une méthode peut être soit d'instance, soit de classe. Une méthode d'instance nécessite une instanciation de la classe pour pouvoir être invoquée, tandis qu'une méthode de classe peut être appelée sans qu'une instanciation de classe ait été effectuée au préalable. **Les méthodes de classe se distinguent par le modificateur *static*.**

```
class Classe {
  public function methodeInstance(){
    //Instructions...
  }
  public static function methodeClasse(){
    //Instructions...
  }
}

Classe->methodeClasse();

$obj = new Classe(); //Instanciation de classe
$obj->methodeInstance();
```

Le modificateur d'accès (*public*, *private* ou *protected*) détermine la visibilité de la méthode dans sa propre classe, ainsi que dans d'autres classes et également dans l'ensemble du programme.

```
class Classe {
  public function methodeInstance_a(){
    //Instructions...
  }
  protected function methodeInstance_b(){
    //Instructions...
  }
  private function methodeInstance_c(){
    //Instructions...
  }
  public static function methodeClasse_a(){
    //Instructions...
  }
  protected static function methodeClasse_b(){
    //Instructions...
  }
  private static function methodeClasse_c(){
    //Instructions...
  }
}
```

Sans aucun modificateur d'accès, la méthode sera publique.

15.4.1 / Les paramètres

Les méthodes peuvent accepter des données provenant de l'extérieur par l'intermédiaire des paramètres, lesquels sont listés entre des parenthèses suivant l'identificateur de la méthode.

```
public function uneMethode([Liste_Parametres]){
   //Instructions...
}
```

Les paramètres sont séparés par des virgules et énoncent pour chacun d'eux un identificateur et éventuellement un type et une valeur par défaut.

```
public function uneMethode(NomClasse $param, $param2, $param3 = 'valeur'){
  //Instructions...
}
```

Le typage des paramètres et équivalent à l'utilisation de l'opérateur *instanceof*.

```
public function uneMethode(
                          NomClasse $param1,
                          $param2,
                          $param3 = 'valeur',
                          array $param4){
  if($param1 instanceof NomClasse
  && $param4 instanceof array){
    //Instructions...
  }
}
```

Le typage des paramètres ne fonctionne qu'avec des objets et des tableaux. Les types scalaires (valeurs numériques et chaînes de caractères) ne sont pas admis.

Par défaut, les arguments sont passés à la méthode par valeur, c'est-à-dire que la fonction dispose d'une copie de la variable. Dans ce cas, la variable passée ne subira pas de modification, même si sa copie a été modifiée dans la méthode.

Le passage d'arguments par référence entraine une modification de la variable passée, si cette dernière a été changée dans la méthode. Pour obtenir ce type de passage d'arguments, il suffit de préfixer le paramètre par un caractère *&*.

```
class Classe {
  public function passerValeur($param){
    $param .= ' !';
    echo $param;
  }
  public function passerReference(&$param){
    $param .= ' ?';
    echo $param;
  }
}
$valeur = 'bonjour';
$obj = new Classe();
$obj->passerValeur($valeur);
echo $valeur;
$obj->passerReference($valeur);
echo $valeur;
```

15.4.2 / Le typage des paramètres

PHP 5 autorise le typage des paramètres d'une méthode et fournit donc la possibilité d'indiquer le type d'objet permis pour le passage d'arguments.

```
public function methode(NomClasse arg){
   //...
}
```

En fait, cette écriture ne correspond pas à un typage réel, à l'instar des langages tels que Java, mais plutôt un moyen de vérifier l'instance de l'objet passé en argument au moment de l'exécution.

```
public function methode(NomClasse $arg){
   //...
}
//correspond à
public function methode($arg){
    if(!($arg instanceof NomClasse)) {
        die('L\'argument doit être une instance de NomClasse');
    }
   //...
}
```

15.4.3 / Les valeurs de retour

Le retour de valeurs est obtenu en employant l'instruction *return*.

```
public function multiplier($opG, $opD){
  return $opG * $opD;
}
```

Il est également possible de **retourner des références** en préfixant le nom de la fonction avec un et commercial *&*.

```
class Classe {
  public $i;
  public function __construct($nb = 0){
    $this->i = $nb;
  }
  public function &incrementer(){
    $this->i++;
    return $this->i;
  }
}
$obj = new Classe(0);
$valeur = &$obj->incrementer();
echo $valeur . ' = ';
echo $obj->i . '; ';
for($i = 0; $i < 10; $i++){
  $obj->incrementer();
  echo $valeur . ' = ';
  echo $obj->i . '; ';
}
```

15.4.4 / L'invocation des méthodes

Les méthodes peuvent être invoquer de manière différente selon qu'elles sont soit des méthodes de classe, soit des méthodes d'instance.

L'invocation des méthodes d'instance s'effectue via une instance de classe. Au sein de leurs classes, les méthodes sont appelées avec la pseudo variable *$this*. A l'extérieur, il faut instancier la classe d'accueil et les appeler via l'objet ainsi créé.

```
class Classe {
  public function uneMethode(){
    $this->uneAutreMethode();
    //Instructions...
  }
  private function uneAutreMethode(){
    //Instructions...
  }
}
$obj = new Classe();
$obj->uneMethode();
```

Les méthodes de classe sont appelées à l'intérieur de leurs classes de déclaration par l'opérateur de résolution de portée *::* associé au mot-clé *self* (leur propre classe) ou *parent* (les classes dérivées). A l'extérieur, le nom de la classe associé à l'opérateur *::* permet d'appeler les méthodes de classe.

```
class Classe {
  public static function uneMethode(){
    self::uneAutreMethode();
    //Instructions...
  }
  private static function uneAutreMethode(){
    //Instructions...
  }
}
class SousClasse extends Classe {
  public static uneMethodeStatique(){
    parent::uneAutreMethode();
    //Instructions...
  }
}
Classe::uneMethode();
SousClasse::uneMethodeStatique();
```

15.4.5 / La surcharge des méthodes

La surcharge des méthodes peut s'opérer par l'intermédiaire de la méthode *__call()*. Cette surcharge consiste à utiliser un nom de méthode identique pour des appels de méthodes avec un nombre variable d'arguments.

```
//Appel d'une méthode avec des signatures différentes
$objet = new Objet();
$objet->calcul(10);
$objet->calcul(10, 2);
$objet->calcul(10, 2, puissance);
```

La surcharge des méthodes reflète un concept de la programmation objet. Il s'agît du polymorphisme, soit la capacité d'un objet à s'adapter aux différentes invocations possibles de méthodes avec un nom commun mais une liste d'arguments différents (signature).

Les méthodes invoquées à partir de l'objet n'existe pas dans la classe sous-jacente. La méthode *__call()* est chargée d'interpréter ces appels de méthodes.

```
class Objet {
  function __call($nom_methode, $arguments){
    //Gestion des méthodes n'existant pas dans la classe...
  }
}
```

Le paramètre *$nom_methode* contient le nom de la méthode qui est invoquée par l'objet. Tandis que *$arguments* contient la liste des arguments passés à la méthode précitée.

```
function __call($nom_methode, $arguments){
  //$nom_methode = 'calcul';
  //$arguments = array(10, 2, 'puissance');
}
```

Si les méthodes existent au sein de la classe, alors la fonction *__call* n'aura aucun effet.

```
class Personne {
  private $civilite;
  private $nom;
  private $prenom;
  private $adresse;
  private $telephone;

  public function __call($nom, $args) {
    switch ($nom) {
      case 'fixer':
        $nom = (sizeof($args) == 3) ? 'fixer_identite' : 'fixer_informations';
        break;
      default:
        die('La fonction Personne::'. $nom .'() n\'est pas définie.');
    }
    return call_user_func_array(array(&$this, $nom), $args);
  }

  public function fixer_informations() {
    $nb = func_num_args();
    if ($nb < 2) {
      die('Le nombre d\'arguments fourni n\'est pas correct.');
    }
    else if($nb == 2){
      for($i = 0; $i < $nb; $i++){
        switch ($i) {
          case 0 :
            $this->adresse = func_get_arg($i);
            break;
          case 1 :
            $this->telephone = func_get_arg($i);
            break;
        }
      }
    }
    else {
```

```
        for($i = 0; $i < $nb; $i++){
          switch ($i) {
            case 0 :
              $this->civilite = func_get_arg($i);
              break;
            case 1 :
              $this->nom = func_get_arg($i);
              break;
            case 2 :
              $this->prenom = func_get_arg($i);
              break;
            case 3 :
              $this->adresse = func_get_arg($i);
              break;
            case 4 :
              $this->telephone = func_get_arg($i);
              break;
          }
        }
        return $nb;
      }
  }

  public function fixer_identite($civilite, $nom, $prenom){
    $this->civilite = $civilite;
    $this->nom = $nom;
    $this->prenom = $prenom;
  }
}

$personne_i = new Personne;

$personne_i->fixer('Monsieur', 'Hussein', 'Hanoun');
var_dump($personne_i);
print('\n');

$personne_i->fixer('Bagdad Irak', '+49-89-8493070');
var_dump($personne_i);
print('\n');

$personne_f = new Personne;
$personne_f->fixer('Madame', 'Aubenas', 'Florence', '2a rue des Rosiers 75010 Paris', '0170204512');
var_dump($personne_f);
print('\n');
```

15.5 / Les modificateurs

Les modificateurs permettent d'affecter un comportement spécifique aux membres d'une classe.

Il existe deux catégories de modificateurs :

- les modificateurs d'accès : définissant le mode d'accès pour les éléments d'un programme PHP,
- les autres modificateurs : indiquant certaines particularités pour des éléments.

15.5.1 / Les modificateurs d'accès

Les modificateurs d'accès fournissent le mode d'accessibilité (ou de visibilité) des membres d'une classe.

Il existe trois types d'accessibilité en PHP.

- **public** : les membres publiques sont **visibles sans restriction** dans tout le programme,
- **protected** : les membres protégés ne peuvent être **accédés que par la classe qui les a déclarés et par les classes qui héritent de cette dernière**,
- **private** : les membres privés ne sont **visibles que dans la classe qui les a déclarés**.

Les attributs d'une classe ne peuvent être **accédés dans la hiérarchie de classes et en dehors de celles-ci**, qu'en utilisant respectivement **la référence à l'objet en cours *$this*** et qu'à partir d'**une instance de classe**.

```
class Classe {
  public $attribut;
  public function Classe($valeur){
    //$attribut = $valeur; //ERREUR !!!
    $this->attribut = $valeur;
  }
}
//echo $attribut; //ERREUR !!! //echo Classe->$attribut; //ERREUR !!!
$o = new Classe(10); //Instanciation d'une classe
echo $o->$attribut();
```

Une classe peut être complétement encapsulée **en déclarant privé tous ses attributs**, mais en permetant à des méthodes dites **accesseurs et mutateurs** de respectivement **accéder et modifier les attributs privés**, créant ainsi une interface publique pour leur accés et leur modification.

```
class Classe {
  private $attribut;
  public function Classe(){
    $this->attribut = null;
  }
  //Accesseur
  public function getAttribut(){
    return $this->attribut;
  }
  //Mutateur
  public function setAttribut($valeur){
    $this->attribut = $valeur;
  }
}
$o = new Classe(); //Instanciation d'une classe
$o->setAttribut(10);
echo $o->getAttribut();
```

Dans la mesure du possible, tous les attributs d'une classe doivent être privés afin de respecter les principes de l'encapsulation.

Les membres statiques obéissent aux même règles d'accessibilité que les membres d'instance, à l'exception de l'accès par un objet, qui dans leur cas, **utilisent le nom de la classe où ils ont été déclarés**. Au sein de leur classe, les membres statiques emploient l'opérateur de résolution de portée *::* avec le mot-clé *self*, au sein d'une classe étendue, il suffit de modifier le mot-clé par *parent*.

```
class Classe {
  public static $membre_statique = 16;
  public static function autreMembre(){
    return sqrt(self::$membre_statique);
  }
}
class SousClasse extends Classe{
  private static $membre_statique_2 = 2;
  public static function unAutreMembre(){
```

```
    return pow(parent::$membre_statique,
            self::$membre_statique_2);
  }
}

$o = new Classe();
//echo $o->membre_statique; //ERREUR!!! //echo $o->autreMembre(); //ERREUR!!!
echo Classe::$membre_statique;
echo Classe::autreMembre();

//echo SousClasse->$membre_statique_2; //ERREUR!!! echo
SousClasse::unAutreMembre();
```

Les constantes nécessitent une méthode d'accès identiques à ceux des membres de classe, hormis qu'ils n'ont pas besoin de modificateurs d'accès et disposent d'une accessibilité publique.

```
class SuperClasse { private $prive; protected $protege; public $public; //$sans = 'Sans
modificateur'; //ERREUR !!! public function __construct(){ $this->prive = 'Attribut privé !';
$this->protege = 'Attribut protégé !'; $this->public = 'Attribut public !'; } public function
__toString(){ return '<p>[' . $this->prive . ', ' . $this->protege . ', ' . $this->public . ']</p>'; }
protected function representer($separateur, $finligne){ return
$this->concatener($separateur) . $finligne; } private function concatener($separateur){
return $this->prive . $separateur . $this->protege . $separateur . $this->public; }
//Mutateur public function setPrive($valeur){ $this->prive = $valeur; } //Accesseur public
function getPrive(){ return $this->prive; } } class ClasseDerivee extends SuperClasse { const
CRLF = '<br/>'; private $separateur; public function __construct($separateur){ //$this->prive
= 'Attribut privé !'; //ERREUR !!! $this->protege = 'Attribut protégé défini dans la classe
dérivée !'; $this->public = 'Attribut public défini dans la classe dérivée !'; $this->separateur
= $separateur; } public function __toString(){ return parent::__toString(); } public
function apercu(){ return $this->representer($this->separateur, self::CRLF);
//SuperClasse->concatener(string sep) est inaccessible ! //return
$this->concatener($this->separateur) . self::CRLF;
  }
}

class UneAutreClasse {
  public function afficher(){
    $sc = new SuperClasse(); //Appel implicite à SuperClasse->__construct()
    echo $sc->publique . ClasseDerivee::CRLF;
  }
}

$superclasse = new SuperClasse(); //Appel implicite à __construct()
echo $superclasse->public;
echo ClasseDerivee::CRLF;
//echo $superclasse->protege; //ERREUR !!! //echo $superclasse->prive; //ERREUR !!!

echo $superclasse; //Appel implicite à __toString()

$superclasse->setPrive('Attribut privé modifié en dehors de la classe !');
echo $superclasse->getPrive();
echo ClasseDerivee::CRLF;

echo $superclasse; //Appel implicite à __toString()
//echo $superclasse->representer(';', '\n'); //ERREUR !!! //echo
$superclasse->concatener('\t') . '\n'; //ERREUR !!!

$classederivee = new ClasseDerivee(); //Appel implicite à __construct()
//echo $classederivee->separateur; //ERREUR !!!
echo $classederivee; //Appel implicite à __toString()
echo $classederivee->apercu();

$uneautreclasse = new UneAutreClasse();
$uneautreclasse->afficher();
```

15.5.2 / Les modificateur *static*

Au sein d'une classe, les champs ou les méthodes déclarées statiques peuvent être appelés, même si la classe sous-jacente n'a pas été instanciée.

Ainsi, il suffit d'utiliser le nom de la classe contenant les membres statiques, avec l'opérateur de résolution de portée *::* pour pouvoir les appeler dans un programme.

```
class Calcul {
  public static $valeur = 10;
  public static function multiplier($multiplicateur){
    echo $multiplicateur . ' * ' . self::$valeur;
    if(is_numeric($multiplicateur)
    && is_numeric(self::$valeur))
      return self::$valeur * $multiplicateur;
    else
      return 0;
  }
}
//Appel des membres statiques
Calcul::$valeur = 12;
$res = Calcul::multiplier(10);
echo ' = ' . $res;
```

Au sein des classes, les membres statiques doivent être appelés par l'opérateur de résolution, précédé du mot clé *self* ou *parent* signifiant respectivement , la classe courante ou la classe parente dans le cadre d'un héritage (*extends*).

```
class Noeud {
  public static $id = 'root';
}

class Element extends Noeud {
  public static function getID(){
    return parent::$id;
  }
}
```

La pseudo variable *$this* associée à l'opérateur ->, ne sont pas utilisable pour accéder aux membres statiques, *$this->* n'étant indiqué que pour l'appel de membres d'instances, c'est-à-dire des objets.

```
$this->UnChampStatique; //ERREUR !
self::$UnChampStatique; //CORRECT !
```

Le modificateur *static* doit être placé avant la déclaration de visibilité, en l'occurrence *public*, *protected* ou *private*. Par défaut de modificateur de visibilité, les membres statiques sont déclarés implicitement publics.

```
static UnChamp = 'x';
//Equivaut à
public static UnChamp = 'x';
```

15.5.3 / Le modificateur *final*

Le modificateur *final* est utilisé pour éviter une extension des classes ou un héritage des méthodes.

Lorsque le modificateur *final* apparaît **dans une déclaration de classe**, cela signifie que **la classe ne peut être étendue**.

```
final class NomClasse {
  //...
}
//ClasseEtendue ne peut hériter de NomClasse
class ClasseEtendue extends NomClasse {
  //...
}
```

Une méthode portant le modificateur *final* ne peut pas être remplacée dans une sous-classe.

```
class NomClasse {
  public final function methodeFinale() {
    //...
  }
  public function autreMethode() {
    //...
  }
}

class ClasseEtendue extends NomClasse {
  public function methode() {
    //methodeFinale() ne peut être héritée de NomClasse
    parent::methodeFinale();
    //autreMethode() peut être héritée de NomClasse
    parent::autreMethode();
  }
  //...
}
```

En outre, une méthode finale pourra être **optimisée par le compilateur**, puisqu'elle ne peut être sous-classée.

Par ailleurs, **les méthodes déclarées avec le modificateur *static* ou *private*** sont implicitement finales.

15.6 / L'opérateur *::*

L'opérateur de résolution de portée *::* permet :

- de **faire référence à une fonction définie dans une super-classe** à partir d'une classe héritant de cette dernière,
- d'**appeler des fonctions ou des variables statiques ou constantes, définies dans une classe avant l'instanciation de cette dernière** dans le corps du programme.

```
NomClasse::UnMembre;
```

Les mots-clé *self* et *parent* associés à l'opérateur *::*, permettent d'accéder respectivement aux membres de la classe en cours et de la classe parente (héritage).

```
parent::unMembre_ClasseParente;
self::unMembre_ClasseCourante;

class super_classe
{
  public const CONSTANTE = 10;
  public static variable = 'x';
  protected autre_variable;

  public function fonction()
  {
    echo "Blocs d'instructions de la fonction fonction()"
        . " dans la super-classe.";
  }
}
//Appel de la variable statique de la classe 'super_classe'
echo super_classe::$variable;

//Appel de la constante de la classe 'super_classe'
echo super_classe::CONSTANTE;

class nouvelle_classe extends super_classe
{
  public autre_variable;
  public function fonction()
  {
    //Appel de la fonction 'fonction' de la classe 'super_classe'
    parent::fonction();
    //Appel de la variable d'instance 'autre_variable' de la classe 'nouvelle_classe'
    echo self::$autre_variable;
    //Appel de la variable d'instance 'autre_variable' de la classe 'super_classe'
    echo parent::$autre_variable;
    echo "Blocs d'instructions de la fonction fonction()"
        . " dans la nouvelle-classe.";
  }
  public function autre_fonction()
  {
    //Appel de la constante 'CONSTANTE' de la classe 'super_classe'
    echo parent::CONSTANTE;
  }
}
```

15.6.1 / Les opérateurs *parent* et *self*

Les opérateurs *parent* et *self* permettent de faire référence à des membres d'une super-classe et d'une classe.

En fait, **l'opérateur *parent* correspond directement à la super-classe** d'une classe ayant héritée de cette dernière.

```
class super_classe
{
  function fonction()
  {
    echo "Blocs d'instructions de la fonction fonction()"
       . " dans la super-classe.";
  }
}

class nouvelle_classe extends super_classe
{
  function fonction()
  {
    echo "Blocs d'instructions de la fonction fonction()"
       . " dans la nouvelle-classe.";

  // se référe à la fonction fonction() de la super_classe
    parent::fonction();
  }
}

$objet = new nouvelle_classe;

$objet->fonction();

/* retourne :
'Blocs d'instructions de la fonction fonction()
dans la nouvelle-classe.' et
'Blocs d'instructions de la fonction fonction()
dans la super-classe.' */
```

L'opérateur *parent* se révèle particulièrement utile lorsqu'**une classe consiste en une spécialisation d'une classe générique**.

L'opérateur *self* désigne quant à lui la classe courante. Ainsi, les membres statiques ou constants de la classe courante peuvent être accédés sans ambiguités, notamment si cette classe est dérivée d'une autre.

```
class super_classe
{
  const CONSTANTE = "VALEUR CONSTANTE";
  static $variable = "valeur statique";
  function fonction()
  {
    echo 'Constante : ' self::CONSTANTE
       . '\nVariable de classe : ' . self::$variable;
  }
}
```

Dans le cas où une classe étendue redéfinit des méthodes de la super-classe, les méthodes de cette dernière ne seront pas invoquées par PHP. Il est donc nécessaire d'appeler les méthodes d'origines si besoin est, à partir des méthodes redéfinies au sein de la sous-classe.

De la même façon que précité, il faudra envisager ce genre d'invocation (avec *self* ou *parent*) lors de surcharges de méthodes d'une classe, de définitions des constructeurs et destructeurs et de la redéfinition des méthodes magiques (*__toString()*, *__sleep()*, *__wakeup()*, *__clone()*, *__construct()*, *__destruct()*, *__call()*, *__get()* et *__set()*).

Exemple [voir]

```
<html>
<body>
<?php
class Chaine {
  public $valeur;
  public $date;
  public function __construct(){
    $this->valeur = func_get_arg(0);
    $this->date = date(Personne::MASQUE_DATE);
  }
  public function set($valeur){
    $this->valeur = $valeur;
  }
  public function get(){
    return $this->valeur;
  }
  public function __toString(){
   return '[' . $this->valeur . ']';
  }
}
class Personne {
  const MASQUE_DATE = 'd M Y H:i:s';
  static $compteur;
  private $date_inscription;
  private $date_modification;
  private $id;
  private $civilite;
  private $nom;
  private $prenom;
  private $adresse;
  private $telephone;
  public function __construct(
                        $civilite,
                        $nom,
                        $prenom,
                        $adresse,
                        $telephone){
    $this->date_inscription = new Chaine(date(self::MASQUE_DATE));
    $this->civilite = new Chaine($civilite);
    $this->nom = new Chaine($nom);
    $this->prenom = new Chaine($prenom);
    $this->adresse = new Chaine($adresse);
    $this->telephone = new Chaine($telephone);
    $this->date_modification = new Chaine($this->date_inscription->get());
    $this->idModif();
  }

  public function __toString(){
   return '[' . $this->civilite->get() . ', '
            . $this->nom->get() . ', '
            . $this->prenom->get() . ', '
            . $this->adresse->get() . ', '
            . $this->telephone->get() . ', '
            . $this->date_inscription->get() . ', '
            . $this->date_modification->get() . ', '
            . $this->id->get() . ']';
  }
  public function setNom($nom){
    $this->nom->set($nom);
    $this->dateModif();
    $this->idModif();
  }
  public function setPrenom($prenom){
    $this->prenom->set($prenom);
    $this->dateModif();
    $this->idModif();
  }
  public function setAdresse($adresse){
    $this->adresse->set($adresse);
    $this->dateModif();
  }
  public function setTelephone($telephone){
    $this->telephone->set($telephone);
    $this->dateModif();
```

```
  }
  private function dateModif(){
    $this->date_modification->set(date(self::MASQUE_DATE));
  }
  private function idModif(){
    $this->id = new Chaine(self::$compteur++ . date('dmy')
                  . '.' . $this->nom->get()
                  . '_' . substr($this->prenom, 0, 1));
  }

  public function __clone(){
    $this->date_inscription = new Chaine(date(self::MASQUE_DATE));
    $this->civilite = new Chaine($this->civilite->get());
    $this->nom = new Chaine($this->nom->get());
    $this->prenom = new Chaine($this->prenom->get());
    $this->adresse = new Chaine($this->adresse->get());
    $this->telephone = new Chaine($this->telephone->get());
    $this->date_modification = new Chaine($this->date_inscription->get());
    $this->idModif();
  }
}
class Employe extends Personne {
  public $specialite;
  public $service;
  public $date_embauche;
  public function __construct($civilite,
                    $nom,
                    $prenom,
                    $adresse,
                    $telephone,
                    $specialite,
                    $service){
    parent::__construct($civilite,
                  $nom,
                  $prenom,
                  $adresse,
                  $telephone);
    $this->specialite = new Chaine($specialite);
    $this->service = new Chaine($service);
    $this->date_embauche = new Chaine(date(parent::MASQUE_DATE));
  }
  public function __toString(){
    return '[' . parent::__toString() .', '
          . $this->specialite->get() . ', '
          . $this->service->get() . ', '
          . $this->date_embauche->get() . ']';
  }
}
$personne = new Personne(
                'Madame',
                'Waldec',
                'Anne',
                '75 rue Belleville 75010 Paris',
                '0142152432');

$employe = new Employe(
                'Madame',
                'Waldec',
                'Anne',
                '75 rue Belleville 75010 Paris',
                '0142152432',
                'Analyste/programmeur',
                'Recherche et développement');

echo 'Personne : ';
echo $personne;
echo '<br/>';
echo 'Employée : ';
echo $employe;
?>
</body>
</html>
```

15.7 / Les attributs

Les classes comportent différents membres dont des variables définies à leur sommet.

```
class Classe
  //Défintion des variables
  private $v1;
  public static v2;
  //...
}
```

Les champs peuvent porter différents modificateurs.

- **Les modificateurs d'accès** (*private*, *protected* et *public*) indiquant la visibilité du membre au sein du programme.
- **Le modificateur *static*** déclare que les membres peuvent être exploités sans l'instanciation de leur classe.

Les champs doivent obligatoirement possèder un modificateur d'accès. A défaut, une erreur est lancée par PHP.

15.7.1 / Les attributs

Les champs qui ne portent pas le modificateur *static* sont dénommés des attributs.

Les attributs sont des éléments variables d'un objet. Ils possèdent chacun un nom, un type implicite, ainsi qu'une valeur qui constitue l'état d'un objet.

```
class Classe {
  public $attribut;
  public function __construct($valeur){
    $this->attribut;
  }
}

$o1 = new Classe(10);
$o2 = new Classe(2);
$o3 = new Classe(5);
$o4 = new Classe(0);
$o4->attribut = $o2->attribut * $o3->attribut;
```

Chaque instance d'une même classe peut possèder un état différent, ceux sont les valeurs de l'ensemble des attributs de ces objets qui sont susceptibles d'avoir été initialisées, puis exploités différement.

Les attributs sont accessibles dans toutes les méthodes de cette classe. Toutefois, il faut utiliser la pseudo variable *$this*, qui fait référence à l'objet courant, pour appeler les champs.

```
class Classe {
  public $attribut;

  public function uneMethode(){
    $this->attribut = 'Une valeur';
  }
}
```

L'accès aux champs, depuis l'extérieur de leur classe, doit passer obligatoirement par l'instanciation de ces classes.

En outre, **la visibilité à l'extérieur de la classe dépend du modificateur d'accès** spécifié.

```
class Classe {
  public $attribut;
  private $atribut_2;
}

$obj = new Classe();
$obj->attribut = 'Une valeur';
echo $obj->attribut;

//$obj->attribut_2 = 'Une valeur'; //ERREUR !!! //echo $obj->attribut_2; //ERREUR !!!
```

15.7.2 / La surcharge des attributs

L'accès aux champs de l'objet peuvent s'effectuer via les fonctions *__get()* et *__set()*.

```
private $champs = array('nom_champ1'=> 'valeur', ..., 'nom_champN' => 'valeur');

function __get($nom_champ){
  //retourne une valeur
}

function __set($nom_champ, $valeur){
  //fixe la valeur d'un champ
}
```

Les fonctions *__get()* et *__set()* possèdent un paramètre commun, en l'occurrence le nom du champ invoqué par le biais d'un objet. Le second paramètre de *__set()* est la valeur qui devra être affectée au champ déterminé par son nom. La fonction *__get()* a la charge de retourner la valeur de ce champ.

Le tableau associatif est chargé de contenir l'ensemble des champs qui pourront être accédés ou fixés par les fonctions de surcharge.

Ce concept de programmation objet s'appele l'encapsulation, c'est-à-dire la capacité d'une classe à masquer les méthodes et les données internes (son implémentation) qu'il n'est pas nécessaire de rendre visible. En conséquence, les champs publics n'ont aucun intérêt à utiliser cette technique, seuls ceux ayant une visibilité limité (*private* ou *protected*) sont concernés.

```
class Coordonnees {
  private $point = array('x' => 1, 'y' => 2, 'z' => 3);

  function __get($nom) {
    if (isset($this->point[$nom])) {
      $val = $this->point[$nom];
      return $val;
    }
    else {
      return Null;
    }
  }

  function __set($nm, $val) {
    if (isset($this->point[$nom])) {
      $this->point[$nom] = $val;
    }
  }
}

$coord = new Coordonnees();
//Appels implicites de __set()
$coord->x = 10;
$coord->y = 100;
$coord->z = 15;

//Appels implicites de __get()
print('x = ' . $coord->x);
print('y = ' . $coord->y);
print('z = ' . $coord->z);
```

15.7.3 / La portée des variables

La portée détermine le partie d'un programme, dans laquelle une variable peut être utilisée.

La portée des variables diffère selon leur emplacement au sein du code PHP. En fait, lorsqu'une variable est déclarée au sein d'un bloc de code comme une classe ou une méthode, elle ne peut être utilisée que dans ce même bloc.

```
class classe_comptage {
  /* i est utilisable dans toute la classe y compris dans ses méthodes */
  $i = 0;

  public function compteur(){
  // pas est seulement utilisable
// dans la méthode compteur
    $pas = 5
    while($this->i <= 100){
      echo $this->i . 'rn';
      $this->i += 5;
    }
  }
}
```

Les variables définies dans la classe, appelées champs ou attributs, sont accessibles dans toutes les méthodes de cette classe. Toutefois, il faut utiliser le mot-clé *$this*, qui fait référence à l'objet courant, pour appeler les attributs.

```
$this->variable;
```

Par défaut, **les variables possèdent une portée essentiellement locale** dans dans leurs blocs respectifs délimités par des accolades.

```
{
  $x = 1;
// seul x est accessible à cet endroit
  {
// x est accessible à cet endroit,
// y n'est pas accessible puisque la
// variable n'est pas encore déclarée
    $y = 2;
// x et y sont accessibles
  }
// y n'est pas accessible,
// x est accessible à cet endroit
}
```

En se fondant sur cette règle, **une variable définie dans une classe** peut être accèdée dans toute la classe et une autre **définie dans une méthode** restera accessible dans cette seule méthode.

```
class calcul {
  $a = 10;
  public afficher(){
    $b = 4;
    echo"resultat : " + $this->a * b);
  }
}
```

Il est possible de **déterminer la portée des variables** plus précisément, par le truchement d'un modificateur d'accès.

Les variables peuvent avoir une portée globale, soit une accessibilité entre toutes les classes, à condition d'utiliser le modificateur d'accès *public* lors de leur déclaration.

```
class affichage {
  public $texte = 'Un texte...';

  public function affiche(){
    echo $this->texte;
```

```
}

class modification {
  public function ajout(){
    $this->texte .= 'Un second paragraphe...';
  }
}
```

La portée entre classe peut être limitée aux seules sous-classes, en déclarant une variable avec le modificateur d'accès *protected*.

Les variables déclarées avec le modificateur d'accès *private* ne peuvent être accédées que dans la classe elle-même.

```
private string chaine = "Une chaîne de caractères...";
```

Un programme reposant sur des variables globales peut être à l'origine de bogues difficiles à détecter. C'est pourquoi, il est préférable d'**utiliser des variables locales** offrant plus de sûreté avec une durée de vie limitée.

Exemple

```
class calcul{
  private $a = 1;
  public function addition(){
    $b = 1;
    $b += $this->a;
    // $this->a = 2
  }
  public function soustraction(){
    $c = 1;
    $c -= $this->a + $b;
/* provoque une erreur car b n'est accessible que dans addition() */
  }
}
```

15.8 / Les constantes

Les constantes contiennent une valeur fixe et sont caractérisées par un identficateur et un type de données à l'instar des variables.

Les constantes ne peuvent être déclarées que dans les classes, en utilisant le mot-clé *const*. Les méthodes ne peuvent contenir des déclaration de constantes.

```
const COULEUR_NOIRE = '#000000';

const CODE = 1537503;

const PI = 3.14;
```

Les constantes ne sont pas précédées par le signe *$* spécifique aux variables.

Par convention, **le nom des constantes est toujours en majuscules** afin de les distinguer sans équivoques des variables.

```
class Surface {
  const PI = 3.14;

  public function calculerAireCercle($rayon) {
   return self::PI * pow($rayon, 2);
  }
}
$rayon = 8;
$surface = new Surface();
$res = $surface->calculerAireCercle($rayon);
echo 'PI = ' . Surface::PI . ' Rayon = ' . $rayon . ' Surface = ' . $res;
```

A l'instar des membres statiques, il n'est pas possible d'accéder aux constantes depuis une instance de classe, c'est-à-dire que *$this->CONSTANTE* n'est pas autorisé.

Exemple [voir]

```
<html>
<body>
<?php
// Un object de gestion des chaines de caractères
class Chaine {
  public $valeur;
  private $date;
  public function __construct(){
    $this->valeur = func_get_arg(0);
    $this->date = date(Personne::MASQUE_DATE);
  }
  public function set($valeur){
    $this->valeur = $valeur;
  }
  public function get(){
    return $this->valeur;
  }
  public function __toString(){
   return '[' . $this->valeur . ']';
  }
}
class Personne {
  const MASQUE_DATE = 'd M Y H:i:s';
  static $compteur;
  private $date_inscription;
  private $date_modification;
  private $id;
  private $civilite;
  private $nom;
  private $prenom;
  private $adresse;
  private $telephone;
  public function __construct(
                        $civilite,
                        $nom,
                        $prenom,
                        $adresse,
                        $telephone){
    $this->date_inscription = new Chaine(date(self::MASQUE_DATE));
    $this->civilite = new Chaine($civilite);
    $this->nom = new Chaine($nom);
    $this->prenom = new Chaine($prenom);
    $this->adresse = new Chaine($adresse);
    $this->telephone = new Chaine($telephone);
    $this->date_modification = new Chaine($this->date_inscription->get());
    $this->id = $this->idModif();
  }

  public function __toString(){
   return '[' . $this->civilite->get() . ', '
            . $this->nom->get() . ', '
            . $this->prenom->get() . ', '
            . $this->adresse->get() . ', '
            . $this->telephone->get() . ', '
            . $this->date_inscription->get() . ', '
            . $this->date_modification->get() . ','
            . $this->id->get() . ']';
  }
  public function setNom($nom){
    $this->nom->set($nom);
    $this->dateModif();
    $this->idModif();
  }
  public function setPrenom($prenom){
    $this->prenom->set($prenom);
    $this->dateModif();
    $this->idModif();
  }
  public function setAdresse($adresse){
    $this->adresse->set($adresse);
    $this->dateModif();
  }
  public function setTelephone($telephone){
    $this->telephone->set($telephone);
```

```
    $this->dateModif();
  }
  private function dateModif(){
    $this->date_modification->set(date(self::MASQUE_DATE));
  }
  private function idModif(){
    $this->id = new Chaine(self::$compteur++ . date('dmy')
                . '.' . $this->nom->get()
                . '_' . substr($this->prenom, 0, 1));
  }

  public function __clone(){
    $this->date_inscription = new Chaine(date(self::MASQUE_DATE));
    $this->civilite = new Chaine($this->civilite->get());
    $this->nom = new Chaine($this->nom->get());
    $this->prenom = new Chaine($this->prenom->get());
    $this->adresse = new Chaine($this->adresse->get());
    $this->telephone = new Chaine($this->telephone->get());
    $this->date_modification = new Chaine($this->date_inscription->get());
    $this->idModif();
  }
}
class Employe extends Personne {
  public $specialite;
  public $service;
  public $date_embauche;
  public function __construct($civilite,
                      $nom,
                      $prenom,
                      $adresse,
                      $telephone,
                      $specialite,
                      $service){
echo $specialite . ' ' . $service;
    $this->specialite = new Chaine($specialite);
    $this->service = new Chaine($service);
    $this->date_embauche = new Chaine(date(parent::MASQUE_DATE));
    parent::__construct($civilite,
                $nom,
                $prenom,
                $adresse,
                $telephone);
  }
  public function __toString(){
    return '[' . parent::__toString() .', '
           . $this->specialite->get() . ', '
           . $this->service->get() . ', '
           . $this->date_embauche->get() . ']';
  }

$employe = new Employe(
                'Madame',
                'Waldec',
                'Anne',
                '75 rue Belleville 75010 Paris',
                '0142152432',
                'Analyste/programmeur',
                'Recherche et développement');

echo $employe;
?>
</body>
</html>
```

15.9 / Le clonage

Le clonage d'objets consiste à copier un objet source vers un objet cible.

Mais ce clonage peut être accompli superficiellement, c'est-à-dire que la référence de l'objet source est transmise à l'objet cible. Cela entraîne que les deux objets pointent vers le même contenu.

```
//Copie superficielle
objet A
  référence_A --> contenu_A
objet B
  référence_A --> contenu_A
```

Dans ce cas, les modifications apportées dans l'objet A seront répercutées dans l'objet B.

```
Objet A = Jean
Clonage de Objet A vers Objet B
Objet B ( = Jean)
//Réaffectation
Objet A = Pierre
//Etat de l'objet cible
Objet B ( = Pierre)
```

Un autre type de clonage est possible, il s'agit de copier l'état d'un objet source vers un autre objet.

```
//Copie profonde
objet A
  référence_A --> contenu_A
objet B
  référence_B --> contenu_A
```

De cette manière, la modification de l'un des objets impliqués dans le clonage, ne se répercute pas sur l'autre objet.

```
Objet A = Jean
Clonage de Objet A vers Objet B
Objet B ( = Jean)
//Réaffectation
Objet A = Pierre
//Etat de l'objet cible
Objet B ( = Jean)
```

Par défaut, PHP 5 fait une copie systématique des propriétés de l'objet source, vers l'objet cible. Ceci a pour conséquence un clonage profond des objets, soit une copie des valeurs des propriétés de l'objet source vers l'objet cible, tant que les propriétés de l'objet source ne sont pas des objets. En effet, dans le cas des propriétés instances d'une classe quelconque, la copie est alors superficielle, c'est à dire une transmission des références d'objet des propriétés vers l'objet cible.

```
Objet A = Pierre, Dupont
Clonage de Objet A vers Objet B
Objet B ( = Pierre, Dupont)
//Réaffectation
Objet A = Jean, Dupont
//Etat de l'objet cible
Objet B ( = Pierre, Dupont)

Objet Prenom = Pierre
Objet Nom = Dupont
Objet A = Objet Prenom, Objet Nom
Clonage de Objet A vers Objet B
Objet B ( = Objet Prenom ( = Pierre), Objet Nom ( = Dupont))
//Réaffectation de l'objet Prenom
Objet A -> Objet Prenom = Jean
//Etat de l'objet cible
Objet B ( = Objet Prenom ( = Jean), Objet Nom ( = Dupont))
```

- Si les propriétés sont des objets, une copie de référence est exécutée.

- Si les propriétés sont des chaînes de caractères ou des valeurs alphanumériques, une copie de valeur est accomplie.

L'appel de la fonction *clone* produit le clonage d'un objet source vers un objet cible dans les conditions précitées.

```
$Objet_Cible = clone $Objet_Source;
```

La redéfinition de la méthode *__clone()* permet d'obtenir un comportement plus approprié pour le clonage d'un objet source. Il devient possible d'obtenir un objet cible indépendant de l'objet source, mais en conservant l'état et le comportement de ce dernier. Avec ce type de clonage, la modification de l'état de l'un des objets, n'entraîne pas de changement dans l'autre.

```
class UneClasse {
  //...
  public function __clone(){
    //Bloc de code
  }
}
```

Chaque objet possède une méthode *__clone()* par défaut, qui effectue une copie superficielle de toutes les propriétés de l'objet source vrs l'objet cible.

Exemple [voir]

```
//Clonage simple
<html>
<body>
<?php
class Personne {
  static $compteur;
  private $date_inscription;
  private $date_modification;
  private $id;
  private $civilite;
  private $nom;
  private $prenom;
  private $adresse;
  private $telephone;
  public function __construct(
                        $civilite,
                        $nom,
                        $prenom,
                        $adresse,
                        $telephone){
    $this->date_inscription = date('d M Y H:i:s');
    $this->civilite = $civilite;
    $this->nom = $nom;
    $this->prenom = $prenom;
    $this->adresse = $adresse;
    $this->telephone = $telephone;
    $this->date_modification = $this->date_inscription;
  }

  public function __toString(){
   return '[' . $this->civilite . ', '
            . $this->nom . ', '
            . $this->prenom . ', '
            . $this->adresse . ', '
            . $this->telephone . ']';
  }
  public function setNom($nom){
    $this->nom = $nom;
    $this->dateModif();
  }
  public function setPrenom($prenom){
    $this->prenom = $prenom;
    $this->dateModif();
  }
  public function setAdresse($adresse){
    $this->adresse = $adresse;
    $this->dateModif();
  }
  public function setTelephone($telephone){
    $this->telephone = $telephone;
    $this->dateModif();
  }
  private function dateModif(){
    $this->date_modification = date('d M Y H:i:s');
  }
}

$personne = new Personne(
                'Madame',
                'Waldec',
                'Anne',
                '75 rue Belleville 75010 Paris',
                '0142152432');

sleep(1);
$clone = clone $personne;
sleep(2);
$personne->setNom('Walewski');

echo 'Original : ';
echo $personne;
echo '<br/>';
echo 'Clone    : ';
echo $clone;
```

```
/*
Original : [Madame, Walewski, Anne, 75 rue Belleville 75010 Paris, 0142152432]
Clone    : [Madame, Waldec, Anne, 75 rue Belleville 75010 Paris, 0142152432]
*/
echo '<br/>';
echo '<br/>';
print_r($personne);
print_r($clone);
?>
</body>
</html>

//Clonage avec redéfinition de la méthode __clone()
<html>
<body>
<?php
// Un object de gestion des chaines de caractères
class Chaine {
  public $valeur;
  public function __construct(){
    $this->valeur = func_get_arg(0);
  }
  public function set($valeur){
    $this->valeur = $valeur;
  }
  public function get(){
    return $this->valeur;
  }
  public function __toString(){
   return '[' . $this->valeur . ']';
  }
}
class Personne {
  static $compteur;
  private $date_inscription;
  private $date_modification;
  private $id;
  private $civilite;
  private $nom;
  private $prenom;
  private $adresse;
  private $telephone;
  public function __construct(
                        $civilite,
                        $nom,
                        $prenom,
                        $adresse,
                        $telephone){
    $this->date_inscription = new Chaine(date('d M Y H:i:s'));
    $this->civilite = new Chaine($civilite);
    $this->nom = new Chaine($nom);
    $this->prenom = new Chaine($prenom);
    $this->adresse = new Chaine($adresse);
    $this->telephone = new Chaine($telephone);
    $this->date_modification = new Chaine($this->date_inscription->get());
  }

  public function __toString(){
   return '[' . $this->civilite->get() . ', '
            . $this->nom->get() . ', '
            . $this->prenom->get() . ', '
            . $this->adresse->get() . ', '
            . $this->telephone->get() . ', '
            . $this->date_inscription->get() . ', '
            . $this->date_modification->get() . ']';
  }
  public function setNom($nom){
    $this->nom->set($nom);
    $this->dateModif();
  }
  public function setPrenom($prenom){
    $this->prenom->set($prenom);
    $this->dateModif();
  }
  public function setAdresse($adresse){
```

```
    $this->adresse->set($adresse);
    $this->dateModif();
  }
  public function setTelephone($telephone){
    $this->telephone->set($telephone);
    $this->dateModif();
  }
  private function dateModif(){
    $this->date_modification->set(date('d M Y H:i:s'));
  }

  public function __clone(){
    $this->date_inscription = new Chaine(date('d M Y H:i:s'));
    $this->civilite = new Chaine($this->civilite->get());
    $this->nom = new Chaine($this->nom->get());
    $this->prenom = new Chaine($this->prenom->get());
    $this->adresse = new Chaine($this->adresse->get());
    $this->telephone = new Chaine($this->telephone->get());
    $this->date_modification = new Chaine($this->date_inscription->get());
  }
}

$personne = new Personne(
                'Madame',
                'Waldec',
                'Anne',
                '75 rue Belleville 75010 Paris',
                '0142152432');

sleep(1);
$clone = clone $personne;
sleep(2);
$personne->setNom('Walewski');

echo 'Original : ';
echo $personne;
echo '<br/>';
echo 'Clone    : ';
echo $clone;
/*
Original : [Madame, Walewski, Anne, 75 rue Belleville 75010 Paris,
          0142152432, 26/Jul/2005 10:08:13, 26/Jul/2005 10:08:16]
Clone    : [Madame, Waldec, Anne, 75 rue Belleville 75010 Paris,
          0142152432, 26/Jul/2005 10:08:14, 26/Jul/2005 10:08:14]

*/
echo '<br/>';
echo '<br/>';
print_r($personne);
echo '<br/>';
print_r($clone);
?>
</body>
</html>
```

Les invocations de la fonction *sleep()* ont pour but d'obtenir une différence de date entre l'instanciation de l'objet source et le clonage de ce dernier ver l'objet cible.

15.10 / Sauvegarde des objets

La sauvegarde et la relecture des objets s'effectuent respectivement par l'intermédiaire de la fonction *serialize* et *unserialize*.

La fonction de sérialisaton permet de transformer un objet en une chaîne de caractères pouvant être facilement transmise à une autre page lors d'une session.

La fonction de désérialisation permet de reconstituer l'objet à partir de la chaîne de caractères précitées.

La fonction *session_register* facilite la transmission d'un objet durant une session en linéarisant et délinéarisant d'une façon automatique les objets.

Néanmoins dans les deux cas, **il est nécessaire d'inclure la définition de la classe instanciant l'objet** dans toutes les pages susceptibles de contenir cet objet lors d'une session. Si cela n'est pas fait, l'objet existera dans des documents PHP sans leurs classes et partant seront inopérants.

Exemple [voir]

```php
<?php
// Page de définition de classes trigo.inc

 class trigonometrie
 {
  var $AB;
  var $BC;
  var $AC;

  function hypothenuse()
  {
   $resultat = sqrt(pow($this->BC, 2) + pow($this->AC, 2));
   return number_format($resultat, 2, ',', ' ');
  }
 }
?>

<?php
// Première page : saisie.php

 // inclusion de la définition de classe
 include("trigo.inc");

 // crée une instance de l'objet
 $trigo = new trigonometrie;

 // sérialise l'objet
 $objet_chaine = serialize($trigo);

 // ouvre un fichier en écriture seule
 $fichier = fopen("fic", "w");

 // écrit l'objet linéarisé dans le fichier
 fputs($fichier, $objet_chaine);

 // ferme le fichier
 fclose($fichier);
?>
<form action="resultat.php" method="post">
 <table border="0">
  <tr>
   <th colspan="2">
    <h3>Calcul de l'hypothénuse d'un triangle rectangle</h3>
   </th>
  </tr>
  <tr>
   <td><u>longueur :</u></td>
   <td><input type="text" name="longueur" size="10" maxlength="10"></td>
  </tr>
  <tr>
   <td><u>hauteur :</u></td>
   <td><input type="text" name="hauteur" size="10" maxlength="10"></td>
  </tr>
  <tr>
   <th colspan="2"><input type="submit" value="Calculer"></th>
  </tr>
 </table>
</form>

<?php
// Seconde page : resultat.php

 // inclusion de la définition de classe
 include("trigo.inc");

 /* regroupe tous les éléments du tableau retourné par la fonction file dans une chaîne */
 $objet_chaine = implode("", file("fic"));

 // désérialise l'objet
 $trigo = unserialize($objet_chaine);

 // appelle deux propriétés et une méthode de l'objet
 $trigo->BC = $hauteur;
```

```
  $trigo->AC = $longueur;
?>
<table border="0">
  <tr>
    <th>
      <h3>Calcul de l'hypothénuse d'un triangle rectangle</h3>
    </th>
  </tr>
  <tr>
    <td>hauteur (BC)</td>
    <td>=</td>
    <td><?php echo $trigo->BC ?></td>
  </tr>
  <tr>
    <td>longueur (AC)</td>
    <td>=</td>
    <td><?php echo $trigo->AC ?></td>
  </tr>
  <tr>
    <td>hypothénuse (AB)</td>
    <td>=</td>
    <td><?php echo $trigo->hypothenuse() ?></td>
  </tr>
</table>
```

15.10.1 / Les fonctions *__sleep* et *__wakeup*

Les fonctions spéciales *__sleep* et *__wakeup* sont appelées respectivement par les commandes *serialize* et *unserialize* afin de traiter l'objet ou la chaîne de caractères représentant un objet avant la linéarisation ou délinéarisation.

```
class nom_classe
{
  function __sleep()
  {
    //Instructions à accomplir avant serialize()...
  }

  function __wakeup()
  {
    //Instructions à accomplir avant unserialize()...
  }
}
```

La fonction *serialize* recherche la méthode *__sleep* dans une classe afin de la lancer avant le processus de linéarisation.

Subséquemment, **la fonction spéciale *__sleep* peut effectuer un traitement prélimiaire de l'objet dans le but de terminer proprement toutes les opérations relatives à cet objet**, comme la fermeture des connexions sur des bases de données, par ailleurs il est possible également d'entreprendre une suppression des informations superflues ne nécessitant pas de sauvegarde ou encore une réorganisation plus judicieuse, etc..

De même, **la fonction *unserialize* recherche la méthode *__wakeup* dans une classe afin de la lancer avant le processus de délinéarisation.**

Ensuite, **la fonction *__wakeup* peut accomplir des opérations de reconstruction de l'objet** en ajoutant des informations, en réouvrant des connexions vers des bases de données, ou encore en initialisant des actions, etc..

Exemple [voir]

```php
<?php
 class semaine
 {
  var $erreur;
  var $jour;

  function semaine()
  {
   $this->jour = array("lundi", "mardi", "mercredi", "jeudi",
                       "vendredi", "samedi", "dimanche");
  }

  function __sleep()
  {
   for($i = 0; $i < sizeof($this->jour); $i++)
   {
    $chaine_result = substr($this->jour[$i], 0, 3);

    $this->jour[$i] = $chaine_result;
   }
   return array('jour');
  }

  function __wakeup()
  {
   $d = "di";
   $this->jour[0] .= $d;
   $this->jour[1] .= $d;
   $this->jour[2] .= "cre".$d;
   $this->jour[3] .= $d;
   $this->jour[4] .= "dre".$d;
   $this->jour[5] .= "e".$d;
   $this->jour[6] .= "anche";
  }
 }

 $objet = new semaine();
?>
<br><h3>Objet initial :</h3>
<?php
 print_r($objet);

 $obj_linearise = serialize($objet);
?>
<br><h3>Objet linéarisé :</h3>
<?php
 print_r($obj_linearise);
?>
<br><h3>Objet délinéarisé :</h3>
<?php
 $obj_delinerarise = unserialize($obj_linearise);
 print_r($obj_delinerarise);
?>
```

15.11 / La comparaison d'objets

La comparaison d'objets permet de vérifier si un objet est égal à un autre.

Il peut y avoir deux types d'égalité d'objets :

- soit les objets possèdent une même référence,
- soit les objets sont des instances d'une même classe et que leurs états sont identiques.

L'opérateur *===* indique si deux objets ont des références égales.

```
$obj1 = new UnObjet();
$obj2 = $obj1;
$obj3 = new UnObjet();

comparer($obj1, $obj2); //même référence
comparer($obj2, $obj3); //références différentes
comparer($obj1, $obj3); //références différentes

function comparer($obj1, $obj2){
  if($obj1 === $obj2)
    echo 'Les objets possèdent une même référence.';
  else
    echo 'Les objets ont des références différentes.';
}
```

L'opérateur *==* indique si les deux objets comparés proviennent d'une instanciation de la même classe et si les attributs et leur valeur sont identiques.

```
$obj1 = new UnObjet('0');
$obj2 = new UnObjet('1');
$obj3 = new UnObjet('0');

comparer($obj1, $obj2); //objets différents
comparer($obj2, $obj3); //objets différents
comparer($obj1, $obj3); //objets identiques

function comparer($obj1, $obj2){
  if($obj1 == $obj2)
    echo 'Les objets sont identiques.';
  else
    echo 'Les objets sont différents.';
}
```

Un objet ayant la même référence qu'un autre et ayant le même état, alors ces objets seront identiques dans les deux type de comparaisons.

Les opérateurs *!==* et *!=* effectuent l'opération de comparaison inverse à ceux précités, en l'occurrence *===* et *==*.

Exemple [voir]

```
<html>
<body>
<?php
class Nombre {
  private $ombre;
  public function Nombre($valeur){
    $this->nombre = $valeur;
  }
  public function getValeur(){
    return $this->nombre;
  }
  public function setValeur($valeur){
    $this->nombre = $valeur;
  }
  public function __toString(){
    return '[' . $this->nombre . ']';
  }
}

$n1 = new Nombre(1);
$n2 = $n1;
$n3 = new Nombre(0);
$n4 = new Nombre(1);

comparer($n1, $n2);
comparer($n1, $n3);
comparer($n1, $n4);
comparer($n2, $n3);
comparer($n2, $n4);
comparer($n3, $n4);


function comparer($obj1, $obj2){
  echo $obj1;
  echo '[' . $obj1 . '] == ';
  echo $obj2;
  echo '[' . $obj2 . '] : ';
  if($obj1 == $obj2)
    echo 'Les objets sont identiques.';
  else
    echo 'Les objets sont différents.';
  echo '
';
}

comparerReferences($n1, $n2);
comparerReferences($n1, $n3);
comparerReferences($n1, $n4);
comparerReferences($n2, $n3);
comparerReferences($n2, $n4);
comparerReferences($n3, $n4);

function comparerReferences($obj1, $obj2){
  echo $obj1;
  echo '[' . $obj1 . '] == ';
  echo $obj2;
  echo '[' . $obj2 . '] : ';
  if($obj1 === $obj2){
    echo 'Les objets ont une même référence.';
  }
  else {
    echo 'Les objets ont des références différentes.';
  }
  echo '
';
}

$n5 = $n1;
$n5->setValeur(2);

comparer($n1, $n5);
comparer($n2, $n5);
comparer($n3, $n5);
comparer($n4, $n5);
```

```
comparerReferences($n1, $n5);
comparerReferences($n2, $n5);
comparerReferences($n3, $n5);
comparerReferences($n4, $n5);
?>
</body>
</html>
```

15.12 / Les fonctions d'objet

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les objets.

Fonction
Description
$resultat = **call_user_method**(methode, $objet [, $param, ..., $paramN]);
appelle une méthode de l'objet spécifié avec éventuellement des paramètres.
$resultat = **call_user_method_array**(methode, $objet [$tab_param]);
appelle une méthode avec un tableau de paramètres.
true | false = **class_exists**(methode);
vérifie si une classe existe.
$chaine = **get_class**($objet);
retourne le nom de la classe d'un objet spécifié.
$tableau = **get_class_methods**(nom_classe);
retourne les noms des méthodes de la classe spécifiée dans un tableau.
$tableau = **get_class_vars**(nom_classe);
retourne les valeurs par défaut des attributs d'une classe.
$tableau = **get_declared_classes**(nom_classe);
retourne un tableau contenant les noms de toutes les classes définies.
$tableau = **get_object_vars**($objet);
retourne un tableau associatif contenant les propriétés d'un objet.
$chaine = **get_parent_class**($objet);
retourne le nom de la classe d'un objet.
true | false = **is_subclass_of**($objet, nom_classe);
détermine si un objet est une instance d'une sous-classe de la classe spécifiée.
method_exists($objet, methode);
vérifie si la méthode existe pour l'objet spécifié.

16 / Les formulaires

Les formulaires permettent de retourner des informations saisies par un utilisateur vers une application serveur.

```
<form method="GET | POST" action="page_cible.php">
 <input type="text" name="Champ_saisie" value="Texte"

 <select name="Liste_Choix" size="3">
  <option value="Option_1">Option_1</option>
  <option value="Option_2">Option_2</option>
  <option value="Option_3">Option_3</option>
 </select>

 <textarea name="Zone_Texte" cols="30" rows="5">
  Texte par défaut
 </textarea>

 <input type="checkbox" name="Case_Cocher[]" value="Case_1">
  Case à cocher 1<br>
 <input type="checkbox" name="Case_Cocher[]" value="Case_2">
  Case à cocher 2<br>
 <input type="checkbox" name="Case_Cocher[]" value="Case_3">
  Case à cocher 3<br>

 <input type="radio" name="Case_Radio" value="Case radio 1">
  Case radio 1<br>
 <input type="radio" name="Case_Radio" value="Case radio 2">
  Case radio 2<br>
  <input type="radio" name="Case_Radio" value="Case radio 3">
 Case radio 3<br>

 <input type="cancel" name="Annulation" value="Annuler">
 <input type="submit" name="Soumission" value="Soumettre">
</form>
```

La transmission d'un formulaire s'effectue selon une des deux méthodes d'envoi *GET* ou *POST*.

La méthode *GET* place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée. Mais cette méthode n'offrant aucune discrétion tend à devenir obsolète.

```
http://www.site.com/cible.php?champ=valeur&champ2=valeur
```

La méthode *POST* regroupe les informations dans l'entête d'une requête HTTP assurant, ainsi, une confidentialité des données efficace.

Lors de la soumission à une page de traitement, chaque élément de saisie est assimilé à une variable PHP dont le nom est constitué par la valeur de l'attribut *name* et son contenu par la valeur de l'attribut *value*.

```
<?php
 $resultat = $Champ_saisie . "<br>";

 $resultat .= $Liste_Choix . "<br>";

 $resultat .= $Zone_Texte . "<br>";

 for ($i = 0; $i < count($Case_Cocher); $i++)
 {
  $resultat .= $Case_Cocher[$i] . "<br>";
 }

 $resultat .= $Case_Radio . "<br>";
?>
```

La plupart des éléments d'un formulaire n'acceptent qu'une seule et unique valeur, laquelle est affectée à la variable correspondante dans le script de traitement.

```
$Champ_Saisie = "Ceci est une chaîne de caractères.";
```

Tandis que **pour des cases à cocher et les listes à choix multiples, plusieurs valeurs peuvent être sélectionnées** entraînant l'affectation d'un tableau de valeurs aux variables correspondantes. C'est pour cette raison qu'il est nécessaire de **déclarer un nom de variable suivi de crochets dans l'attribut *name* d'un champ *input type="checkbox"***. Cela a pour effet de simuler un tableau après la soumission du formulaire courant.

```
$Case_Cocher[0] = "Case radio 1";
$Case_Cocher[1] = "Case radio 3";
```

La page de traitement d'un formulaire peut être aussi bien une document cible spécifique que la page elle-même. Dans ce cas, la page doit contenir un dispositif spécial permettant de recueillir ses propres données et **la valeur de l'attribut *action* de la balise *form* doit être sa propre adresse**. Un moyen simple consisterait à **utiliser une variable d'environnement** telle que *$PHP_SELF*, *$PATH_INFO* ou encore *$SCRIPT_NAME*.

```
<?php
 if ($champ_saisie != 'Texte')
 echo 'Vous avez écrit dans le champ de saisie, '
                              . 'le texte suivant : ' . $champ_saisie;
?>
<form method="POST" action="<?php echo $PHP_SELF ?>">
 <input type="text" name="champ_saisie" value="Texte">

 <input type="submit" name="Soumission" value="Soumettre">
</form>
```

Une fonction très intéressante permet de détecter si une variable existe évitant ainsi l'affichage d'une erreur PHP lorsqu'une variable d'un formulaire n'a pas été renseignée par l'utilisateur.

```
<html>
 <body>
  <?
   if(isset($choix)&&($choix==2)){
    echo 'Un message particulier !';
   }
  else{
  ?>
  <form name="form" method="post">
   <input type="radio" name="choix"
          value="1" checked>
    Premier choix <br>
   <input type="radio" name="choix"
          value="2"
          onclick="form.action='essai_form.php';
                form.submit();">
    Google <br>
   <input type="submit" value="Valider">
  </form>
  <?
   }
  ?>
 </body>
</html>
```

Exemple [voir]

```
<!-- Fichier : form.html -->
<html>
 <head>
 <title>Formulaire</title>
 </head>
 <body>
  <form method="POST" action="traitement.php">
   <table align="center">
    <tr>
     <td>Sexe</td>
     <td><select name="Sexe" size="1">
        <option selected value="Monsieur">Monsieur</option>
        <option value="Madame">Madame</option>
        <option value="Mademoiselle">Mademoiselle</option>
       </select>
     </td>
    </tr>
    <tr>
     <td>Nom</td>
     <td><input type="text" name="nom" size="30"></td>
    </tr>
    <tr>
     <td>Prénom</td>
     <td><input type="text" name="prenom" size="30"></td>
    </tr>
    <tr>
     <td>eMail</td>
     <td><input type="text" name="email" size="30"></td>
    </tr>
    <tr>
     <td>Adresse</td>
     <td><input type="text" name="adresse" size="30"></td>
    </tr>
    <tr>
     <td>Code Postal</td>
     <td><input type="text" name="cp" size="30"></td>
    </tr>
    <tr>
     <td>Ville</td>
     <td><input type="text" name="ville" size="30"></td>
    </tr>
    <tr>
     <td>Age</td>
     <td>
      <input type="hidden" name="age" value="" >
      <input type="radio" name="age" value="0 - 7 ans" >
          0 - 7 ans
      <input type="radio" name="age" value="8 - 14 ans">
          8 - 14 ans
      <input type="radio" name="age" value="15 - 18 ans">
          15 - 18 ans
      <input type="radio" name="age" value="19 - 30 ans">
          19 - 30 ans
      <input type="radio" name="age" value="30 - 50 ans">
          30 - 50 ans
      <input type="radio" name="age" value="au-delà de 51 ans">
          au-delà de 51 ans
     </td>
    </tr>
    <tr>
     <td>Langage(s) connu(s)</td>
     <td><input type="hidden" name="langage[]" value="">
      <input type="checkbox" name="langage[]" value="HTML">
          HTML<br>
      <input type="checkbox" name="langage[]" value="XML">
          XML<br>
      <input type="checkbox" name="langage[]" value="XHTML">
          XHTML<br>
      <input type="checkbox" name="langage[]" value="Javascript">
          Javascript<br>
      <input type="checkbox" name="langage[]" value="ASP">
          ASP<br>
```

```
            <input type="checkbox" name="langage[]" value="PHP">
                PHP<br>
            <input type="checkbox" name="langage[]" value="SQL">
                SQL<br>
            <input type="checkbox" name="langage[]" value="Perl">
                Perl<br>
           </td>
          </tr>
          <tr>
           <td>Commentaire</td>
           <td>
            <textarea name="commentaire" cols="30" rows="3">
            </textarea>
           </td>
          </tr>
          <tr>
           <td>
            <input type="reset" value="Annuler" name="annulation">
           </td>
           <td>
            <input type="submit" value="Soumettre" name="soumission">
           </td>
          </tr>
        </table>
      </form>
    </body>
</html>
<!-- Fichier : traitement.php -->
<html>
  <head>
  <title>Traitement du formulaire</title>
  </head>
  <body>
    <?
      define("CRLF", "\r\n");

      $destinataire = "messagerie@laltruiste.com";
      $sujet = "Questionnaire d'admission";

      $exp_reg_email = ""
         . "^[a-z0-9]+((.|-|_)[a-z0-9]+)*@[a-z0-9]+((.|-|_)[a-z0-9]+)*(.([a-z]){2,4})$";
      $exp_reg_cp = "^([0-9]{5})$";

      $valide = true;

      $corps = "<table>";
      $corps .= "<tr><td>Sexe</td><td> : </td><td>" . $sexe . "</td></tr>";
      if ($nom == "")
      {
        echo("<h3>Le champ nom est vide !</h3>");
        $valide = false;
      }
      else
      {
        $corps .= "<tr><td>Nom</td><td> : </td><td>" . $nom . "</td></tr>";
      }
      if ($prenom == "")
      {
        echo("<h3>Le champ prénom est vide !</h3>");
        $valide = false;
      }
      else
      {
        $corps .= "<tr><td>Prénom</td><td> : </td><td>" . $prenom . "</td></tr>";
      }
      if (($email != "") && ereg($exp_reg_email, $email))
      {
        $corps .= "<tr><td>eMail</td><td> : </td><td>" . $email . "</td></tr>";
      }
      else
      {
        echo("<h3>Le champ eMail est vide ou incorrect !</h3>");
        $valide = false;
      }
```

```
if ($adresse == "")
{
  echo("<h3>Le champ adresse est vide !</h3>");
  $valide = false;
}
else
{
  $corps .= "<tr><td>Adresse</td><td> : </td><td>" . $adresse . "</td></tr>";
}
if (($cp != "") && ereg($exp_reg_cp, $cp))
{
  $corps .= "<tr><td>Code Postal</td><td> : </td><td>" . $cp . "</td></tr>";
}
else
{
  echo("<h3>Le champ code postal est vide ou incorrect !</h3>");
  $valide = false;
}
if ($ville == "")
{
  echo("<h3>Le champ ville est vide !</h3>");
  $valide = false;
}
else
{
  $corps .= "<tr><td>Ville</td><td> : </td><td>" . $ville . "</td></tr>";
}
if ($age == "")
{
  echo("<h3>Aucun choix n'a été sélectionné pour l'âge !</h3>");
  $valide = false;
}
else
{
  $corps .= "<tr><td>Age</td><td> : </td><td>" . $age . "</td></tr>";
 }
if (count($langage) <= 1)
{
  echo("<h3>Aucun langage n'a été sélectionné !</h3>");
  $valide = false;
}
else
{
  $nb = sizeof($langage);
  $s = ($nb > 1) ? "s" : "";
  $corps .= "<tr><td>Langage" . $s . "</td><td> : </td><td> ";
  for($i = 1; $i < $nb; $i++)
  {
    $separateur = $i != $nb - 1 ? ", " : ".</td></tr>";
    $corps .= $langage[$i] . $separateur;
  }
}
if ($commentaire == "")
{
  echo("Le champ commentaire est vide !");
  $valide = false;
}
else
{
  $corps .= "<tr><td>Commentaire</td><td> : </td><td>"
                                    . $commentaire . "</td></tr>";
}
$corps .= "</table>";

$entete = "From: " . $prenom . " " . $nom . " <" . $email . ">" . CRLF
     . "To: Administration <" . $destinataire . ">" . CRLF
     . "Reply-To: Webmaster <webmaster@laltruiste.com>" . CRLF
     . "Subject: " . $sujet . CRLF
     . "Date: " . date("r") . CRLF
     . "Message-ID: <1255388558@laltruiste.com>" . CRLF
     . "MIME-Version: 1.0" . CRLF
     . "Content-Type: text/html; charset=\"iso-8859-1\"" . CRLF
     . "Content-Transfer-Encoding: 7bit" . CRLF;
```

```
    if ($valide == true)
    {
      if (mail($destinataire, $sujet, $corps, $entete))
      {
        echo "<h3>Le message a été envoyé avec succès</h3>"
             . ""<u>Le contenu du message est le suivant:</u><br>" . $corps;
      }
      else
      {
        echo "<h2 style=\"color:#FF0000; text-align:center\">"
             . "Une erreur s'est produite !<br>"
             . "Veuillez réessayer la soumission ultérieurement.<br>Merci.</h2>";
       }
    }
    else
    {
      echo '<a href="javascript:history.go(-1)">Retournez à la page précédente</a>';
    }
   ?>
  </body>
</html>
```

17 / L'envoi de courrier électronique

L'envoi de courrier électronique s'effectue par l'intermédiaire de la fonction *mail()*.

```
true | false = mail(destinataires, sujet,
                    message[, entête][, paramètres])
```

Cette instruction permet d'**envoyer des messages simples à un ou plusieurs destinataires**.

```
$sujet = "Newsletter n°125";
$message = "Sommaire de la Newsletter..."
                . "\r\nContenu...\r\nMerci\r\n";
mail(abonne@dom.com, lecteur@free.fr,
                dev@lmb.ca, $sujet, $message);
// Envoi d'une email simple à plusieurs destinataires.
```

Toutefois, il est nécessaire de **configurer correctement le fichier *php.ini*** pour utiliser cette fonction sous win32 ou sous UNIX.

```
[mail function]
SMTP = localhost ; for win32 only
sendmail_from = adresse@localhost.com ; for win32 only
sendmail_path   =   ; for unix only, may supply arguments as well
                    (default is sendmail -t)
```

Des courriers électroniques plus complexes peuvent être construits comportant non seulement une structure de message élaborée en HTML par exemple mais aussi un entête (*headers*) spécifique correspondant au contenu et devant suivre les spécifications MIME.

Un courrier électronique est décomposable en plusieurs parties dont la première constitue l'entête du message avec des champs indiquant les adresses du destinataire et de l'expéditeur, ainsi que le sujet du message entre autres et en une seconde partie correspondant au corps du message. Ce dernier peut également être divisé en plusieurs sous-parties renfermant un contenu mixte ou alternatif.

```
$entete = "From: expediteur@dom.com";
$entete .= "To: destinataire@dom.net";

mail($adresse, $sujet, $corps, $entete);
```

Exemple [voir]

```
<!-- Fichier : formulaire.php -->
<html>
 <head>
 <title>Formulaire</title>
 </head>
 <body>
  <form action="traitement.php" method="POST">
   <table>
    <tr>
     <td>Nom</td>
     <td><input type="text" name="nom" size="20"></td>
    </tr>
    <tr>
     <td>Prénom</td>
     <td><input type="text" name="prenom" size="20"></td>
    </tr>
    <tr>
     <td><u>email :</u></td>
     <td>
      <input type="text" name="email" size="30">
     </td>
    </tr>
    <tr>
     <td><u>Fichier :</u></td>
     <td>
      <input type="file" name="fichier">
     </td>
    </tr>
    <tr>
     <td>
      <input type="reset" name="annulation" value="Annuler">
     </td>
     <td>
      <input type="submit" name="soumission" value="Envoyer">
     </td>
    </tr>
   </table>
  </form>
 </body>
</html>

<?php
 // Fichier : traitement.php
 function traiterErreur($champ){
  echo("<h3>Le champ ".$champ." est vide ou incorrect !</h3>");
  echo '<a href="javascript:history.go(-1)">Retournez à la page précédente</a>';
  return false;
 }
 $valide = true;
 $exp_reg_email = ""
      . "^[a-z0-9]+((\.|-|_)[a-z0-9]+)*@[a-z0-9]+((\.|-|_)[a-z0-9]+)*(\.([a-z]){2,4})$";

 if (isset($prenom) && ($prenom != ""))
 {
  $expediteur = $prenom;
 }
 else
 {
  $valide = traiterErreur('prenom');
 }
 if(isset($nom) && ($nom != ""))
 {
  $expediteur = " " . $nom;
 }
 else
 {
  $valide = traiterErreur('nom');
 }
 if(isset($email) && ($email != "") && ereg($exp_reg_email, $email))
 {
  $expediteur .= " <" . $email . ">";
 }
 else
 {
```

```
    $valide = traiterErreur('email');
  }

  if($valide == true)
  {
    define("CRLF", "\r\n");
    $destinataire = "Prénom Nom <webmaster@laltruiste.com>";
    $reponse = "Réponse <messagerie@laltruiste.com>";
    $sujet = "Envoi d'une image JPEG";

    $fic = fopen($fichier, "r");

    $piece_attachee = fread($fic, filesize($fichier));
    $piece_attachee = chunk_split(base64_encode($piece_attachee));

    fclose($fic);

    $boundary = "-----=".md5(uniqid (rand()));

    $entete = "To: " . $destinataire . CRLF;
    $entete .= "From: " . $expediteur . CRLF;
    $entete .= "Reply-To: " . $reponse . CRLF;
    $entete .= "Date: " . date("r") . CRLF;
    $entete = "MIME-Version: 1.0" . CRLF;
    $entete .= "Content-Type: multipart/mixed; boundary=". $boundary . CRLF . CRLF;

    $corps = "--$boundary" . CRLF;
    $corps .= "Content-Type: text/plain; charset=\"iso-8859-1\"" . CRLF;
    $corps .= "Content-Transfer-Encoding:8bit" . CRLF;
    $corps .= "Le fichier envoyé contient un fichier attaché..." . CRLF . CRLF;
    $corps .= "--$boundary" . CRLF;
    $corps .= "Content-Type: image/jpeg; name=". $fichier . CRLF;
    $corps .= "Content-Transfer-Encoding: base64" . CRLF;
    $corps .= "Content-Disposition: inline; filename=". $fichier . CRLF . CRLF;
    $corps .= $piece_attachee . CRLF . CRLF;
    $corps .= "--$boundary--" . CRLF;

    email($destinataire, $sujet, $corps, $entete);
  }
?>
```

18 / Les cookies

Les cookies permettent de sauvegarder des informations chez le client relative à sa visite sur un site.

Lors de la prochaine visite, PHP est capable de lire ces informations afin d'authentifier l'utilisateur et de reconstituer un environnement adapté.

La fonction *setcookie* permet d'envoyer un cookie sur l'ordinateur client.

```
setcookie(nom_cookie [, valeur [, date_expiration
                    [, chemin [, domaine [, sécurisation]]]]]);

setcookie("Session_site", SID, time()+3600*24, "/", ".site.com", 0);
/* envoi d'un cookie non sécurisé contenant
l'identifiant de session du site 'site.com'
avec une durée de vie de 24 heures */
```

La date d'expiration indique la durée de vie du cookie chez le client. En général, la fonction *time* est utilisée avec une expression numérique précisant un intervalle de temps.

```
time()+3600*12
// Le cookie sera valable entre : 10:12:00 - 22:12:00
```

Une date précise peut être également entrée en utilisant la fonction *mktime*.

```
mktime(0,0,0,31,12,2002)
// Le cookie sera valable jusqu'au 31 décembre 2002
```

Le chemin indique le répertoire du site où le cookie possède une validité. La valeur par défaut est la racine du site web.

Le domaine indique le nom DNS du site sur lequel le cookie possède une validité. Par défaut, le nom du domaine est celui à partir duquel le cookie a été envoyé. La valeur de cet argument doit comporter deux points '.', un au début et l'autre avant le suffixe.

L'argument sécurisation est un nombre entier indiquant si le cookie est sécurisé (1) ou s'il ne l'est pas (0 - par défaut).

Les cookies faisant partie des en-têtes HTTP, **la fonction *setcookie* doit être appelée avant l'affichage d'un document**, c'est pourquoi, il faut placer l'instruction avant le balisage HTML.

```
<?php
  setcookie("Nom", "Valeur");
?>
<html>
  ...
</html>
```

Les cookies envoyés au client sont automatiquement retournés à l'application PHP afin d'être convertis en une variable à l'instar des champs de saisie des fomulaires.

```
<?php
  echo $Session_site;

  echo $_COOKIE["Session_site"];

  echo $HTTP_COOKIE_VARS["Session_site"];
?>
```

Il existe deux possiblités pour **récupérer les valeurs d'un cookie**, soit par la **variable globale** correspondante, soit par le **tableau associatif global** *$_COOKIE* ou *$HTTP_COOKIE_VARS*.

Il est possible de **passer plusieurs valeurs à un cookie par l'intermédiaire de crochets** '[]' suivant le nom du cookie.

```
<?php
  setcookie( "site['sid']", SID );
  setcookie( "site['couleur']", $couleur );
```

```
 setcookie( "site['police']", $police );
 if (isset($site))
 {
  while(list($nom, $valeur) = each($site))
  {
     echo $nom . " = " . $valeur . "<br>";
  }
 }
?>
```

La suppression d'un cookie peut s'effectuer par l'intermédiaire de la réémission d'un cookie de même nom mais avec une valeur vide. Toutefois, toutes les informations contenues par le cookie sont supprimées contrairement au cookie lui même. Celui-ci existe toujours, mais il est devenu inutilisable.

```
setcookie("nom_cookie");
```

Exemple [voir]

```
<html>
 <!-- Fichier : formulaire.html -->
 <body>
  <?
   if(!isset($name))
    $name = 'laltruiste';
   $e = "";
   $pw = "";
   $cnt = 0;
   if (isset($_COOKIE[$name]) && is_array($_COOKIE[$name])){
    print_r($_COOKIE." ".count($_COOKIE[$name]));
    echo "<h3>Le cookie contient les informations suivantes :</h3>";
    echo "<h4>Extraction par la variable globale</h4>";

    while(list($nom, $valeur) = each(${$name}))
    {
     echo "<u>" . $nom . " :</u> " . $valeur . "<br>";
    }

    echo "<h4>Extraction par le tableau associatif global</h4>";

    $i = 0;
    while(list($nom, $valeur) = each($_COOKIE[$name]))
    {
     echo "<u>" . $nom . " :</u> " . $valeur . "<br>";
     switch($i++)
     {
      case 0:
       $e = $valeur;
       break;
      case 1:
       $pw = $valeur;
       break;
      case 2:
       $cnt = $valeur;
       break;
     }
    }
   }
  ?>
  <form method="post" action="fichier.php" name="form">
   <table border="0">
    <tr>
     <td>eMail</td>
     <td>: <input type="text" name="email" size="20"></td>
    </tr>
    <tr>
     <td>Mot de passe</td>
     <td>: <input type="password" name="password" size="20"></td>
    </tr>
    <tr>
     <td>Nom du cookie</td>
  <?if($cnt == 0){?>
     <td>: <input type="text" name="cookie" value="<?echo $name?>"></td>
  <?}else{?>
     <td>: <?echo $name?>
     <input type="hidden" name="cookie" value="<?echo $name?>"></td>
  <?}?>
    </tr>
    <tr>
     <td><input type="submit" name="soumettre" value="Envoyer"></td>
     <td><input type="submit" name="detruire" value="Détruire"
          onclick="document.form.action='detruire.php'; true"></td>
    </tr>
   </table>
   <input type="hidden" name="counter" value="<?echo $cnt?>">
  </form>
 </body>
</html>

<?php
// Fichier : traitement.php
setcookie($cookie."['email']", $email);
setcookie($cookie."['password']", $password);
```

```
setcookie($cookie."['count']", $counter + 1);
?>
<html>
  <body>
    <a href="cookie.php?name=<?echo $cookie?>">Retourner à la page précédente</a>
  </body>
</html>

<?php
  // Fichier : detruire.php
  if (sizeof($_COOKIE) > 0)
  {
    while(list($nom, $valeur) = each($_COOKIE))
    {
      setcookie($nom);
    }
  }
?>
<html>
  <body>
    <h4>Le cookie est détruit...</h4>
    <a href="cookie.php">Retourner à la page précédente</a>
  </body>
</html>
```

18.1 / Les fonctions HTTP

Le langage PHP dispose de nombreuses fonctions permettant de manipuler les informations HTTP.

Fonction
Description
$nombre = **header**($chaine);
envoie une entête HTTP selon le format défini dans la RFC 2616 et surtout avant toute commande PHP.
true | false = **headers_sent**();
vérifie si les entêtes HTTP ont bien été envoyés.
$nombre = **setcookie**(nom, valeur [, $date_expiration [, $chemin [, $domaine [, $securisation]]]]);
envoie un cookie.

19 / Les chaînes de requêtes

Les chaînes de requêtes (QueryStrings) correspondent à des informations associées à l'adresse URL avec un point d'interrrogation comme séparateur.

```
http://www.laltruiste.com/page.php?chaine_de_requete
```

Une chaîne de requête doit être constituée d'un ou plusieurs éléments dont chacun est associé à une valeur. Si la chaîne de requête contient plusieurs éléments, alors **chaque couple élément/valeur doit être séparé par un caractère esperluette** (&).

```
?element=valeur&element2=valeur2&...&elementN=valeurN
```

L'inconvénient principal des chaînes de requêtes réside dans le fait que **les données transmises au serveur par ce biais sont visibles par les utilisateurs dans le champ *Adresse* de leur navigateur**.

De plus, **une chaîne de requêtes ne peut dépasser le maximum de 255 caractères**.

Ainsi, l'utilisation de cette technique peut générer des **problèmes de sécurité** et des **limitations quant à la taille des données à transmettre** à une application Web.

Il existe deux méthodes pour l'utilisation des chaînes de requêtes :

- **L'insertion des informations directement après l'adresse URL de la page à atteindre.**

```
<form action="page.php?element=valeur"
        method="post">
  <input type="text" name="nom" value="DUPUIS">
  <input type="text" name="prenom" value="Michel">
  <input type="submit" name="Soumettre" value="Soumission">
</form>
```

- **L'application de la valeur *GET* à l'attribut *method* d'un formulaire.**

```
<form action="page.php" method="get">
  <input type="text" name="nom" value="DUPUIS">
  <input type="text" name="prenom" value="Michel">
  <input type="submit" name="Soumettre" value="Soumission">
</form>
```

Dans le premier cas, **seules les informations contenues dans l'adresse indiquée par l'attribut *action*, seront envoyées** par l'intermédiaire de la collection *QueryString*.

```
element=valeur
```

Dans le second cas, **tous les éléments du formulaire seront transmis au moyen de la collection *QueryString***, à la page destinataire soit *page.php*.

```
nom=DUPUIS&prenom=Michel
```

En conséquence, **l'expression *method="get"* provoque la transmission complète et automatique d'un formulaire à partir d'une chaîne de requête**.

La variable d'environnement *QUERY_STRING* permet d'extraire une chaîne de requêtes contenu dans l'adresse URL transmise, l'ensemble des éléments et leur valeur y compris celle du bouton de soumission.

```
<? echo $QUERY_STRING; ?>
// Retourne
nom=DUPUIS&prenom=Michel&Soumettre=Soumission
```

Plus précisément, en spécifiant simplement **une variable portant l'identificateur d'un élément**, il devient possible de récupérer la valeur liée.

```
<?
 echo $nom . " " . $prenom;

// Retourne
DUPUIS Michel
```

La variable prédéfinie HTTP *$_GET* permet également de récupérer la valeur d'un élément d'une chaîne de requête. La variable *$_GET* est en fait un tableau associatif dont les clés sont les identificateurs et les d'une chaîne de requêtes associé aux valeurs liées.

```
<?
 echo $_GET["nom"] . " " . $_GET["prenom"];

// Retourne
DUPUIS Michel
```

Une boucle peut être utilisée pour parcourir automatiquement l'ensemble des éléments d'une chaîne de requête.

```
<?
 echo "<ol>";
 while (list($cle, $valeur) = each($_GET)) {
  echo "<li>".$cle." : ".$valeur."</li>";
 }
 echo "</ol>"
?>
// affiche
<ol>
 <li>nom : DUPUIS</li>
 <li>prenom : Michel</li>
 <li>Soumettre : Soumission</li>
</ol>
```

La boucle *foreach* plus compacte produit le même effet.

```
foreach($HTTP_GET_VARS as $cle => $valeur) {
 echo $cle." : ".$valeur."\n";
}
```

Le tableau associatif *$_GET* étant définie dans la version PHP 4.1.0, il faudra **utiliser la variable *$HTTP_GET_VARS* pour assurer la compatibilité avec des versions plus anciennes**.

Un autre tableau associatif *$_POST* ou *$HTTP_POST_VARS* contient toutes les variables passées au script courant par l'intermédiaire d'une requête HTTP *POST* (voir les formulaires).

Exemple [voir]

```
<!-- Formulaire -->
<html>
  <body>
    <form
        action="traitement.php"
        method="get"
        name="formGet">

      <u>Saisir un titre :</u><br>
      <input type="text" name="Titre" value="Fatrasie" size="20"><br>

      <textarea name="Paragraphe" cols="30" rows="4">
        La chose va très mal
        Où point n'a de justice
        La chose va très mal
        Dit un veau de métal
      </textarea>

      <input type="submit" name="Soumettre" value="Soumission">

    </form>

  </body>
</html>

<html>
  <body>
    <h2>La chaîne de requête</h2>
    <h3>
      <? echo $QUERY_STRING; ?>
    </h3>
    <table>
      <tr>
        <th>Elément</th>
        <th>Valeur</th>
      </tr>
      <?
        $index = 0;
        foreach($HTTP_GET_VARS as $cle => $valeur) {
          $index++;
          echo "<tr><td><u>" . $cle . "(" . $index
                       . ") </u></td><td><b>" . $valeur
                           . "</b></td></tr>";
        }
      ?>
    </table>
  </body>
</html>
```

20 / Les sessions

Les sessions permettent de conserver des informations relatives à un utilisateur lors de son parcours sur un site web.

Ainsi, **des données spécifiques à un visiteur pourront être transmises de page en page** afin d'adapter personnellement les réponses d'une application PHP.

Chaque visiteur en se connectant à un site reçoit un numéro d'identification dénommé *identifiant de session* (*SID*).

La fonction *session_start* se charge de générer automatiquement cet identifiant unique de session et de créer un répertoire

```
<?php
 session_start();

 $Session_ID = session_id();
 // $Session_ID = 7edf48ca359ee24dbc5b3f6ed2557e90
?>
```

La fonction *session_start* doit être **placée au début de chaque page afin de démarrer ou de continuer une session**.

Par ailleurs, **un fichier est créé sur le serveur à l'emplacement désigné par le fichier de configuration *php.ini***, afin de recueillir les données de la nouvelle session.

```
[Session]
session.save_path= C:\PHP\sessiondata
; Fichier session = \sess_7edf48ca359ee24dbc5b3f6ed2557e90
```

Le fichier ***php.ini* peut également préciser un nom de session** par l'option *session.name* ou sa durée de vie par *session.gc_maxlifetime*.

La session en cours peut être détruite par le truchement de la fonction *session_destroy*. Cette commande supprime toutes les informations relatives à l'utilisateur.

```
session_destroy();
```

20.1 / Le traitement des variables de session

Les variables de session sont chargées dans une session par l'intermédiaire de la fonction *session_register*.

```
<?php
 session_start();
 ...
 session_register("nom_variable");
 ...
 session_register("nom_variableN");
?>
```

Cette fonction n'enregistre pas la valeur d'une variable dans un fichier de session. Elle permet seulement de déclarer une variable de session dans laquelle une valeur pourra transiter au travers de la session en cours.

Deux options de configuration déterminent le traitement des variables d'une session dans une application PHP.

Si l'option de configuration *track_vars* est activée et *register_globals* désactivée, les variables de la session seront disponibles seulement à partir du tableau associatif global *$HTTP_SESSION_VARS* ou plus récemment *$_SESSION*.

```
session_register("nom_variable");
...
echo $HTTP_SESSION_VARS["nom_variable"];

//ou directement
echo $_SESSION["nom_variable"];
```

En effet, avec *$_SESSION*, il n'est pas nécessaire d'utiliser les fonctions *session_register()*, *session_unregister()* et *session_is_registered()*. Les variables de sessions sont accessibles comme toute autre variable.

Si l'option de configuration *register_globals* est activée, les variables de la session seront disponibles à partir des variables globales correspondantes. Mais ceci n'est pas recommandé pour des raisons de sécurité.

```
session_register("nom_variable");
...
echo $nom_variable;
```

Si les options *track_vars* et *register_globals* sont activées, les variables globales et *$HTTP_SESSION_VARS* ou *$_SESSION* contiendront les valeurs des variables de la session en cours.

Le déchargement d'une variable de session s'accomplit par l'intermédiaire de la fonction *unset()* ou *session_unregister()* selon que l'option *register_globals* soit respectivement inactive ou active.

```
<?php
session_start();

//l'option register_globals est inactive
if(isset($_SESSION['variable']))
   unset($_SESSION['variable']);

//l'option register_globals est active
if(session_is_registered('variable'))
   session_unregister('variable');
?>
```

Le transport des informations entre les documents est réalisé par l'entremise soit d'un cookie, soit d'une requête HTTP. La dernière solution semble être la plus fiable puisque les cookies peuvent ne pas être acceptés par le client ou celui-ci pourrait les détruire en cours de session.

```
$Session_ID = session_id();
```

```
$nom_session = session_name();
$adresse = 'http://www.site.com/doc.html';
$lien = 'Le prochain document';

print_f('<a href="%s?%s=%s">%s</a>',
      $adresse_url, $nom_session, $Session_ID, $lien);

echo '<a href="' . $adresse_url . SID . '">' . $lien . '</a>'

/* affichent <a href="http://www.site.com/
doc.html?PHPSESSID=7edf48ca359ee24dbc5b3f6ed2557e90">
Le prochain document
</a> */
// redirection automatique
header("Location:" . "$adresse . "?" . SID);
```

Il suffit donc simplement de **concaténer l'identifiant de session à l'adresse URL** de la page cible pour que cette dernière puisse accéder aux informations conservées par la session. De plus, un mot-clé spéciale *SID* contient le nom de session et l'identifiant séparé par un signe égal.

Si **la directive de compilation *--enable-trans-sid*** est activée, il sera possible de ne pas ajouter l'identifiant de session à l'adresse URL pointée. Pour savoir si cela est le cas, il suffit de rechercher la directive dans la page résultant de la fonction *phpinfo* ou sinon de modifier l'option de configuration *session.use_trans_sid* dans *php.ini*.

```
[Session]
session.use_trans_sid = 1
```

Par défaut, PHP tente de passer par les cookies pour sauvegarder l'identifiant de session dans le cas où le client les accepterait. Pour éviter cela, il suffit de désactiver l'option de configuration *session.use_cookies* dans le fichier *php.ini*.

```
[Session]
session.use_cookies = 0
; désactive la gestion des sessions par cookie
```

Dans la situation ou il serait nécessaire d'utiliser les cookies, il est impératif de doubler les moyens de conservation du *SID*.

Exemple [voir]

```
<!-- Fichier : formulaire.html -->
<html>
  <body>
    <form method="post" action="traitement.php">
      <table border="0">
        <tr>
        <td><u>Nom :</u></td>
        <td>
          <input type="text" name="cNom" size="20" value="RIVES">
        </td>
        </tr>
        <tr>
        <td><u>Prénom :</u></td>
        <td>
          <input type="text" name="cPrenom" size="20" value="Jean-Pierre">
        </td>
        </tr>
        <tr>
        <td><u>eMail :</u></td>
        <td>
          <input type="text" name="cEmail" size="20"
                 value="j_p.rives@domaine.com">
        </td>
        </tr>
        <tr>
        <td> </td>
        <td>
         <input type="submit" name="soumettre" value="Envoyer">
        </td>
        </tr>
      </table>
```

```
    </form>
  </body>
</html>
<!-- Fichier : traitement.php -->
<?
  session_start();

  $nom = $cNom;
  $prenom = $cPrenom;
  $email = $cEmail;

  session_register("nom");
  session_register("prenom");
  session_register("email");

  header("Location: session.php?" . session_name() . "=". session_id());
?>
<!-- Fichier : session.php -->
<?
  session_start();
?>
 <html>
  <body>
    <?
     echo("<u>Identifiant de session :</u> <b>"
                          . session_id() . "</b><br>");
      echo("<u>Nom de la session :</u> <b>"
                          . session_name() . "</b><br><br>");
      echo("<u>Nom :</u> <b>". $nom . "</b><br>");
      echo("<u>Prénom :</u> <b>" . $prenom . "</b><br>");
      echo("<u>eMail :</u> <b>" . $email . "</b><br>");
    ?>
  </body>
</html>
<?
  session_destroy();
?>
```

20.2 / Les fonctions de sessions

Le langage PHP dispose de nombreuses fonctions permettant de travailler avec les sessions.

Fonction
Description
true | false = **session_start**();
initialise les données d'une session.
true | false = **session_destroy**();
détruit les données d'une session en cours.
$chaine = **session_name**([$chaine]);
retourne ou affecte le nom de la session en cours.
$chaine = **session_module_name**([$chaine]);
retourne ou affecte le nom du module en cours de la session actuelle.
$chaine = **session_save_path**([$chaine]);
retourne ou affecte le chemin de sauvegarde de la session en cours.
$chaine = **session_id**([$chaine]);
retourne ou affecte l'identifiant de la session en cours.
true | false = **session_register**("nom_variable", ..., "nom_variableN");
enregistre une ou plusieurs variables dans la session en cours.
true | false = **session_unregister**("nom_variable");
supprime une variable dans la session en cours.
session_unset();
supprime la totalité des variables de la session en cours.
true | false = **session_is_registered**("nom_variable");
vérifie si une variable a été enregistrée dans la session en cours.
$tableau = **session_get_cookie_params**();
retourne un tableau associatif contenant les paramètres du cookie (clés : lifetime, path, domain) de la session en cours.
session_set_cookie_params($duree, $chemin, $domaine);
définit les paramètres du cookie (durée de vie, chemin d'accès, domaine d'accès) de la session en cours.
true | false = **session_decode**($chaine);
décode les données de session à partir d'une chaîne de caractères fournie.
true | false = **session_encode**();
encode les données de session dans une chaîne de caractères.
session_set_save_handler($ouverture, $fermeture, $lecture, $ecriture, $destruction, $duree);
définit les fonctions à utiliser pour la gestion du stockage des sessions.
$chaine = **session_cache_limiter**([$parametre]);
retourne ou spécifie le limiteur de cache. Le paramètre peut prendre la valeur *nocache* désactivant la mise en cache côté client, *public* le permettant ainsi que *private* mais en imposant des restrictions.

session_end();
enregistre les données de la session en cours et la termine.
session_readonly();
lit les variables de la session en cours en lecture seule.

21 / La gestion des connexions

Le langage PHP dispose de plusieurs outils permettant de gérer les connexions des utilisateurs sur un site web.

Il existe trois états possibles en ce qui concerne le statut des connexions.

1. **NORMAL** : la connexion est ouverte et le script est en cours d'exécution.
2. **ABORTED** : le client a annulé la connexion et le script est arrêté par défaut.
3. **TIMEOUT** : le script a provoqué une déconnexion due à la fin de la durée d'exécution convenue.

Les deux états *ABORTED* et *TIMEOUT* peuvent survenir en même temps dans le cas ou d'une part si PHP est configuré de telle sorte à ignorer les déconnexions et d'autre part lorsque le script arrive à expiration.

L'état *ABORTED* en principe interrompt logiquement le script dès que l'utilisateur se déconnecte du site. Toutefois, il est possible de **modifier ce comportement en activant l'option de configuration *ignore_user_abort*** dans le fichier *php.ini*.

En outre, **si une fonction de fermeture a été enregistrée avec l'instruction *register_shutdown_function***, le moteur de script se rendra compte après que le client se soit déconnecté puis lors de la prochaine exécution du script que celui ci n'est pas terminé, ce qui provoquera l'appel de la fonction de fermeture mettant fin au script. losque le script se termine normalement, cette fonction sera également appelée.

L'instruction *connection_aborted* permet d'exécuter une fonction spécifique à la déconnexion d'un client. Si la déconnexion est effective, la fonction *connection_aborted* retourne *true*.

Un script expire par défaut après une durée de 30 secondes. Néanmoins, ce temps peut être **modifié par l'intermédiaire de l'option de configuration *max_execution_time*** dans le fichier *php.ini*.

Une fonction se chargeant de terminer le script sera appelée si le délai arrive à expiration ou si le client se déconnecte.

Les fonctions *connection_timeout*, *connection_aborted* et *connection_status* permettent respectivement de vérifier si l'état de la connexion est *ABORTED*, *TIMEOUT* et les trois états possibles.

21.1 / Les fonctions de connexions

Le langage PHP dispose de plusieurs fonctions permettant de gérer les connexions des clients sur une application PHP.

Fonction
Description
$nombre = **connection_aborted**();
vérifie si le client a abandonné la connexion.
$nombre = **connection_status**();
retourne les bits représentant le statut de la connexion.
true | false = **connection_timeout**();
vérifie l'expiration du script en cours.
$nombre = **ignore_user_abort**($parametre);
détermine si lors de la déconnexion d'un client, le script doit continuer ou arrêter son exécution. Le paramètre peut valoir soit '0' pour un comportement normal, '1' pour un abandon et '2' pour une expiration.

22 / Les dates et les heures

Les dates et les heures sont utilisées très couramment dans les applications PHP. A cet effet, ce langage propose de nombreuses solutions permettant de travailler avec des valeurs temporelles.

```
25/03/2002 12:45:58
```

Plusieurs fonctions retournent directement la date et l'heure courantes en proposant directement des masques de formatage.

```
$date = date("d/m/Y H:i:s 0");
// ex. : $date = '26/03/2002 19:57:02 +0100'
```

Néanmoins, **les éléments littéraux de dates sont essentiellement disponibles dans la langue anglaise**, si bien que pour obtenir une date dans un format français, il faut passer par une opération de substitution à l'aide des tableaux.

```
echo $date("l j F Y");
// retourne 'Sunday 24 Marsh 2002'
```

22.1 / Les fonctions de dates et d'heures

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les dates et les heures.

Fonction
Description
true | false = **checkdate**($mois, $jour, $annee);
vérifie la validité d'une date.
$chaine = **date**($format [, $nombre]);
retourne une chaîne de caractères date/heure selon le format spécifié et représentant la date courante par défaut.
$tableau = **getdate**([$nombre]);
retourne les éléments de date et d'heure dans un tableau associatif.
$tableau = **gettimeofday**();
retourne l'heure courante dans un tableau associatif.
$chaine = **gmdate**($format [, $nombre]);
retourne une chaîne de caractères date/heure GMT/CUT selon le format spécifié et représentant la date courante par défaut.
$nombre = **gmmktime**($heure, $minute, $seconde, $mois, $jour, $année [, 1/0]);
retourne l'instant UNIX d'une date GMT spécifiée et avec éventuellement une heure d'hiver (1).
$chaine = **gmstrftime**($format [, nombre]);
formate une date/heure GMT/CUT en fonction des paramétrages locaux définis par *setlocale*.
$tableau = **localtime**([$nombre][, $tab_associatif]);
retourne l'heure locale dans un tableau indicé par défaut ou associatif (1).
$chaine = **microtime**();
retourne l'instant UNIX courant en secondes et microsecondes.
$nombre = **mktime**($heure, $minute, $seconde, $mois, $jour, $année [, 1/0]);
retourne l'instant UNIX d'une date spécifiée et avec éventuellement une heure d'hiver (1).
$chaine = **strftime**($format [, $instant]);
formate une date/heure locale avec les options locales.
$nombre = **time**();
retourne l'instant UNIX courant.
$nombre = **strtotime**($chaine [, $instant]);
transforme un texte anglais représentant une date en instant UNIX.

22.2 / Les formats de date et d'heure

Plusieurs fonctions traitant des dates et des heures contiennent un argument spécifique pour le formatage d'expressions temporelles.

```
$format = "d / m / Y";
$chaine = date($format);
// $chaine contient '24 / 03 / 2002'
```

Format	Description
a	représente *am* (matin) ou *pm* (après-midi).
a	représente *AM* (matin) ou PM (après-midi).
b	représente une heure Internet Swatch.
d	représente le jour du mois sur deux chiffres allant de 01 à 31.
D	représente le jour de la semaine en trois lettres et en anglais (Sun, ..., Sat).
F	représente le mois complet en anglais (January, ..., December).
g	représente une heure au format 12 heures allant de 1 à 12.
G	représente une heure au format 24 heures allant de 1 à 24.
h	représente une heure au format 12 heures avec un zéro de complément allant de 00 à 11.
H	représente une heure au format 24 heures allant de 00 à 23.
i	représente les minutes allant de 00 à 59.
i	est égal à 1 si l'heure d'été est activée ou 0 pour l'heure d'hiver.
j	représente le jour du mois allant de 1 à 31.
l	représente le jour de la semaine complet et en anglais (Sunday, ..., Saturday).
L	est égal à 1 si l'année est bissextile, sinon 0.
m	représente un mois allant de 01 à 12.
M	représente un mois en trois lettres et en anglais (Jan, ..., Dec).
n	représente un mois allant de 1 à 12.
O	représente la différence d'heures avec l'heure de Greenwich (*+0100*).
r	représente un format de date conforme au RFC 822 (*Mon, 25 Mar 2002 05:08:26 +0100*).
s	représente les secondes allant de 00 à 59.
S	représente le suffixe ordinal d'un nombre en anglais et sur deux lettres *th* ou *nd*.
t	représente le nombre de jours dans le mois (28, 29, 30 ou 31).
T	représente le fuseau horaire.
u	représente les secondes depuis une époque.
w	représente le jour de la semaine allant de 0 (Dimanche) à 6 (Samedi).
Y	représente une année sur quatres chiffres (2002).
y	représente une année sur 2 chiffres (02).
z	représente le jour de l'année allant de 0 à 365.

Z	représente le décalage horaire en secondes.

Exemple [voir]

```
<?php
  $tab_jour = array("Dimanche", "Lundi", "Mardi",
                    "Mercredi", "Jeudi", "Vendredi", "Samedi");
  $tab_mois = array("Janvier", "Février", "Mars",
                    "Avril", "Mai", "Juin",
                    "Juillet", "Août", "Septembre",
                    "Octobre", "Novembre", "Décembre");
  $format = "w";
  $jour = date($format);
  $format = "n";
  $mois = date($format);
  echo $tab_jour[$jour] . " " . date("d") . " " . $tab_mois[$mois-1] . " " . date("Y");
  $format = "l d F Y";
  echo date($format);
?>
```

22.3 / Les fonctions de calendrier

Le langage PHP dispose de plusieurs fonctions permettant de travailler sur les calendriers grégoriens, juliens ou encore français.

Fonction
Description
$chaine = **JDToGregorian**($nb_jour);
convertit le nombre de jours du calendrier Julien en date grégorienne (MM/JJ/AAAA).
$nombre = **GregorianToJD**($MM,$JJ,$AAAA);
convertit une date grégorienne en nombre de jours du calendrier Julien.
$chaine = **JDToJulian**($nb_jour);
convertit le nombre de jours du calendrier Julien en date du calendrier Julien.
$nombre = **JulianToJD**($MM,$JJ,$AAAA);
convertit le nombre de jours du calendrier Julien en date du calendrier Julien.
$nombre = **JDToJewish**($MM,$JJ,$AAAA);
convertit le nombre de jours du calendrier Julien en date du calendrier juif.
$chaine = **JewishToJD**($nb_jour);
convertit une date du calendrier juif en nombre de jours du calendrier Julien.
$chaine = **JDToFrench**($nb_jour);
convertit le nombre de jours du calendrier Julien en date du calendrier français républicain.
$chaine = **FrenchToJD**($MM,$JJ,$AAAA);
convertit une date du calendrier français républicain en nombre de jours du calendrier Julien.
$chaine = **JDMonthName**($nb_jour, $mode);

retourne le nom du mois à partir d'un nombre de jours Julien et selon l'un des modes suivants :

Mode	Signification
0	Grégorien - abrégé
1	Grégorien
2	Julien - abrégé
3	Julien
4	Juif
5	Républicain français

$valeur = **JDDayOfWeek**($nb_jour, $mode);

retourne le numéro du jour de la semaine à partir d'un nombre de jours Julien et selon l'un des modes suivants :

Mode	Signification
0	retourne le numéro du jour comme un entier (0=dimanche, 1=lundi, etc.).
1	retourne une chaîne contenant le nom du jour (anglais grégorien).
2	retourne une chaîne contenant le nom abrégé du jour de la semaine (anglais grégorien).

$date = **easter_date**($AAAA);
retourne une date UNIX pour Pâques, à minuit.
$nombre = **easter_days**($AAAA);
retourne le nombre de jours entre le 21 mars et Pâques pour une année indiquée.
$nombre = **unixtojd**($duree_unix);
convertit une durée UNIX en nombre de jours Julien.
$durée_unix = **jdtounix**($nb_jour);

convertit un nombre de jours Julien en timestamp UNIX.

23 / Le système de fichiers

Le langage PHP propose de nombreux outils permettant de manipuler les fichiers et les dossiers présents sur les disques durs d'un serveur.

Evidemment, une telle application ne devra pas être **accessible que pour des utilisateurs disposant d'une permission appropriée**. Les risques de malveillances sont d'autant plus important, qu'il est nécessaire de **renforcer la sécurisation du serveur**.

23.1 / La manipulation des fichiers et dossiers

Le langage PHP dispose de nombreuses fonctions relatives au système de fichiers, permettant de manipuler aisément tout élément d'une arborescence.

Un répertoire peut être ouvert par l'intermédiaire de la fonction *opendir()* délivrant une ressource à partir de laquelle, il sera possible de lire ou écrire des fichiers.

```
$repertoire = "repertoire/";
 $dossier = opendir($repertoire);
```

La fonction *readdir()* permet de lire une entrée d'un dossier, soit le fichier sur lequel est disposé le pointeur de dossier.

```
<?
 $repertoire = "un_repertoire";
 $id_dossier = opendir($repertoire);
 while ($fichier = readdir($id_dossier)) {
  $id_fichier = $repertoire.$fichier;
  if(is_file($id_fichier)) {
   echo "<table border="1"><tr><th>Nom</th><td>".$fichier;
   echo "</td></tr><tr><th>Type</th><td>"
    .filetype($id_fichier);
   echo "</td></tr><tr><th>Taille</th><td>"
    .filesize($id_fichier)." octets";
   echo "</td></tr><tr><th>Création</th><td>"
    .date("d/m/Y H:i:s ", filectime($id_fichier));
   echo "</td></tr><tr><th>Modification</th><td>"
    .date("d/m/Y H:i:s ", filemtime($id_fichier));
   echo "</td></tr><tr><th>Dernier accès</th><td>"
    .date("d/m/Y H:i:s ", fileatime($id_fichier));
   echo "</td></tr><tr><th>ID Propriétaire</th><td>"
    .fileowner($id_fichier);
   echo "</td></tr><tr><th>Permission</th><td>"
    .fileperms($id_fichier);
   echo "</td></tr></table>";
  }
 }
 closedir($id_dossier);
?>
```

La fonction *readdir()* retourne les éléments dans n'importe quel ordre. De plus, tous les dossiers sont retournés, y compris "." et "..".

```
while ($fichier = readdir($dossier)) {
 // Cette instruction permet d'éviter de lister // les deux éléments "." et ".."
 if ($id_fichier != "." && $id_fichier != ".."){ ... }
}
```

La création d'un nouveau dossier peut être réalisée par le biais de la fonction *mkdir()*.

```
<?php
 $id_dossier = mkdir("chemin/dossier", 0700);
?>
```

La fonction *is_dir()* retourne le booléen *true* si l'élément entré en argument est un véritable dossier, dans le cas contraire, elle retourne *false*.

```
true | false = is_dir(string dossier);
```

Il est possible de **parcourir les entrées d'un dossier en utilisant une boucle *while***, dont la condition peut être la lecture d'un dossier par la fonction *readdir()* renvoyant lorsque la fin est atteinte, la valeur booléenne *false*.

```
<?php
 $i = 0;
 $dir = "C:\Inetpub\wwwroot\laltruiste\coursphp\";
 $dossier = opendir($dir);
 while ($entree = readdir($dossier))
 {
```

```
    $i++;
    echo $i . ' : <a href="cousphp/'
         . $entree . '">' . $entree . '</a><br>';
  }
  closedir($dossier);
?>
```

D'autre part, **le pointeur de fichier se place directement au début, soit à l'octet *0* d'un fichier lors de l'ouverture de ce dernier**. La fonction *fseek()* est capable de modifier la position du pointeur d'un certain nombre d'octets. Tandis que *ftell()* retourne cette position. Quant à *rewind()*, il le ramène à la position initiale du fichier.

```
<?php
  $chemin = "c:\autoexec.bat";
  $id_fichier = fopen($chemin, "r");
  echo "Position initiale : " . ftell($id_fichier) . "<br>";
  fseek($id_fichier, filesize($chemin));
  echo "Position de fin : " . ftell($id_fichier) . "<br>";
  rewind($id_fichier);
  echo "Position de retour : " . ftell($id_fichier) . "<br>";
  fclose($id_fichier);
?>
```

La fermeture d'un dossier s'opère par la fonction *closedir()*.

```
closedir($id_dossier);
```

La suppression d'un dossier s'effectue par la fonction *rmdir()*, mais les éléments contenus par ce dossier doivent être auparavant effacés. C'est-pourquoi, il est nécessaire de recourir à une fonction récursive pour supprimer un dossier non vide.

```
function effacer($fichier) {
  if (file_exists($fichier)) {
    chmod($fichier,0777);
    if (is_dir($fichier)) {
      $id_dossier = opendir($fichier);
      while($element = readdir($id_dossier)) {
        if ($element != "." && $element != "..")
        delete($fichier."/".$element);
      }
      closedir($id_dossier);
      rmdir($fichier);
    }
    else unlink($fichier);
  }
}

$repertoire = "c:\un_repertoire_inutile\";
effacer($repertoire);
```

23.2 / La manipulation des fichiers

Suite à l'ouverture d'un dossier, tous les fichiers peuvent être accédés pour une utilisation quelconque telle qu'une lecture ou une modification ou une suppression.

L'ouverture d'un fichier s'effectue par l'intermédiaire de la fonction *fopen()* délivrant une ressource.

```
$fichier = readdir($dossier);
$id_fichier = fopen($fichier, "r+");
```

La ressource peut être ouverte en lecture, en écriture ou en lecture et écriture. Pour cela le second argument doit valoir respectivement *r*, *w* ou *r+*, *w+*, *a* et *a+*.
Les deux derniers permettent non seulement d'accéder à un fichier en écriture seule ou lecture et écriture mais aussi de créer un fichier s'il n'existe pas.

```
<?
 $fichier = "chemin/un_fichier.ext";
 $id_fichier = fopen($fichier, "a+");
 fputs($id_fichier, $texte);
?>
```

Plusieurs fonctions PHP servent à vérifier une caractéristique intrinsèque, par exemple, si une variable de système de fichiers est un répertoire (*is_dir()*), un fichier (*is_file()*), un exécutable (*is_executable()*), un lien (*is_link*) ou encore si le fichier est ouvert en lecture seule (*is_readable()*), en écriture seule (*is_writeable()*) et enfin si le fichier a été téléchargé par la méthode HTTP POST (*is_uploaded_file()*).

```
<?php
 $chemin = "c:\\site\\";
 if (is_dir($chemin))
 { echo $chemin . " est un dossier !"; }

 $chemin = "c:\\site\\index.html";
 if (is_file($chemin))
 { echo $chemin . " est un fichier !"; }
?>
```

La fonction *file_exists()* s'assure de l'existence d'un fichier pour le chemin spécifié.

```
$valide = file_exists($fichier);
```

La lecture d'un fichier est réalisable par différents moyens :

- soit par **la fonction *fgets()* retournant une partie d'un fichier sous forme d'une chaîne de caractères**,

```
<?
 $fichier = "chemin/un_fichier.ext";
 $id_fichier = fopen($fichier,"r");
 while (!feof($id_fichier)) {
  $contenu = fgets($id_fichier, 4096);
  echo $contenu;
 }
?>
```

- soit par **la fonction *file()* retournant tout le fichier dans un tableau**,

```
<?
 $fichier = "chemin/un_fichier.ext";
 if(file_exists($fichier))
  $tableau = file($fichier);
 while(list($cle,$valeur) = each($tableau)) {
  echo "<p>".$valeur."</p>";
}
?>
```

- soit par la fonction *fread()* permettant une **lecture binaire du fichier** et retournant également une chaîne de caractères

```
<?
 $fichier = "chemin/un_fichier.ext";
 $id_fichier = fopen($fichier, "rb");
 $contenu = fread($id_fichier, filesize ($fichier));
?>
```

- soit par **la fonction *fgetss()* permettant de lire un fichier HTML ou PHP en éliminant toutes les balises** relatives aux langages précités,

```
<?
 $fichier = "url/un_fichier.html";
 $id_fichier = fopen($fichier, "rb");
 $contenu =  fgetss(element, filesize ($fichier));
?>
```

- soit par **la fonction *fgetcsv()* spécialisée dans la lecture des fichiers de valeurs séparées par des virgules** (.csv),

```
<?php
 $ligne = 1;
 $id_fichier = fopen ("chemin/un_fichier.csv","r");
 while ($tableau = fgetcsv($id_fichier, 1024, ".")) {
  $nb = count($tableau);
  echo "<h3>$nb champs dans la ligne $ligne: </h3>";
  $ligne++;
  for ($i=0; $i<$nb; $i++) {
   echo "<p>".$tableau[$i]."</p>";
  }
 }
?>
```

Un fichier peut être **affiché directement à l'écran par le truchement da la fonction *readfile()***.

```
<?
 $fichier = "url/un_fichier.html";
 $longueur_fichier = readfile($fichier);
?>
```

Les fichiers peuvent délivrer plusieurs informations à l'aide de fonctions appropriées telles que *filesize* pour leur taille, *filetype* pour le type, *fileatime*, *filectime* et *filetmime* pour la date et l'heure du dernier accès ou de la dernière modification, ou encore *stat*, *fstat*, et *lstat* retournant un tableau de caractéristiques.

```
<?php
 $chemin = "c:\\autoexec.bat";
 $id_fichier = fopen($chemin, "r");
 echo "taille totale de $chemin : " . filesize($chemin) . " octets<br>";
 echo "Date et heure du dernier accès : "
     . fileatime($fichier) . " " . filectime($fichier) . "<br>";
 echo "Date de la dernière modification : "
     . filemtime($fichier) . "<br>";

 print_r(fstat($id_fichier));

 fclose($id_fichier);
?>
```

La fermeture d'un fichier s'opère par la fonction *fclose()*.

```
fclose($id_fichier);
```

La suppression d'un fichier s'effectue par l'intermédiaire de la fonction *unlink()*.

```
int | false = unlink(string fichier)
```

23.3 / Le téléchargement de fichier

Le langage PHP 4 dispose de plusieurs outils facilitant le téléchargement vers le serveur et la gestion des fichiers provenant d'un client.

Un simple formulaire comportant **un champ *input* de type *file* suffit au téléchargement d'un fichier** qui subséquemment, devra être traité par un script PHP adapté.

```
<form method="POST" action="traitement.php"
        enctype="multipart/form-data">
  <input type="hidden" name="MAX_FILE_SIZE" value="Taille_Octets">
  <input type="file" name="fichier" size="30"><br>
  <input type="submit" name="telechargement" value="telecharger">
</form>
```

Un champ caché doit être présent dans le formulaire afin de spécifier une taille maximum (*MAX_FILE_SIZE*) pour le fichier à télécharger. Cette taille est par défaut égale à deux mégaoctets.

En PHP 4, le tableau associatif global *$HTTP_POST_FILES* ou *$_FILES* contient plusieurs informations sur le fichier téléchargé à condition que l'option de configuration *track_vars* soit activée dans le fichier *php.ini*.

- **$_FILES['fichier']['name']** : fournit le nom d'origine du fichier.
- **$_FILES['fichier']['type']** : fournit le type MIME du fichier.
- **$_FILES['fichier']['size']** : fournit la taille en octets du fichier.
- **$_FILES['fichier']['tmp_name']** : fournit le nom temporaire du fichier.
- **$_FILES['fichier']['error']** : fournit le code d'erreur associé au téléchargement.

La valeur *error* a été introduite à partir de la version 4.2.0.

PHP 3 quant à lui, peut faire appel aux variables globales ou/et au tableau associatif globale *$HTTP_POST_VARS* à condition que respectivement les options de configuration *register_globals* et/ou *track_vars* soient activées dans le fichier *php.ini*.

- **$fichier** : renvoie le nom temporaire du fichier.
- **$fichier_name** : renvoie le nom d'origine du fichier.
- **$fichier_size** : renvoie la taille en octets du fichier.
- **$fichier_type** : renvoie le type MIME du fichier.
-
- **$HTTP_POST_VARS['fichier']** : fournit le nom temporaire du fichier.
- **$HTTP_POST_VARS['fichier_name']** : fournit le nom d'origine du fichier.
- **$HTTP_POST_VARS['fichier_type']** : fournit le type MIME du fichier.
- **$HTTP_POST_VARS['fichier_size']** : fournit la taille en octets du fichier.

Par défaut, **le fichier envoyé par le client est stocké directement dans le répertoire indiqué par l'option de configuration *upload_tmp_dir*** dans le fichier *php.ini*.

```
upload_tmp_dir = c:\PHPuploadtemp
```

La méthode *putenv()* permet de modifier pour la durée du script, l'emplacement du dossier temporaire.

```
putenv('upload_tmp_dir=/idsite/temp');
```

Plusieurs fonctions spécialisées permettent la validation d'un fichier téléchargé pour son utilisation ultérieure.

La fonction *is_uploaded_file* indique si le fichier a bien été téléchargé par la méthode HTTP POST.

```
$booleen =
  is_uploaded_file($_FILES['fichier']['tmp_name']);
```

La fonction *move_uploaded_file* vérifie si le fichier a été téléchargé par la méthode HTTP POST, puis si c'est le cas le déplace vers l'emplacement spécifié.

```
$booleen =
  move_uploaded_file($_FILES['fichier']['tmp_name'],
  "c:\temporaire\fichier_telecharge\");
```

Il est possible de **télécharger plusieurs fichiers** en même temps, en utilisant des crochets à la suite du nom du champ afin d'indiquer que **les informations relatives aux fichiers seront stockées dans un tableau**.

```
<form action="traitement.php" method="POST"
        enctype="multipart/form-data">
  <input type="file" name="fichier[]"><br>
  ...
  <input type="file" name="fichierN[]"><br>
  <input type="submit" value="Envoyer" name="soumission">
</form>
...
for($i = 0; $i < sizeof($_FILES['fichier']['name']); $i++)
{
  echo "Nom du fichier : " . $_FILES['fichier']['name'][$i];
}
```

Les fichiers téléchargés sont automatiquement effacés du répertoire temporaire au terme du script. Ainsi, il est nécessaire de déplacer les fichiers vers un autre endroit ou de les renommer si ceux-ci doivent être conservés.

Exemple [voir]

```
<!-- Fichier : formulaire.html -->
<html>
  <body>
    <form method="POST" action="traitement.php"
              enctype="multipart/form-data">
      <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
      <input type="file" name="fichier" size="30"><br>
      <input type="submit" name="telechargement" value="telecharger">
    </form>
  </body>
</html>

<?php
// Fichier : traitement.html
  $repertoire = "f:\PHP\uploadtemp";

  if (is_uploaded_file($HTTP_POST_FILES['fichier']['tmp_name']))
  {
    $fichier_temp = $HTTP_POST_FILES['fichier']['tmp_name'];
    echo "<h3>Le fichier a été téléchargé avec succès "
       . "à l'emplacement suivant : <br>'" . $fichier_temp . "'</h3>";

    $nom_fichier = $HTTP_POST_FILES['fichier']['name'];
    echo "<h3>Le nom d'origine du fichier est '"
          . $nom_fichier . "'.</h3>";
    echo "<h3>Le type du fichier est '"
          . $HTTP_POST_FILES['fichier']['type'] . "'.</h3>";
    echo "<h3>La taille du fichier est de '"
          . $HTTP_POST_FILES['fichier']['size'] . " octets'.</h3>";

    copy($HTTP_POST_FILES['fichier']['tmp_name'], $repertoire . $nom_fichier);
  }
  else
  {
    echo '<h3 style="color:#FF0000">ATTENTION, ce fichier peut être à l'origine'
        . ' d'une attaque : ' . $HTTP_POST_FILES['fichier']['name'] . "!</h3>";
  }
?>
```

23.4 / La manipulation de fichiers distants

Il peut être utile de manipuler des fichiers à distance, c'est-à-dire par le biais des protocoles de transferts HTTP ou FTP.

Le langage PHP autorise **l'ouverture d'un fichier par l'intermédiaire d'une ardresse URL** dans la fonction *fopen* et depuis la version 4.3.0 avec également les fonctions d'inclusions *include()*, *include_once()*, *require()* et *require_once()*.

```
$id_fichier = fopen("http://www.site.com/index.html", "r");

include 'http://www.site.com/fichier.php?var=val';
```

A partir de ce moment, **toutes les informations contenues dans le fichier sont accessibles en lecture seule** dans une application PHP.

```
$taille = filesize("fichier.html");
echo str_replace("<", "&lt;", fread($id_fichier, $taille));
```

L'écriture sur un fichier distant est également possible, à condition de passer en argument une adresse FTP à la fonction *fopen* et que ce fichier soit nouveau.

```
$id_fichier = fopen("ftp://site.ftp.com/nouvelle_page.html", "w");
```

Evidemment, l'accès en écriture directement sur un site, nécessite souvent, **la saisie d'un nom d'utilisateur et d'un mot de passe** dans l'adresse afin d'éviter toutes intrusions inopportunes.

```
ftp://nom_utilisateur:mot_passe@ftp.site.com/nouvelle_page.html
```

La modification d'un fichier distant n'est pas réalisable par ce moyen.

Sous un environnement *Windows*, il est impossible d'utiliser les fichiers distants dans les expressions *include* et *require*.

Exemple [voir]

```
<?php
 function recherche_contenu($adresse)
 {
  $id_fichier = fopen($adresse, "r");
  if ($id_fichier)
  {
   $regexp = "<!-- Début contenu -->.*<!-- Fin contenu -->";
   $contenu = fread($id_fichier, filesize($adresse));
   if (eregi($regexp, $contenu, $donnee))
   {
    echo "<h3><u>Données contenu :</u></h3> "
        . str_replace("<", "&lt;", $donnee[0]);
   }
   else echo "<p>Impossible de trouver la chaîne.\n";
  }
  else echo "<p>Impossible d'ouvrir le fichier distant.\n";

  fclose($id_fichier);
 }

 recherche_contenu("http://www.site.com/page.html");
?>
```

23.5 / Les fonctions de système de fichiers

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur le système de fichiers.

Fonction
Description
$chaine = **basename**($chemin_fichier);
retourne le nom du fichier à partir de l'adresse du fichier spécifiée.
true | false = **chgrp**($nom_fichier, $groupe_proprietaire);
modifie le groupe propriétaire du fichier.
true | false = **chmod**($nom_fichier, $mode);
modifie le mode exprimé en nombre octal, du fichier.
true | false = **chown**($nom_fichier, $proprietaire);
modifie le groupe propriétaire du fichier.
clearstatcache();
efface la mémoire cache remplie par les fonctions *lsat* et *stat*.
true | false = **copy**($fichier, $nouveau_fichier);
copie un fichier vers une nouvelle destination.
delete($fichier);
efface le fichier.
$chaine = **dirname**($chemin);
retourne le nom du dossier parent.
$nombre = **disk_free_space**($dossier);
retourne l'espace disponible sur le disque sur lequel est le dossier.
$nombre = **diskfreespace**($dossier);
identique à *disk_free_space*.
$nombre = **disk_total_space**($dossier);
retourne la taille totale d'un dossier.
true | false = **fclose**($ID_fichier);
ferme un fichier indiqué par un identificateur retourné par *fopen* ou *fsockopen*.
true | false = **feof**($ID_fichier);
teste la fin du fichier.
true | false = **fflush**($ID_fichier);
écrit les données présentes dans la mémoire tampon (*buffer*), dans un fichier.
$chaine = **fgetc**($ID_fichier);
retourne le caractère sélectionné par le pointeur du fichier.
$tableau = **fgetcsv**($ID_fichier, $nombre, $chaine);
retourne la ligne courante et cherche les champs d'un fichier CSV (fichier de valeurs séparées par des virgules).
$chaine = **fgets**($ID_fichier, $longueur);
retourne la ligne courante jusqu'à soit un retour charriot, soit la fin du fichier, soit la longueur spécifiée.
$chaine = **fgetss**($ID_fichier, $longueur [, $balises]);
retourne la ligne courante à l'instar de *fgets* en supprimant les balises HTML et PHP ou juste celles spécifiées.
$tableau = **file**($chaine, $longueur);

lit le fichier et retourne le résultat dans un tableau.

file_exists($fichier [, $inclure_chemin]);

vérifie l'existance d'un fichier et éventuellement de son chemin si le second argument est égale à '1'.

$date | false = **fileatime**($fichier);

retourne la date du dernier accès sur le fichier.

$heure | false = **filectime**($fichier);

retourne l'heure du dernier accès sur le fichier.

$nombre | false = **filegroup**($fichier);

retourne le nom du groupe sous une forme numérique.

$nombre | false = **fileinode**($fichier);

retourne le numéro d'inode du fichier.

$date | false = **filemtime**($fichier);

retourne la date de dernière modification du fichier.

$nombre | false = **fileowner**($fichier);

retourne sous forme numérique, le nom du propriétaire du fichier.

$nombre | false = **fileperms**($fichier);

retourne sous forme numérique, les permissions affectées au fichier.

$nombre = **filesize**($fichier);

retourne la taille du fichier en octets.

$chaine = **filetype**($fichier);

retourne le type de fichier (*block*, *char*, *dir*, *fifo*, *file*, *link*, et *unknown*).

true | false = **flock**($ID_fichier, $nombre);

verrouille le fichier avec un nombre égal à '1', en écriture '2' ou le déverrouille '3'.

ID_fichier | false = **fopen**($fichier, $mode, $inclure_chemin);

ouvre un fichier ou une adresse URL selon un mode et éventuellement en incluant le chemin si le dernier argument est égal à '1'.

Mode	Description
r	ouvre le fichier en lecture seule.
r+	ouvre le fichier en lecture et en écriture.
w	ouvre le fichier en écriture seule ou tente de le créer s'il n'existe pas.
w+	ouvre le fichier en lecture et en écriture ou tente de le créer s'il n'existe pas.
a	ouvre le fichier en écriture seule ou tente de le créer s'il n'existe pas.
a+	ouvre le fichier en lecture et en écriture; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.
b	utilisable uniquement sous Windows, ouvre un fichier en mode binaire.

$nombre | false = **fpassthru**($ID_fichier);

lit le fichier du pointeur jusqu'à la fin et dirige le résultat vers la sortie standard.

$nombre = **fputs**($ID_fichier, $chaine [, $longueur]);

écrit la chaîne de caractères dans un fichier et éventuellement jusqu'à une longueur fournie.

$chaine = **fread**($ID_fichier, $longueur);

lit le fichier en mode binaire et éventuellement jusqu'à une certaine longueur.

$valeur = **fscanf**($ID_fichier, $format [, &$var, ..., &$varN]);

retourne les valeurs d'un fichier selon un format précis dans un tableau ou affecte ces valeurs aux variables spécifiées en renvoyant le nombre de valeurs affectées.

true | false = **fseek**($ID_fichier, $position);

déplace le pointeur de fichier à la position spécifiée.

$tableau = **fstat**($ID_fichier);

retourne des informations sur un fichier.

Mode	Description
1	volume
2	inode
3	mode de protection du inode
4	nombre de liens
5	id de l'utilisateur propriétaire
6	id du groupe propriétaire
7	type du volume de l'inode
8	taille en octets
9	date du dernier accès
10	date de la dernière modification
11	date du dernier changement
12	taille de bloc du système pour les entrées-sorties
13	Nombre de blocs alloués

$nombre | false = **ftell**($ID_fichier);

retourne la position du pointeur du fichier.

true | false = **ftruncate**($ID_fichier, $taille);

tronque un fichier à la taille spécifiée.

$nombre = **fwrite**($ID_fichier, $chaine [, $longueur]);

écrit en mode binaire, la chaîne de caractères dans un fichier et éventuellement jusqu'à une longueur fournie.

0 | EOF = **set_file_buffer**($ID_fichier, $taille);

détermine la taille de la mémoire tampon (*buffer*) utilisée en écriture dans le fichier.

true | false = **is_dir**($fichier);

vérifie si le nom du fichier est un dossier.

true | false = **is_executable**($fichier);

vérifie si le fichier est un exécutable.

true | false = **is_file**($fichier);

vérifie si le fichier en est effectivement un.

true | false = **is_link**($fichier);

vérifie si le fichier est un lien.

true | false = **is_readable**($fichier);

vérifie si le fichier est autorisé en lecture.

true | false = **is_writable**($fichier);

vérifie si le fichier est autorisé en écriture.

true | false = **is_writeable**($fichier);

vérifie si le fichier est autorisé en écriture.

true | false = **is_uploaded_file**($fichier);

vérifie si le fichier a bien été téléchargé par la méthode HTTP POST.

true | false = **link**($cible, $lien);

crée un lien.

$nombre | false = **linkinfo**($chemin);

retourne le champ *st_dev* de la structure d'information UNIX, à propos d'un lien.

true | false = **mkdir**($chemin, $mode_octal);

crée un dossier selon le chemin spécifié.

true | false = **move_uploaded_file**($fichier, $destination);

déplace un fichier téléchargé vers un emplacement spécifié.

$tableau = **parse_ini_file**($fichier [, $bool_section]);

retourne un tableau associatif contenant les champs et les valeurs d'un fichier de configuration **.ini*. Si le second argument est égal à *true*, un tableau multidimensionnel sera retourné avec pour clés les noms de section.

$tableau = **pathinfo**($chemin);

retourne des informations sur un chemin système sous forme d'un tableau associatif avec les clés *dirname*, *basename* et *extension*.

$nombre = **pclose**($ID_fichier);

ferme un processus de pointeur de fichier.

$nombre = **popen**($ID_fichier);

ouvre un processus de pointeur de fichier.

$nb_octets = **readfile**($fichier [, $inclure_dossier]);

lit un fichier et l'envoie à la sortie standard. Si le dernier argument vaut '1' alors la recherche du fichier inclut le dossier.

$chaine | false = **readlink**($lien);

retourne le nom du fichier vers lequel pointe le lien.

true | false = **rename**($nom_fichier, $nouveau_nom_fichier);

renomme un fichier.

true | false = **rewind**($ID_fichier);

replace le pointeur au début du fichier.

true | false = **rmdir**($chemin);

efface un dossier.

$tableau = **stat**($fichier);

retourne les informations à propos d'un fichier dans un tableau (voir *fstat*).

$tableau = **lstat**($fichier);

retourne les informations à propos d'un fichier ou d'un lien à l'instar de la fonction *stat*.

$chaine = **realpath**($chemin);

retourne le chemin absolu du chemin spécifié.

true | false = **symlink**($cible, $lien);

crée un lien.

$chaine \| NULL = **tempnam**($chemin, $prefixe);
crée un fichier temporaire unique dans le dossier spécifié.
$ID = **tmpfile**();
crée un fichier temporaire et retourne un identificateur semblable à celui de *fopen*.
true | false = **touch**($fichier, $date);
force la date de modification du fichier à la date spécifiée, par défaut à la date courante.
$nombre = **umask**([$nombre_octal]);
modifie le *umask* courant de PHP.
true | false = **unlink**($fichier);
efface un fichier.

23.6 / Les fonctions de dossiers

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les dossiers.

Fonction
Description
$nombre = **chroot**($chaine);
définit la chaîne de caractères comme la nouvelle racine.
$nombre = **chdir**($chaine);
définit le dossier en cours comme celui celui précisé par la chaîne de caractères.
$objet = **dir**($chaine);
crée un objet à partir du dossier spécifié.
closedir($identificateur_dossier);
ferme le dossier à partir d'un identificateur retourné par *opendir*.
$chaine = **getcwd**();
retourne le nom du dossier en cours.
$identificateur_dossier = **opendir**($chaine);
ouvre un dossier et retourne un identificateur de l'objet.
$chaine = **readdir**($identificateur_dossier);
retourne le fichier suivant dans le dossier spécifié par l'entremise de don identificateur.
rewinddir($identificateur_dossier);
retourne à la première entrée du dossier.

23.7 / Les fonctions FTP

Le langage PHP dispose de nombreuses fonctions permettant de travailler directement à partir du protocole de transfert de fichiers (FTP).

Fonction
Description
$id_connexion = **ftp_connect**($nom_hote [, $num_port]);
ouvre une connexion FTP.
true | false = **ftp_login**($id_connexion, $nom_utilisateur, $mot_passe);
authentifie une connexion FTP.
$nom_dossier = **ftp_pwd**($id_connexion);
retourne le nom du dossier en cours.
true | false = **ftp_cdup**($id_connexion);
change de dossier et passe au dossier parent.
true | false = **ftp_chdir**($id_connexion, $nom_dossier);
définit le dossier spécifié comme dossier en cours.
$nom \| false = **ftp_mkdir**($id_connexion, $nom_dossier);
crée un nouveau dossier et en retourne le nom.
true | false = **ftp_rmdir**($id_connexion, $nom_dossier);
efface un dossier.
$tab_fichiers = **ftp_nlist**($id_connexion, $nom_dossier);
retourne la liste des noms de fichiers d'un dossier dans un tableau.
$tableau = **ftp_rawlist**($id_connexion, $nom_dossier);
retourne une liste détaillée des fichiers d'un dossier dans un tableau.
$chaine = **ftp_systype**($id_connexion);
retourne un identifiant de type du serveur FTP.
true | false = **ftp_pasv**($id_connexion, true | false);
active (*true*) ou désactive (*false*) le mode passif.
true | false = **ftp_get**($id_connexion, $fichier_local, $fichier_distant, $mode);
télécharge un fichier distant à partir du serveur FTP sous un nom local. Le mode peut prendre l'une de ces valeurs : *FTP_ASCII* ou *FTP_BINARY*
true | false = **ftp_fget**($id_connexion, $id_fichier_local, $fichier_distant, $mode);
télécharge un fichier distant à partir du serveur FTP et le sauvegarde dans un fichier local.
true | false = **ftp_put**($id_connexion, $fichier_distant, $fichier_local, $mode);
télécharge un fichier local sous un nom distant sur un serveur FTP.
true | false = **ftp_fput**($id_connexion, $fichier_distant, $id_fichier_local, $mode);
télécharge un fichier local sous un nom distant sur un serveur FTP.
$taille = **ftp_size**($id_connexion, $fichier_distant);
retourne la taille d'un fichier sur le serveur FTP.
$date = **ftp_mdtm**($id_connexion, $fichier_distant);
retourne la date de dernière modification d'un fichier sur un serveur FTP.

true | false = **ftp_rename**($id_connexion, $nom_fichier, $nouveau_nom);
renomme un fichier sur un serveur FTP.
true | false = **ftp_delete**($id_connexion, $chemin);
supprime un fichier sur un serveur FTP.
true | false = **ftp_site**($id_connexion, $chaine_commande);
envoie une commande au serveur FTP.
true | false = **ftp_quit**($id_connexion);
ferme une connexion FTP.

Exemples [voir]

```
<?php
 $nomrep = 'undossier';
 $nomfichier = 'unepagehtml';
 $contenu = 'un contenu quelconque, mais extrêmement important...';
 $nom_hote = 'ftp.serveur.net';
 $num_port = 21;
 $nom_utilisateur = 'nom';
 $mot_passe = 'pAssWorD';
 //Ouverture d'une connexion à l'hôte FTP
 $id_connexion = ftp_connect($nom_hote, $num_port);
 //Connexion de l'utilisateur
 ftp_login($id_connexion, $nom_utilisateur, $mot_passe);
 //Accès au répertoire 'www'
 ftp_chdir($id_connexion, 'www');
 //Création du répertoire 'undossier' puis 'temp'
 if (!file_exists($nomrep) && !is_dir($nomrep))
   ftp_mkdir($id_connexion, $nomrep);
 $nom = 'temp';
 if (!file_exists($nom) && !is_dir($nom)){
   ftp_mkdir($id_connexion, $nom);
 }
 echo '<h3>Répertoire : '.ftp_pwd($id_connexion).'</h3>';
 $fichier = $nomrep.'/'.$nomfichier.'.html';
 //Création d'un fichier temporaire
 $tmp = tempnam($nom, "fichier");
 echo '<h3>Fichier temporaire : '.$tmp.'</h3>';
 //Ouverture du fichier temporaire
 $id_f = fopen($tmp, "w")
 //Ecriture dans le fichier temporaire
 fwrite($id_f, $contenu.'<br>'.$contenu.'<br>'.$contenu);
 echo '<h3>Taille du fichier : '.filesize($tmp).'</h3>';
 echo '<h3>Affichage du fichier :</h3>';
 readfile($tmp);
 //Fermeture du fichier temporaire
 fclose($id_f);
 //Réouverture du fichier temporaire en lecture seule
 $id_f = fopen($tmp, "r");
 //Téléchargement du fichier temporaire vers son nouvel emplacement
 ftp_put($id_connexion, $fichier, $tmp, FTP_BINARY);
 echo '<h3>Identificateur de fichier : '.$id_f.'</h3>';
 echo '<h3>Date de modification du fichier : '
          .ftp_mtdm($id_connexion, $fichier).'</h3>';
 //Fermeture du fichier temporaire
 fclose($id_f);
 //Fermeture de la connexion FTP
 ftp_quit($id_connexion);
?>
```

23.8 / Les fonctions ZLIB

Le langage PHP dispose de nombreuses fonctions permettant d'utiliser les méthodes de dompression et de décompression de fichiers proposées par la librairie ZLIB.

- Les fonctions Zlib
- Les fonctions ZIP

Fonction
Description
ZLib
true | false = **gzclose**($id_fichier_zip);
ferme la ressource d'un fichier compressé.
true | false = **gzeof**($id_fichier_zip);
teste la fin d'un fichier compressé.
$tableau = **gzfile**($id_fichier_zip [, include_path]);
retourne la totalité d'un fichier compressé lu dans un tableau. Le second argument permet de rechercher un fichier dans le dossier spécifié par *include_path* (include_path=".[{;|:}chemin_dossier{;|:}...]") dans le fichier de configuration de PHP.
$caractere \| false = **gzgetc**($id_fichier_zip);
retourne un caractère lu dans un fichier compressé.
$chaine \| false = **gzgets**($id_fichier_zip, $longueur);
retourne une chaîne de caractères d'une certaine longueur d'un fichier compressé.
$chaine \| false = **gzgetss**($id_fichier_zip, $longueur [, $balises]);
retourne une chaîne de caractères d'une certaine longueur d'un fichier compressé et supprime les balises HTML hormis celle qui sont spécifiées.
$id_fichier_zip = **gzopen**($fichier, $mode [, include_path]);
ouvre un fichier compressé selon un mode de lecture (r, rb) et/ou d'écriture (w, wb), à partir d'un fichier. Un ou plusieurs autres caractères peut être utilisé immédiatement après le mode afin d'indiquer la méthode de compression (wb**9** : compression maximale, wb**1** : compression minimale, wb6**h** : compresion Huffman, wb6**f** : filtrage des données). Pour le dernier argument voir *gzfile*.
true | false = **gzpassthru**($id_fichier_zip);
affiche toutes les informations d'un fichier compressé.
$nombre = **gzputs**($id_fichier_zip, $chaine [, $longueur]);
écrit une chaîne de caractères dans un fichier compressé.
$chaine = **gzread**($id_fichier_zip, $longueur);
retourne une chaîne de caractères lue dans un fichier compressé en mode binaire.
true | false = **gzrewind**($id_fichier_zip);
replace le pointeur courant au début du fichier.
true | false = **gzseek**($id_fichier_zip, $position);
déplace le pointeur courant dans un fichier compressé jusqu'à une certaine position.
$position = **gztell**($id_fichier_zip);
retourne la position courante du pointeur interne.
$nombre = **gzwrite**($id_fichier_zip, $chaine [, $longueur]);
écrit une chaîne de caractères dans un fichier compressé en mode binaire.

$nb_octets_lus \| false = **readgzfile**($id_fichier_zip [, 1]);
affiche le contenu d'un fichier compressé.
$chaine_compressee \| false = **gzcompress**($chaine [, niveau]);
compresse une chaîne de caractères dans le format ZLIB selon un niveau de compression compris entre 0 et 9.
$chaine_decompressee \| false = **gzuncompress**($chaine, $longueur);
décompresse une chaîne de caractères ZLIB jusqu'à une certaine longueur.
$chaine_compressee \| false = **gzdeflate**($chaine [, niveau]);
compresse une chaîne de caractères dans le format DEFLATE selon un niveau de compression compris entre 0 et 9.
$chaine_decompressee \| false = **gzinflate**($chaine, $longueur);
décompresse une chaîne de caractères DEFLATE jusqu'à une certaine longueur.
$chaine_compressee \| false = **gzencode**($chaine [, $niveau]);
crée une chaîne de caractères compressée avec gzip selon un niveau de compression compris entre 0 et 9.
ZIP
zip_close($id_fichier_zip);
ferme une archive Zip ouverte par *zip_open*.
zip_entry_close($id_element_zip);
ferme un élément d'archive ouvert par *zip_read*.
$taille = **zip_entry_compressedsize**($id_fichier_zip);
retourne la taille compressée de l'entrée archivée.
$methode = **zip_entry_compressionmethod**($id_element_zip);
retourne la méthode de compression de l'élément d'archive.
$taille = **zip_entry_filesize**($id_element_zip);
retourne la taille réelle de l'élément d'archive.
$nom = **zip_entry_name**($id_element_zip);
retourne le nom de l'élément d'archive.
true | false = **zip_entry_open**($id_fichier_zip, $id_entree_zip [, $mode]);
ouvre un nouvel élément dans une archive. L'argument *$mode* ne peut valoir actuellement que *rb*.
$chaine \| false = **zip_entry_read**($id_element_zip [, $longueur]);
retourne les données lues dans une archive jusqu'à une certaine longueur.
$id_fichier_zip = **zip_open**($fichier);
ouvre une archive Zip et retourne un identifiant.
$id_element_zip = **zip_read**($id_fichier_zip);
lit le prochain élément d'une archive et retourne un identifiant.

Exemples [voir]

```
<html>
 <body>
  <?php
   $fichier = tempnam('C:\www\laltruiste\temp', 'archive').'.gz';

   $chaine = "Un contenu quelconque à sauvegarder !";
   //Ouverture le fichier en écriture avec une compression maximale wb9
   $id_arch = gzopen($fichier, "wb9");
   //Sauvegarde d'un contenu dans l'archive
   gzwrite($id_arch, $chaine);
   //Fermeture de l'archive
   gzclose($id_arch);

   //Réouverture de l'archive en lecture
   $id_arch = gzopen($fichier, "r");
   //Lecture de 10 caractères
   echo gzread($id_arch, 10);
   //Retour du pointeur de fichier au début de l'archive
   gzrewind($id_arch);
   //Affichage puis fermeture du contenu de l'archive
   gzpassthru($id_arch);

   //Ouverture et affichage de l'archive
   if(readgzfile($fichier) != strlen($chaine))
    echo 'Impossible de lire l\'archive '.$fichier.'!';
  ?>
 </body>
</html>
```

24 / La gestion des erreurs

Le langage PHP propose divers moyens permettant l'interception des erreurs se produisant dans un script.

Plusieurs fonctions sont disponibles dans PHP, afin de gérer efficacement n'importe quelle type d'erreur.

Ainsi, il devient possible d'**adapter une réponse personnalisée lorsqu'intervient une erreur** dans une application.

```
trigger_error ("Message d'erreur personnalisée",
                                        E_USER_ERROR);

error_reporting (ERREUR_FATALE | ERREUR | AVERTISSEMENT);
```

Les erreurs peuvent être dûes à diverses raisons telles que, des fautes de syntaxes, des problèmes d'exécutions ou encore à une mauvaise compilation.

Seules certaines erreurs possèdent un message défini par PHP, il s'agît des erreurs de type *E_ERROR*, *E_WARNING*, et *E_PARSE*.

```
Numéro d'erreur  : 8
Message d'erreur : Undefined variable
```

Non seulement il est donc possible de personnaliser les erreurs en conformité avec les spécificités d'une application, mais il est également possible de **sauvegarder les erreurs dans un fichier journal ou de les envoyer directement à un responsable** de la page fautive.

```
error_log ("Attention une erreur s'est produite !",
                    1, "responsable@omaine.com");
```

24.1 / Les types d'erreur

Les erreurs peuvent être classifiées selon leur type et leur gravité.

Les erreurs peuvent être susceptibles de **provoquer simplement des avertissements sans affecter le programme en cours** (*WARNING* et *NOTICE*), d'autres **entraînent l'interruption définitive du script** (ERROR).

Les erreurs sont classifiées selon cinq types différents :

- les erreurs à l'**initialisation** (*E_CORE_ERROR*, *E_CORE_WARNING*),
- les erreurs de **compilation** (*E_COMPILE_ERROR*, *E_COMPILE_WARNING*),
- les erreurs d'**analyse syntaxique** (*E_PARSE*),
- les erreurs d'**exécution** (*E_ERROR*, *E_WARNING*, *E_NOTICE*),
- les erreurs **utilisateurs** (*E_USER_ERROR*, *E_USER_WARNING*, *E_USER_NOTICE*)

Type	valeur	Description
E_ERROR	1	représente une erreur impossible à corriger. Elle est différente d'une erreur d'analyse.
E_WARNING	2	représente un message d'alerte provoquée par une erreur qui n'interrompt pas le script.
E_PARSE	4	représente une erreur d'analyse dans le domaine syntaxique dont la correction est impossible.
E_NOTICE	8	représente un avertissement ou une erreur n'ayant pas provoqué un arrêt du script.
E_CORE_ERROR	16	représente un avertissement ou une erreur n'ayant pas provoqué un arrêt du script.
E_CORE_WARNING	32	représente un avertissement ou une erreur n'ayant pas provoqué un arrêt du script.
E_COMPILE_ERROR	64	représente une erreur de compilation provoquant l'interruption du script.
E_COMPILE_WARNING	128	représente un message d'avertissement provenant du compilateur sans interrompre le script.
E_USER_ERROR	256	représente une erreur due à l'utilisateur provoquant l'interruption du script.
E_USER_WARNING	512	représente un message d'avertissement dû à l'utilisateur ne provoquant pas l'arrêt du script.
E_USER_NOTICE	1024	représente un message d'avertissement ou une erreur n'ayant pas provoqué l'arrêt du script dû à l'utilisateur.
E_ALL	2047	représente toutes les constantes E_....
E_STRICT	2048	permet d'obtenir des notifications pour la modification du code (Depuis PHP 5).

Les valeurs d'erreur peuvent être combinées dans les fonctions par l'intermédiaire des opérateurs au niveau du bit.

```
error_reporting (E_ERROR | E_WARNING | E_PARSE);

error_reporting (1 | 2 | 4);

// (0001 | 0010 | 0100 = 0111)2 correspond à (7)10
error_reporting(7);
```

Le fichier de configuration de PHP 3 *php3.ini* n'accepte quant à lui qu'une valeur numérique devant l'option *error_reporting*, puisque les constantes n'y sont pas utilisables, ce qui en revanche, n'est pas le cas pour PHP 4, paramétré par défaut à *E_ALL*.

```
[PHP]
```

```
error_reporting= E_ALL; #display all errors, warnings and notices
```

24.2 / Les fonctions d'erreurs

Le langage PHP dispose de nombreuses fonctions permettant de gérer les erreurs.

Fonction
Description
$tableau = **debug_backtrace**();

génère un contexte de déboguage et retourne un tableau associatif.

Nom	Type	Description
function	String	contient le nom de la fonction courante (voir *__FUNCTION__*).
line	Integer	contient le numéro courant de ligne (voir *__LINE__*).
file	String	contient le nom du fichier courant (voir *__FILE__*).
class	String	contient le nom de la classe courante class (voir *__CLASS__*).
type	String	contient le type de classe courante. Si une méthode est appelée, "->" est retourné. Si une méthode statique est appelé, "::" est retourné. Si une fonction est appelée, rien ne sera retourné.
args	Array	contient la liste des arguments ou la liste des fichiers si respectivement le contexte est à l'intérieur d'une fonction ou inclus si dans un fichier inclus.

debug_print_backtrace();

affiche une trace de déboguage.

error_log($message, $type_msg [, $destination [, $entete]);

envoie un message d'erreur d'un certain type, au fichier log du serveur web, à un port TCP ou à un autre fichier.

Type	Description
0	envoi à l'historique d'erreur PHP.
1	envoi d'un courrier électronique à l'adresse destination avec éventuellement un entête.
2	envoi par la connexion de debuggage PHP si remote_debugging a été désactivée. Le paramètre destination indique l'hôte ou l'adresse IP et éventuellement le numéro de port.
3	ajout au fichier destination.

$nombre = **error_reporting**($niveau);

indique le niveau de rapport d'erreurs PHP. Le niveau peut être l'une de ces valeurs ou leur combinaison.

Type	Description
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING

1024 E_USER_NOTICE
2047 E_ALL
2048 E_STRICT
restore_error_handler();
réactive l'ancienne fonction de gestion des erreurs.
restore_exception_handler();
réactive l'ancienne fonction de gestion d'exceptions (PHP 5).
$chaine = **set_error_handler**(gestionnaire_erreur);
sélectionne une fonction comme gestionnaire d'erreurs.
$fonction \| false = **set_exception_handler**($fonction);
détermine une fonction de gestion d'exceptions (PHP 5).
trigger_error($message [, $type]);
déclenche une erreur utilisateur.
user_error($message [, $type]);
génère un message d'erreur utilisateur.

24.3 / Exemple de gestion d'erreur

Exemple [voir]

```php
<?php
 define ("ERREUR_FATALE", E_USER_ERROR);
 define ("ERREUR", E_USER_WARNING);
 define ("AVERTISSEMENT", E_USER_NOTICE);

 error_reporting (ERREUR_FATALE | ERREUR | AVERTISSEMENT);

 function gestionnaire_erreur ($numero_erreur, $message_erreur,
                               $fichier_erreur, $ligne_erreur)
 {
  switch ($numero_erreur)
  {
   case ERREUR_FATALE:
    echo "<h3 style='color:#FF0000'>Une erreur du type "
         . "<i>E_USER_ERROR</i> s'est produite :</h3>"
         . "<u>Numéro :</u> " . $numero_erreur
         . "<br><u>Message :</u> " . $message_erreur
         . "<br><u>Fichier :</u> " . $fichier_erreur
         . "<br><u>Ligne :</u> " . $ligne_erreur
         . "<br>Interruption du script en cours !";
    exit -1;
    break;

   case ERREUR:
    echo "<h3 style='color:#FF0000'>Une erreur du type "
         . "<i>E_USER_WARNING</i> s'est produite :</h3>"
         . "<u>Numéro :</u> " . $numero_erreur
         . "<br><u>Message :</u> " . $message_erreur. "<br>";
    break;

   case AVERTISSEMENT:
    echo "<h3 style='color:#FF0000'>Une erreur du type "
         . "<i>E_USER_NOTICE</i> s'est produite :</h3>"
         . "<u>Numéro :</u> " . $numero_erreur
         . "<br><u>Message :</u> " . $message_erreur . "<br>";
    break;

   default:
    echo "<h3 style='color:#FF0000'>Une erreur d'un type "
         . "inconnu s'est produite :</h3>"
         . "<u>Numéro :</u> " . $numero_erreur
         . "<br><u>Message :</u> " . $message_erreur . "<br>";
    break;
  }
 }

 function calc_moy ($note, $total)
 {
  $temp = 0;
  if ( !is_numeric($total) || $total != sizeof($note))
     trigger_error("Le nombre de notes n'est pas un nombre ou "
                 . "n'est pas égal au total des notes : " . $total, ERREUR_FATALE);
  if (!is_array($note))
  {
   trigger_error("Un tableau de notes est attendu : " . $note . " !", ERREUR);
   return null;
  }
  for ($i = 0; $i < sizeof($note); $i++)
  {
   if (!is_numeric($note[$i]))
   {
    trigger_error("La valeur à la position " . $i . "
                 . "n'est pas un nombre : " . $note[$i] . " !", AVERTISSEMENT);
    $total--;
   }
   else
   {
    $temp += $note[$i];
    $moyenne = $temp/$total;
   }
  }
  return number_format($moyenne, 2, ',', ' ');
 }
```

```
$gestion_erreur = set_error_handler("gestionnaire_erreur");

echo "<h4>Création d'un tableau avec des notes.</h4>";
$tableau = array(11.25, 14.5, "AB", 15.75, 18, 9.75, 10.25, "C");
echo "tableau[";
for($i = 0; $i < sizeof($tableau); $i++)
{
  $sep = $i == sizeof($tableau) - 1 ? "]<br>" : ", ";
  echo $tableau[$i] . $sep;
}
echo "################################<br>";

echo "<h4>Appel de fonctions provoquant plusieurs avertissements</h4>";
$moy = calc_moy($tableau, sizeof($tableau));
echo "Moyenne : " . $moy;
echo "<br>################################<br>";

echo "<h4>Argument erroné provoquant une erreur.</h4>";
$moy = calc_moy("20, 11, 8, 12.85", 1);
var_dump($moy);
echo "<br>################################<br>";

$tableau = array(11.25, 14.5, 15.75, 18, 9.75, 10.25);
echo "<h4>Appel de fonctions provoquant une erreur d'un type inconnu.</h4>";
$moy = calc_moy($tableau, sizeof($tableau));
echo "Moyenne : " . $moyenne;
echo "<br >################################<br>";

echo "<h4>Argument erroné provoquant une erreur fatale.</h4>";
$moy = calc_moy($tableau, "A");
echo "################################<br>";
?>
```

24.4 / Le préfixe @

Le préfixe arobase @ devant une expression, entraîne l'annulation du rapport d'erreur de cette expression tout en conservant les messages d'erreur dues aux erreurs d'analyse.

```
@file("http://www.site.com/fichier.txt");
```

Toutefois, si une erreur se produisait alors que **l'option de suivi des erreurs dans *php.ini* serait activée** (*track_errors TRUE*), il sera possible de **récupérer le message d'erreur** dans la variable globale, *$PHP_ERRORMSG*.

```
Message : <?php echo $PHP_ERRORMSG ?>.
```

L'opérateur de désactivation de rapport d'erreur @ agît pour toutes les erreurs ce qui pourrait **provoquer un arrêt impromptu du script sans aucune indication**. C'est pour cela, qu'il est nécessaire d'utiliser très précautionneusement cet opérateur afin d'éviter tout problème inopportun.

Exemple [voir]

```
<?
 function calc_moy($note, $total)
 {
  $temp = 0;
  if ( !is_numeric($total) || $total != sizeof($note))
   print ("Le nombre de notes n'est pas un nombre "
          .  "ou n'est pas égal au total des notes : " . $total);
  if (!is_array($note))
  {
   print ("Un tableau de notes est attendu : " . $note . " !");
   return null;
  }
  for ($i = 0; $i < sizeof($note); $i++)
  {
   if (!is_numeric($note[$i]))
   {
    print ("La valeur à la position " . $i
           . " n'est pas un nombre : " . $note[$i] . " !");
    $total--;
   }
   else
   {
    $temp += $note[$i];
    $moyenne = $temp/$total;
   }
  }
  return number_format($moyenne, 2, ',', ' ');
 }

 echo "<h4>Création d'un tableau avec des notes.</h4>";
 $tableau = array(11.25, 14.5, "AB", 15.75, 18, 9.75, 10.25, "C");

 echo "tableau[";
 for($i = 0; $i < sizeof($tableau); $i++)
 {
  $sep = $i == sizeof($tableau) - 1 ? "]<br>" : ", ";
  echo $tableau[$i] . $sep;
 }

 echo "<h4>Appel de la fonction <calc_moy()</i></h4>";
 $moy = @calcul_moyenne($tableau, sizeof($tableau));
 echo "<u>Moyenne :</u> " . $moy;
?>
```

24.5 / Les erreurs HTTP 404

Un message d'erreur du type *HTTP 404* correspond à un fichier non trouvé car l'adresse n'est pas valide ou le fichier n'existe pas sur le serveur.

Dans le cas ou le serveur web du site est Apache, alors il suffit de **copier un fichier dénommé *.htaccess* sous la racine du site**.

Le fichier *.htaccess* doit comporter une ligne spécifiant le fichier personnalisé auquel le système devra accéder en cas d'erreur *HTTP 404*.

ErrorDocument 404 /error.php

Ainsi, lorsqu'un utilisateur essaiera d'accéder à une page par un lien invalide, **la page *error.php* s'ouvrira en affichant un contenu personnalisé**.

L'adresse URL fautive peut être précisée dans la page en utilisant une variable spéciale *$REDIRECT_URL*.

L'adresse URL *<? echo $REDIRECT_URL?>* n'est pas valide.

Par ailleurs, **le fichier *error.php* peut contenir diverses fonctionnalités** comme une redirection automatique sur la page précédente avec un envoi d'un rapport d'erreur à la boite email d'un responsable ou à un fichier journal ou encore à une base de données.

Exemple [voir]

```
<html>
  <body>
    <?php
      $url_precedente = $HTTP_REFERER;

      function redirection($adresse_url, $temps)
      {
        print('<meta http-equiv="refresh" content="'
              . $temps . ';URL='.$adresse_url.'">');
      }
    ?>
    <h2>Le fichier sollicitée n'existe pas</h2>
    <p style="color:red">Adresse URL : <?php echo $REDIRECT_URL ?></p>

    <h3>Nous sommes désolés pour ce désagrément.</h3>
    <p>Dans 5 secondes, la page sera automatiquement redirigée vers : </p>
    <blockquote>
      <a href="coursphp/<?php echo $url_precedente ?>">
        <?php echo $url_precendente ?>
      </a>
    </blockquote>
    <a href="<?php echo $url_precedente ?>">Retour à la page suivante</a>

    <a href="mailto:webmaster@domaine.com">Contactez le webmestre</a>
    <?
      redirection($url_precedente, 5);
    ?>
  </body>
</html>
```

25 / Les fonctions prédéfinies

x

25.1 / Les fonctions d'options et d'informations PHP

Le langage PHP dispose de nombreuses fonctions relatives aux options et aux informations PHP.

Fonction
Description
$nombre = **assert**($assertion);
vérifie si une assertion booléenne ou littérale, est fausse afin de prendre les mesures appropriées.
$valeur \| false = **assert-options**($nombre [, $valeur]);
fixe et lit différentes options d'assertions.

Option	Paramètre ini	Val. par déf.
Description		
ASSERT_ACTIVE	**assert.active**	**1**
active l'évaluation de la fonction assert()		
ASSERT_WARNING	**assert.warning**	**1**
génère une alerte PHP pour chaque assertion fausse		
ASSERT_BAIL	**assert.bail**	**0**
termine l'exécution en cas d'assertion fausse		
ASSERT_QUIET_EVAL	**assert.quiet_eval**	**0**
inactive le rapport d'erreur durant l'évaluation d'une assertion		
ASSERT_CALLBACK	**assert_callback**	**null**
fonction utilisateur de traitement des assertions fausses		

true | false = **extension_loaded**($nom_extension);
vérifie si une extension est chargée.
$nombre = **dl**($nom_librairie);
charge une librairie PHP à la volée.
$chaine = **getenv**($nom_variable);
retourne la valeur de la variable d'environnement.
$chaine = **get_cfg_var**($nom_variable);
retourne la valeur d'une option de PHP.
$chaine = **get_current_user**();
retourne le nom du propriétaire du script en cours.
$tableau = **get_defined_constants**();
retourne la liste des constantes et leur valeur dans un tableau.
$tableau = **get_extension_funcs**($nom_extension);
retourne la liste de fonctions d'une extension dans un tableau.
$GID | false = **getmygid**();
retourne le GID du propriétaire du script.
$tableau = **get_included_files**();
retourne un tableau avec les noms de fichiers inclus dans un script.
$tableau = **get_loaded_extensions**();
retourne dans un tableau, la liste de tous les modules compilés et chargés.
$tableau = **get_required_files**();
retourne un tableau comprenant les noms de fichiers requis et inclus dans un script.
$noombre = **get_magic_quotes_gpc**();
retourne la configuration courante de l'option *magic_quotes_gpc*.
$nombre = **get_magic_quotes_runtime**();

retourne la configuration courante de l'option *magic_quotes_runtime*.

$date = **getlastmod**();

retourne la date de dernière modification de la page.

$nombre | false = **getmyinode**();

retourne l'inode du script.

$nombre | false = **getmypid**();

retourne le numéro de processus en cours.

$UDI | false = **getmyuid**();

retourne l'UID du propriétaire du script en cours.

$tableau = **getrusage**([$nombre]);

retourne le niveau d'utilisation des ressources dans un tableau.

$chaine = **ini_alter**($nom_variable, $nouvelle_valeur_chaine);

modifie la valeur littérale d'une option de configuration.

$chaine = **ini_get**($nom_variable);

retourne la valeur d'une option de configuration.

$tableau = **ini_get_all**($nom_extension);

retourne toutes les valeurs de configuration dans un tableau.

$chaine = **ini_restore**($nom_variable);

restaure la valeur de l'option de configuration.

$chaine = **ini_set**($nom_variable, $nouvelle_valeur_chaine);

modifie la valeur d'une option de configuration.

phpcredits([$parametre]);

affiche les crédits de PHP.

$nombre = **phpinfo**([$parametre]);

affiche toutes les informations à propos de PHP. L'argument permet d'afficher seulement certaines informations (*INFO_GENERAL*, *INFO_CREDITS*, *INFO_CONFIGURATION*, *INFO_MODULES*, *INFO_ENVIRONMENT*, *INFO_VARIABLES*, *INFO_LICENSE* et *INFO_ALL*).

$chaine = **phpversion**();

retourne le numéro de la version courante de PHP.

$GUID = **php_logo_guid**();

retourne le logo GUID.

$chaine = **php_sapi_name**();

retourne le type d'interface utilisé entre le serveur web et PHP.

$chaine = **php_uname**();

retourne les informations sur le système d'exploitation.

putenv($nom_variable);

fixe la valeur d'une variable d'environnement durant la vie du script en cours.

$nombre = **set_magic_quotes_runtime**(0 | 1);

active ('1') ou désactive ('0') l'option *magic_quotes_runtime*.

set_time_limit($nb_secondes);

fixe le temps maximum d'exécution d'un script.

$chaine = **zend_logo_guid**();
Retourne le logo de Zend.
$nombre = **version_compare**($version, $autre_version, $operateur_comparaison);
compare deux versions de PHP.
$chaine = **zend_version**();
retourne la version en cours du moteur PHP Zend.

25.2 / Les fonctions Apache

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les chaines de caractères.

Fonction
Description
true | false = **apache_child_terminate**();
Termine un processus Apache après une requête.
$entier = **ascii2ebcdic**($Chaine_ASCII);
Transforme une chaîne ASCII en EBCDIC.
$entier = **ebcdic2ascii**($chaine_EBCDIC);
Transforme une chaîne EBCDIC en ASCII.
$Objet = **apache_lookup_uri**($Chaine);
Effectue une requête partielle pour l'URI spécifiée et renvoie toutes les informations.
$chaine = **apache_note**($nom_note [, $note_valeur]);
Affiche ou affecte le paramètre "apache request notes".
$tableau = **getallheaders**();
Récupère toutes les en-têtes des requêtes HTTP.
$entier = **apache_setenv**($var_chaine, $valeur_chaine [, $booleen]));
Affecte une variable Apache 'subprocess_env'.
$entier = **virtual**($nom_fichier);
Effectue une sous-requête Apache.

25.3 / Les fonctions d'exécution

Le langage PHP dispose de nombreuses fonctions permettant d'exécuter des programmes externes.

Fonction
Description
$chaine = **escapeshellarg**($chaine);
échappe une chaîne de caractères pour une utilisation en ligne de commande.
$chaine = **escapeshellcmd**($chaine);
échappe les méta-caractères Shell.
$chaine = **exec**($chaine_commande [, $tab_resultat [, $var_resultat]]);
exécute un programme externe et place éventuellement les résultats dans un tableau ou/et dans une variable.
passthru($chaine_commande [, $var_resultat]);
exécute un programme externe et affiche le résultat brut.
$chaine = **system**($chaine_commande [, $var_resultat]);
exécute un programme externe et affiche le résultat.

25.4 / Les fonctions d'adresse URL

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les adresses URL (Uniform esource Locator).

Fonction
Description
$chaine = **base64_decode**($donnee_encodee);
décode une chaîne de caractères en MIME base64.
$chaine_encodee = **base64_encode**($donnee);
encode une chaîne de caractères en MIME base64.
$tableau = **parse_url**($chaine_URL);
analyse une adresse URL et retourne ses composants dans un tableau associatif (clés : scheme, host, port, user, pass, path, query, et fragment).
$chaine = **rawurldecode**($chaine_URL);
décode une chaîne de caractères encodée en URL.
$chaine_URL = **rawurlencode**($chaine);
encode une chaîne de caractères en adresse URL.
$chaine = **urldecode**($chaine_URL);
décode une chaîne de caractères encodée en adresse URL.
$chaine_URL = **urlencode**($chaine);
encode une chaîne de caractères en adresse URL.

25.5 / Les fonctions d'images

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les images.

Fonction
Description
$tableau = **getimagesize**($fichier [, $tab_info]);
retourne la taille d'une image et un couple de balises HTML. Le tableau contient la largeur (index '0'), la hauteur ('1'), le type de l'image ('2') dont les valeurs sont : 1 = GIF, 2 = JPG, 3 = PNG et enfin le jeu de balises HTML ("height='hauteur' width='largeur'").
image2wbmp($ID_img [, $fichier [, $niveau_seuil]]);
crée une image WBMP et la conserve dans un fichier ou l'affiche dans le navigateur.
$nombre = **ImageAlphaBlending**($ID_img, $mode_blending);
modifie le mode de fondu (*blending*) d'une image si le mode est égal à *true*.
$nombre = **ImageArc**($ID_img, , $pos_x, $pos_y, largeur, hauteur, $angle_début, $angle_fin, $couleur);
dessine une ellipse partielle.
$nombre = **imagefilledarc**($ID_img, , $pos_x, $pos_y, $largeur, $hauteur, $angle_début, $angle_fin, $couleur, $style);
dessine une ellipse partielle et la remplit. L'argument style prend les valeurs suivantes :

Style	Constante	Description
1	**img_ARC_CHORD**	connecte les angles de début et de fin avec une ligne droite.
2	**img_ARC_PIE**	produit une ligne courbe.
3	**img_ARC_NOFILL**	indique que l'arc doit être dessiné mais non rempli.
4	**img_ARC_EDGED**	utilisé conjointement avec IMG_ARC_NOFILL, indique que les angles de début et de fin doivent être connectés au centre.

$nombre = **ImageEllipse**($ID_img, , $pos_x, $pos_y, $largeur, $hauteur, $couleur);
dessine une ellipse complète.
$nombre = **ImageFilledEllipse**($ID_img, , $pos_x, $pos_y, $largeur, $hauteur, $couleur);
dessine une ellipse pleine avec une certaine couleur.
$nombre = **ImageChar**($ID_img, $police, $pos_x, $pos_y, $chaine, $couleur);
dessine horizontalement le premier caractère de la chaîne.
ImageCharUp($ID_img, $police, $pos_x, $pos_y, $chaine, $couleur);
dessine verticalement le premier caractère de la chaîne.
$nombre = **ImageColorAllocate**($ID_img, $rouge, $vert, $bleu);
alloue une couleur à une image.
$nombre = **ImageColorDeAllocate**($ID_img, $rouge, $vert, $bleu);
désalloue une couleur à une image.
$nombre = **ImageColorAt**($ID_img, $pos_x, $pos_y);
retourne l'index de la couleur à la position du pixel spécifié.
$nombre = **ImageColorClosestAlpha**($ID_img, $rouge, $vert, $bleu, $alpha);
retourne la couleur la plus proche en tenant compte du canal alpha.
$nombre = **ImageColorClosest**($ID_img, $rouge, $vert, $bleu);
retourne l'index de la couleur la plus proche d'une couleur spécifiée.
$nombre = **ImageColorExact**($ID_img, $rouge, $vert, $bleu);
retourne l'index de la couleur indiquée.
$nombre = **ImageColorExactAlpha**($ID_img, $rouge, $vert, $bleu, $alpha);

retourne l'index d'une couleur avec son canal alpha.

```
$nombre = ImageColorResolve($ID_img, $rouge, $vert, $bleu);
```

retourne l'index de la couleur donnée ou la plus proche possible.

```
$nombre = ImageColorResolveAlpha($ID_img, $rouge, $vert, $bleu, $alpha);
```

retourne un index de couleur ou son alternative la plus proche avec le canal alpha.

```
$nombre = ImageGammaCorrect($ID_img, $entree_gamma, $sortie_gamma);
```

applique une correction gamma à l'image.

```
true | false = ImageColorSet($ID_img, $index, $rouge, $vert, $bleu);
```

modifie la couleur dans une palette à l'index donné.

```
$tab_couleur = ImageColorsForIndex($ID_img, $index);
```

retourne la couleur associée à un index dans un tableau associatif dont les clés sont : *red*, *green*, *blue*.

```
$nb_couleur = ImageColorsTotal($ID_img);
```

retourne le nombre de couleurs d'une image.

```
ID_nouvelle_couleurImageColorTransparent($ID_img [, $ID_couleur]);
```

choisit la couleur transparente et retourne l'identificateur de la nouvelle couleur.

```
$nombre = ImageCopy($ID_img_dest,$ID_img_source,
$pos_x_dest, $pos_y_dest, $pos_x_source, $pos_y_source,
$largeur_source, $hauteur_source);
```

copie une partie d'une image source dans une image de destination.

```
$nombre = ImageCopyMerge($ID_img_dest,$ID_img_source,
$pos_x_dest, $pos_y_dest, $pos_x_source, $pos_y_source,
$largeur_source, $hauteur_source, $param);
```

copie et fusionne une partie d'une image avec un paramètre de fusion pouvant valoir de 0 à 100.

```
$nombre = ImageCopyMergeGray($ID_img_dest,$ID_img_source,
$pos_x_dest, $pos_y_dest, $pos_x_source, $pos_y_source,
$largeur_source, $hauteur_source, $param);
```

à l'instar de *ImageCopyMerge*, copie et fusionne une partie d'une image en niveaux de gris.

```
$nombre = ImageCopyResized($ID_img_dest,$ID_img_source,
$pos_x_dest, $pos_y_dest, $pos_x_source, $pos_y_source,
$largeur_dest, $hauteur_dest, $largeur_source, $hauteur_source);
```

copie et redimensionne une partie d'une image.

```
$nombre = ImageCopyResampled($ID_img_dest,$ID_img_source,
$pos_x_dest, $pos_y_dest, $pos_x_source, $pos_y_source,
$largeur_dest, $hauteur_dest, $largeur_source, $hauteur_source);
```

copie, redimensionne et rééchantillonne une image.

```
$ID_img = ImageCreate($hauteur, $largeur);
```

crée une nouvelle image à certaines dimensions.

```
$ID_img = imagecreatefromgif($fichier_gif);
```

crée une nouvelle image à partir d'un fichier ou d'une URL vers une image GIF.

```
$ID_img = ImageCreateTrueColor($hauteur, $largeur);
```

crée une nouvelle image en couleurs vraies.

```
ImageTrueColorTopalette($ID_img, $bool_granularité, $couleur);
```

convertit une image en couleurs vraies en image à palette.

```
$ID_img = ImageCreateFromJPEG($fichier_JPEG);
```

crée une nouvelle image JPEG à partir d'un fichier ou d'une URL.

$ID_img = **ImageCreateFromPNG**($fichier_PNG);

crée une nouvelle image PNG à partir d'un fichier ou d'une URL.

$ID_img = **ImageCreateFromWBMP**($fichier_WBMP);

crée une image depuis un fichier WBMP.

$ID_img = **ImageCreateFromString**($chaine);

crée une image à partir d'une chaîne.

$ID_img = **ImageCreateFromXBM**($fichier_XBM);

crée une image à partir d'un fichier XBM.

$ID_img = **ImageCreateFromXPM**($fichier_XPM);

crée une image à partir d'un fichier XPM.

$nombre = **ImageDashedLine**($ID_img, $x, $y, $X, $Y, $couleur);

trace une ligne pointillée entre les points x-y et X-Y.

$nombre = **ImageDestroy**($id_image);

détruit une image.

$nombre = **ImageFill**($id_image, $x, $y, $couleur);

remplit une image avec une certaine couleur et à partir des coordonnées spécifiées.

$nombre = **ImageFilledPolygon**($id_image, $tab_points, $nb_points, $couleur);

remplit d'une certaine couleur, un polygone dont les coordonnées des points sont spécifiées dans un tableau indicé (x1, y1, ..., xN, yN), le nombre de points devant être spécifié.

$nombre = **ImageFilledRectangle**($id_image, $x, $y, $X, $Y, $couleur);

remplit un rectangle d'une certaine couleur.

$nombre = **ImageFillToBorder**($id_image, $x, $y,
$couleur_bordure, $couleur_remplissage);

remplit une zone avec une certaine couleur délimité par la couleur de bordure.

$hauteur = **ImageFontHeight**($id_police);

retourne la hauteur de la police.

$largeur = **ImageFontWidth**($id_police);

retourne la largeur de la police.

$nombre = **ImageGif**($id_image [, $fichier]);

crée une image GIF vers un navigateur ou un fichier à partir d'une image.

$nombre = **ImagePNG**($id_image [, $fichier]);

crée une image PNG vers un navigateur ou un fichier à partir d'une image.

$nombre = **ImageJPEG**($id_image [, $fichier]);

crée une image JPEG vers un navigateur ou un fichier à partir d'une image.

$nombre = **ImageWBMP**($id_image [, $fichier [, $couleur_fond]]);

crée une image WBMP vers un navigateur ou un fichier à partir d'une image.

$nombre = **ImageInterlace**($id_image [, 0 | 1]);

active ou désactive le bit d'entrelacement.

ImageLine($id_image, $x, $y, $X, $Y, $couleur);

trace une ligne du point xy au point XY d'une certaine couleur, dans l'image.

$id_police = **ImageLoadFont**($fichier);

charge une nouvelle police à partir d'un fichier et retourne un identifiant.

$nombre = **ImagePaletteCopy**($id_img_destination, $id_img_source);

copie la palette d'une image source vers une autre de destination.

$nombre = **ImagePolygon**($id_image, $tab_points, $nb_points, $couleur);

dessine un polygone dans une image à partir de points spécifiés dans un tableau.

$tableau = **ImagePSBBox**($texte, $id_police, $taille, $espace, $largeur, $angle);

retourne un rectangle entourant un texte et dessiné avec une police PostScript Type 1.

$nombre = **ImagePSEncodeFont**($id_police, $fichier_encodage);

modifie le codage vectoriel d'un caractère dans une police.

ImagePSFreeFont($id_police);

libère les ressources occupées par une police PostScript Type 1.

$id_image = **ImagePSLoadFont**($fichier);

charge une police PostScript Type 1 à partir d'un fichier.

true | false = **ImagePsExtendFont**($id_police, $nb_reel);

étend ou condense une police de caractères selon une valeur à virgule flottante, dont une valeur inférieure à 1 provoquera une condensation.

$nombre = **ImagePsSlantFont**($id_police, $nb_reel);

incline une police de caractères selon une valeur à virgule flottante.

ImagePSText($id_image, $texte,
$id_police, $taille, $couleur_police, $couleur_fond, x, y
[, $espace [, $nb_espaces [, $angle [, $anti-aliasing]]]]);

dessine un texte sur une image avec une police PostScript Type 1 des couleurs, un positionnement, une taille et un nombre d'espacement, un angle d'inclinaison et une valeur d'anti-aliasage valant soit 4, soit 16. Le tableau indicé retourné contient dans l'ordre : l'abscisse inférieure gauche, l'ordonnée inférieure gauche, l'abscisse supérieure droite, l'ordonnée supérieure droite.

$nombre = **ImageRectangle**($id_image, $x, $y, $X, $Y, $couleur);

dessine un rectangle selon les coordonnnées et la couleur spécifiées.

$nombre = **ImageSetPixel**($id_image, $x, $y, $couleur);

dessine un pixel à la coordonnée spécifiée.

$nombre = **imagesetbrush**($id_image, $id_brosse);

modifie la brosse pour le dessin des lignes.

$nombre = **ImageSetTile**($id_image, $id_carrelage);

modifie le carrelage pour l'image.

$nombre = **ImageSetThickness**($id_image, $epaisseur);

modifie l'épaisseur d'un trait.

$nombre = **ImageString**($id_image, $num_police, $x, $y, $chaine, $couleur);

dessine une chaîne horizontale à la position xy, dans une des polices par défaut désignée par sa place (1, 2,...) et dans une certaine couleur.

$nombre = **ImageStringUp**($id_image, $num_police, $x, $y, $chaine, $couleur);

dessine une chaîne verticale à la position xy, dans une des polices par défaut désignée par sa place (1, 2,...) et dans une certaine couleur.

$largeur = **ImageSX**($id_image);

retourne la largeur d'une image.

$hauteur = **ImageSY**($id_image);

retourne la hauteur de l'image.

$tableau = **ImageTTFBBox**($taille, $angle, $fichier_police, $texte);

retourne les coordonnées d'un rectangle entourant un texte dessiné avec une police TrueType.

$nombre = **ImageTTFText**($id_image, $taille, $angle, $x, $y,
$couleur, $fichier_police, $texte);

dessine un texte avec une police TrueType.

ImageTypes();

Retourne les types d'images supportés par la version courante de PHP.

$nombre = **JPEG2WBMP**($nom_img_jpeg, $nom_img_wbmp,
$hauteur, $largeur);

convertit une image JPEG en image WBMP.

PNG2WBMP($nom_img_png, $nom_img_wbmp, $hauteur, $largeur);

convertit une image PNG en image WBMP.

$tableau = **read_exif_data**($fichier_jpeg);

retourne les entêtes EXIF d'une image JPEG dans un tableau.

25.6 / Les fonctions de mathématiques

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des expressions mathématiques.

Fonction
Description
$valeur = **abs**($nombre);
calcule la valeur absolue d'un nombre.
$valeur = **acos**($nombre);
calcule l'arc cosinus.
$valeur = **acosh**($nombre);
Arc cosinus hyperbolique.
$valeur = **asin**($nombre);
calcule l'arc sinus.
$valeur = **asinh**($nombre);
calcule l'arc sinus hyperbolique.
$valeur = **atan**($nombre);
calcule l'arc tangente.
$valeur = **atan2**($nombre_1, $nombre_2);
calcule l'arc tangente de deux nombres.
$valeur = **atanh**($nombre);
calcule l'arc tangente hyperbolique.
$chaine = **base_convert**($nombre, $base_depart, $base_fin);
retourne une chaîne de caractères contenant un nombre représenté par la base de départ, converti dans la base de fin. les valeurs de base doivent être comprises entre 2 et 36 inclus.
$valeur = **bindec**($nombre);
convertit le nombre binaire en décimal.
$valeur = **ceil**($nombre);
arrondit au nombre supérieur.
$valeur = **cos**($nombre);
calcule le cosinus.
$valeur = **cosh**($nombre);
calcule le cosinus hyperbolique.
$valeur = **decbin**($nombre);
convertit un nombre décimal en binaire.
$valeur = **dechex**($nombre);
convertit un nombre décimal en hexadécimal.
$valeur = **decoct**($nombre);
convertit un nombre décimal en octal.
$valeur = **deg2rad**($nombre);
convertit un angle en degrés, en radians.

$valeur = **exp**($nombre);
calcule l'exponentielle.
$valeur = **floor**($nombre);
calcule l'arrondi à l'entier inférieur.
$valeur = **getrandmax**();
retourne la plus grande valeur aléatoire possible produite par la fonction *mt_rand*.
$valeur = **hexdec**($nombre);
convertit un nombre hexadécimal en décimal.
$valeur = **lcg_value**($nombre);
retourne un nombre pseudo-aléatoire compris entre 0 et 1.
$valeur = **log**($nombre);
calcule le logarithme naturel ou népérien.
$valeur = **log10**($nombre);
calcule le logarithme en base 10.
$valeur = **max**($nombre, ..., $nombreN);
retourne la plus grande valeur.
$valeur = **min**($nombre, ..., $nombreN);
retourne la plus petite valeur.
$valeur = **mt_rand**($nombre_min, $nombre_max);
retourne une valeur aléatoire comprise entre l'intervalle des nombres spécifiés.
$valeur = **mt_srand**($nombre);
initialise le générateur de valeur aléatoire.
$valeur = **mt_getrandmax**();
retourne la plus grand valeur aléatoire possible.
$valeur = **number_format**($nombre, $nb_decimal, $separateur_decimal, séparateur_millier);
formate un nombre avec un certain nombre de chiffres après le séparateur décimal et éventuellement avec un séparateur entre les groupes de milliers.
$valeur = **octdec**($nombre);
convertit un nombre octal en décimal.
$valeur = **pi**();
retourne la valeur de PI.
$valeur = **pow**($nombre, $exposant);
calcule la puissance d'un nombre avec un certain exposant.
$valeur = **rad2deg**($nombre);
convertit un angle en radians, en degrés.
$valeur = **rand**($nombre_min, $nombre_max);
retourne une valeur aléatoire entre un intervalle de nombres.
$valeur = **round**($nombre);
calcule l'arrondi.
$valeur = **sin**($nombre);
calcule le sinus.

$valeur = **sinh**($nombre);
calcule le sinsus hyperbolique.
$valeur = **sqrt**($nombre);
calcule la racine carrée d'un nombre.
$valeur = **srand**($nombre);
initialise le générateur de nombres aléatoires.
$valeur = **tan**($nombre);
calcule la tangente.
$valeur = **tanh**($nombre);
calcule la tangente hyperbolique.

25.7 / Diverses fonctions

Le langage PHP propose de nombreuses fonctions permettant de travailler sur les constantes, d'évaluer une chaîne de caractères ou encore de diverses autres choses.

Fonction
Description
$nombre = **define**(NOM_CONSTANTE, $valeur [, $sensibilite_casse]);
crée une constante avec un nom, une valeur et éventuellement une sensibilité à la casse exprimée sous forme booléenne (*true* ou *false*).
$valeur = **constant**(NOM_CONSTANTE);
retourne la valeur d'une constante.
true | false = **defined**(NOM_CONSTANTE);
vérifie l'existence d'une constante.
die($texte);
affiche un texte et termine le script.
eval($chaine);
évalue une chaîne de caractères comme un script PHP.
exit([$message]);
termine le script en cours en utilisant éventuellement le message de fin d'exécution.
$objet = **get_browser**([$chaine_navigateur]);
indique les capacités du navigateur client.
true | false = **highlight_file**($fichier);
affiche le fichier spécifiée avec une colorisation syntaxique.
true | false = **highlight_string**($chaine);
affiche une chaîne de caractères avec un colorisation syntaxique.
$tableau = **iptcparse**();
recherche un balisage IPTC dans un bloc binaire et retourne un tableau associatif avec les noms de balise comme clés et leur valeur comme contenu.
leak($nb_octets);
crée un fuite de mémoire d'un nombre d'octets précisé.
$chaine = **pack**($format, $arg, ..., $argN);
compacte des arguments selon un format spécifié dans une chaîne binaire.

Format	Description
a	représente une chaîne de caractères avec NULL.
a	représente une chaîne de caractères avec espace (SPACE).
h	représente une chaîne hexadécimale avec un bit de poids faible en premier.
H	représente une chaîne hexadécimale avec un bit de poids fort en premier.
c	représente un caractère signé.
C	représente un caractère non signé.
s	représente un entier court signé toujours sur 16 bits et un ordre de bits dépendant du serveur.
S	représente un entier court non signé toujours sur 16 bits et un ordre de bits dépendant du serveur.
n	représente un entier court signé toujours sur 16 bits et un ordre de bits *big endian*.
v	représente un entier court non signé toujours sur 16 bits et un ordre de bits *little endian*.
i	représente un entier signé de taille et d'ordre de bits dépendants du serveur.

i	représente un entier non signé de taille et d'ordre de bits dépendants du serveur.
I	représente un entier long signé toujours sur 32 bits et d'ordre de bits dépendant du erveur.
L	représente un entier long non signé toujours sur 32 bits et d'ordre de bits dépendant du serveur.
N	représente un entier long non signé toujours sur 16 bits et d'ordre de bits *big endian*.
V	représente un entier long non signé toujours sur 16 bits et d'ordre de bits *little endian*.
f	représente un nombre à virgule flottante de taille et de représentation dépendantes du serveur.
d	représente un nombre à virgule flottante double de taille et de représentation dépendantes du serveur.
x	représente un bit NULL.
X	recule d'un octet.
@	remplit avec NULL jusqu'à une position absolue.

show_source($fichier);

affiche le fichier spécifiée avec une colorisation syntaxique.

sleep($nb_secondes);

retarde l'exécution d'un script d'un certain nombre de secondes.

$ID = **uniqid**($prefixe, true | false);

génère un identifiant unique sur 13 caractères ou 32 si le second argument est *true*.

$tab_valeur = **unpack**($format, $donnee);

décompacte des données depuis une chaîne binaire dans un tableau.

usleep($nb_microsecondes);

retarde l'exécution d'un script d'un certain nombre de micro-secondes.

26 / Les bases de données SQL

Le principal intérêt d'un langage Web dynamique est sa capacité à la gestion de bases de données SQL.

A cet effet, **le langage PHP propose de nombreux outils permettant de travailler avec la plupart des SGBDR** (Système de Gestion de Bases de données Relationnelles) telles que Oracle, Sybase, Microsoft SQL Server, PostgreSQL ou encore MySQL, son système de gestion de prédilection.

Avant d'aller plus loin dans ce cours, il est nécessaire de **connaître non seulement la struture d'une base de données, mais également les notions élémentaires du langage SQL**.

26.1 / La connexion à un SGBDR

La connexion à un système de gestion de base de données s'effectue par l'entremise des fonctions spécialisées.

```
mysql_connect("nom_serveur","nom_utilisateur","mot_passe");

mssql_connect("nom_serveur","nom_utilisateur","mot_passe");

ocilogon("nom_utilisateur","mot_passe", "nom_base");

pg_connect("dbname=nom_base host=nom_serveur port=num_port "
        . "user=nom_utilisateur password=mot_passe");

sybase_connect("nom_serveur","nom_utilisateur","mot_passe");
```

Il existe deux façons de se connecter à une base de données.

- Les connexions non-persistantes (*base_**connect***).
- Les connexions persistantes (*base_**pconnect***).

Tout d'abord, **les deux type de connexions sont parfaitement identiques au niveau des fonctionnalités** qu'elles apportent. Néanmoins, **les connexions persistantes ne se referment pas** automatiquement à la fin du script. Lorsqu'une telle connexion est demandée, PHP s'assure qu'il n'existe pas un processus semblable, déjà ouvert avec les noms de serveur et d'utilisateur ainsi que le mot de passe. Si tel est le cas, ce processus est réutilisé sinon une nouveau est ouvert.

Ainsi, **le principal avantage des connexions persistantes et leur réutilisabilité** et partant l'impossiblité d'ouvertures multiples de connexions à un SGBDR. Toutefois, il est impératif de **s'assurer de la fermeture correcte des processus au terme de leur utilisation** puisque si le serveur n'admet qu'un certain nombre de clients alors une connexion inutilisée constituera une perte de capacités.

La déconnexion des bases de données s'effectue par l'intermédiaire des fonctions de fermeture.

```
mysql_close($id_connexion);

mssql_close($id_connexion);

ocilogoff($id_connexion);

pg_close($id_connexion);

sybase_close($id_connexion);
```

Plusieurs fonctions PHP permettent de retourner des informations à propos de la connexion en cours.

```
$chaine_numero_version = mysql_get_client_info();
$type_connexion = mysql_get_host_info($id_connexion);
$protocole_connexion = mysql_get_proto_info($id_connexion);
$chaine_version_serveur = mysql_get_server_info($id_connexion);

$nom_hote = pg_host($id_connexion);
$option_connexion = pg_options($id_connexion);
$num_port = pg_port($id_connexion);
pg_set_client_encoding($id_connexion, $encodage);
$encodage = pg_client_encoding($id_connexion);
```

Exemple [voir]

```php
<?php
// Fichier : traitement.php
 function connexion($sgbdr, $hote, $port,
                    $utilisateur, $mot_passe, $param_sup)
 {
 if ($type == "")
 {
    echo ("Aucun SGBDR n'a été spécifié !");
    return false;
 }
 else
 {
    switch ($type)
    {
     case "MySQL" :
     {
      if ($port != "") $hote .= ":". $port;
      $id_connexion = mysql_connect($hote, $utilisateur, $mot_passe);
     }
     case "mSQL" :
     {
      if ($port != "") $hote .= ":". $port;
      $id_connexion = msql_connect($hote, $utilisateur, $mot_passe);
     }
     case "SQLSever" :
     {
      $id_connexion = mssql_connect($hote, $utilisateur, $mot_passe);
     }
     case "ODBC" :
     {
     if ($param_sup == "")
      $id_connexion =
          odbc_connect($hote, $utilisateur, $mot_passe);
     else
      $id_connexion =
          odbc_connect($hote, $utilisateur, $mot_passe, $param_sup);
     }
     case "Oracle" :
     {
      $id_connexion = ocilogon($utilisateur, $mot_passe);
     }
     case "PostgreSQL" :
     {
      $chaine_connexion = "host=". $hote . " ";
      if ($port != "") $chaine_connexion .= "port=". $port . " ";
      $chaine_connexion .= "user=". $utilisateur ." password=". $mot_passe";
      if ($param_sup != "") $chaine_connexion .= " ".$param_sup;
      $id_connexion = pg_connect($chaine_connexion);
     }
     case "Sybase" :
     {
      if ($param_sup == "")
      {
       $id_connexion =
          sybase_connect($hote, $utilisateur, $mot_passe);
      }
      else
      {
       $id_connexion =
          sybase_connect($hote, $utilisateur, $mot_passe, $param_sup);
      }
     }
     default :
     {
      echo ("Le SGBDR " . $sgbdr . " n'est pas géré.");
      return false;
     }
    }
    return true;
  }
 }

 connexion($nom_sgbdr, $nom_serveur, $num_port,
```

```
                $nom_utilisateur, $mot_de_passe, $autre_param);
?>
<!-- Fichier : formulaire.php -->
<html>
 <body>
  <form action="traitement.php" method="POST">
   <table>
    <tr>
     <td>Nom</td>
     <td>: <input type="text" size="20" name="nom_utilisateur"></td>
    </tr>
    <tr>
     <td>Mot de passe</td>
     <td>: <input type="password" size="20" name="mot_de_passe"></td>
    </tr>
    <tr>
     <td>Serveur hôte</td>
     <td>: <input type="text" size="20" name="nom_serveur"
                value="<?php echo $SERVER_NAME ?>"></td>
    </tr>
    <tr>
     <td>Numéro de port</td>
     <td>: <input type="text" size="20" name="num_port"
                value="<?php echo $SERVER_PORT ?>"></td>
    </tr>
    <tr>
     <td>Nom SGBDR</td>
     <td>: <input type="text" size="20" name="nom_sgbdr"></td>
    </tr>
    <tr>
     <td>Autre paramètre</td>
     <td>: <input type="text" size="20" name="autre_param"></td>
    </tr>
    <tr>
     <td> </td>
     <td><input type="submit" name="Soumission" value="Soumettre"></td>
    </tr>
   </table>
  </form>
 </body>
</html>
```

26.2 / L'accès aux bases de données

Suite à la connexion à un SGBDR, il faut soit sélectionner la base de données si elle existe déjà, soit la créer si ce n'est pas le cas.

La sélection de bases de données s'effectuent par le truchement de fonctions adaptées.

```
msql_select_db($nom_base_donnee, $id_connexion);

mysql_select_db($nom_base_donnee, $id_connexion);

sybase_select_db($nom_base_donnee, $id_connexion);

pg_connect( "dbname=nom_base "
          . "host=nom_serveur "
          . "port=num_port "
          . "user=nom_utilisateur "
          . "password=mot_passe");
```

La création des bases de données peut être réalisée par des fonctions PHP dévolues à cette tâche.

```
msql_create_db($nom_base_donnee, $id_connexion);

mssql_create_db($nom_base_donnee, $id_connexion);

mysql_create_db ($nom_base_donnee, $id_connexion);
```

Si la création d'une base de données n'est pas possible à l'aide de fonctions, il est possible de **créer une base à l'aide d'une requête SQL**.

```
$requete = "CREATE DATABASE nom_base_donnee";
$id_requete = ociparse($requete, $id_connexion);
ociexecute($id_requete);
```

La suppression des bases de données est permises, de la même façon qu'il est possible de les créer.

```
msql_drop_db($nom_base_donnee, $id_connexion);

mssql_drop_db($nom_base_donnee, $id_connexion);

mysql_drop_db ($nom_base_donnee, $id_connexion);
```

Certaines fonctions permettent de retourner **la liste des bases de données et de leurs tables**.

```
$id_connexion =
      mysql_connect('localhost', 'administrateur', 'md2N50Oml');
$liste_bases = mysql_list_dbs();
$nb_bases = mysql_num_rows($liste_bases);
echo "<h3>Liste des bases de données</h3>";
for($i = 0; $i < $nb_bases; $i++)
{
 $nom_base_donnee = mysql_db_name($liste_bases, $i) . "<br>";

 $liste_tables =
     mysql_list_tables($nom_base_donnee, $id_connexion);
 $nb_tables = mysql_num_rows($liste_tables);

 echo "<h3>" . $nom_base_donnee . "</h3>";
 echo "<h4>Liste des tables :</h4>";
 for($j = 0; $j < $nb_tables; $j++)
 {
  echo mysql_tablename($liste_tables, $j) . "<br>";
 }
}
```

Exemple [voir]

```
<!-- Formulaire -->
<html>
 <body>
  <form method="POST" action="traitement.php">
   <input type="text" name="nom" size="20" value="nom"><br>
   <input type="text" name="prenom" size="20" value="prenom"><br>
   <input type="text" name="email" size="20" value="email"><br>
   <input type="submit" name="soumission" value="Soumettre">
  </form>
 </body>
</html>

<?php
// fichier : traitement.php
 $id_connex = mysql_connect("localhost","root","emma")
      or die("La connexion a échoué !");

 $id_liste_bases = mysql_list_dbs($id_connex);
 $trouve = false;
 for($i = 0; $i < mysql_num_rows($id_liste_bases); $i++)
 {
  if(mysql_db_name($id_liste_bases, $i) == 'utilisateur')
  {
   $trouve = true;
  }
 }
 if(!$trouve)
 {
  mysql_create_db("utilisateur")
       or die("La création de la base a échoué !");
 }

 $id_select = mysql_select_db("utilisateur")
      or die("La sélection de la base a échoué !");

 $id_liste_tables = mysql_list_tables('utilisateur', $id_connex);
 $trouve = false;
 for($i = 0; $i < mysql_num_rows($id_liste_tables); $i++)
 {
  if (mysql_tablename($id_liste_tables, $i) == 'tbl_utilisateur')
  {
   $trouve = true;
  }
 }
 if(!$trouve)
 {
  mysql_query("create table tbl_utilisateur "
                . "(date CHAR(30) NOT NULL, email CHAR(50) UNIQUE, "
                . "nom CHAR(50) NOT NULL)", $id_connex)
      or die("La création de la table a échoué !");
 }

 if($id_select)
 {
  mysql_query("insert into tbl_utilisateur (date, email, nom) "
                . "values ('" . date("d/m/Y H:i:s") . "', '" . $email
                .  "', '" . $prenom . " " . $nom . "')", $id_connex)
      or die("Impossible d'insérer les informations !");
 }
 else
 {
  echo "<h3>Impossible de sélectionner la table !</h3>";
 }

 $id_requete = mysql_query("select * from tbl_utilisateur", $id_connex);
 if($id_requete)
 {
  echo '<table border="0">'
       . '<tr bgcolor="#000000" style="color:#FFFFFF">'
       . '<th>Date</th><th>eMail</th><th>Nom</th></tr>';
  while($ligne = mysql_fetch_array($id_requete))
  {
   echo '<tr bgcolor ="#FFFF00">'
```

```
        . '<td>' . $ligne['date'] . '</td>'
        . '<td>' . $ligne['email'] . '</td>'
        . '<td>' . $ligne['nom'] . '</td>';
      echo '<tr>';
    }
    echo '</table>';
    mysql_free_result($id_requete);
  }
  else
  {
    echo "<h3>Impossible d'exécuter la requête de sélection !</h3>";
  }
  mysql_close();
?>
```

26.3 / Les requêtes SQL

Les requêtes SQL permettent d'accomplir une action sur une base de données comme la sélection d'informations, la création de tables, l'ajout, la suppression ou la modification des enregistrements.

```
$requete = "SELECT * FROM table WHERE champ = \"valeur\"";

$id_resultat = mssql_query($requete, $id_connexion);

$id_resultat = msql_query($requete, $id_connexion);

$id_resultat = mysql_query($requete, $id_connexion);

// analyse de la requête
$id_resultat = ociparse($id_connexion, $requete);
// exécution de la requête
ociexecute ($id_resultat);

$id_resultat = pg_exec($id_connexion, $requete);

$id_resultat = sybase_query($requete, $id_connexion);
```

Toutes **ces fonctions prennent comme argument une requête SQL** qui sera envoyée à la base de données définie par un identificateur de connexion.

```
$requete = "CREATE TABLE tbl_nom ("
            . "nom_champ INTEGER PRIMARY KEY,"
            . "nom_champ2 CHAR(50) UNIQUE,"
            . "nom_champ3 DATETIME)";

$requete = "INSERT INTO tbl_nom "
            . "(nom_champ, nom_champ2, nom_champ3) "
            . "VALUES('valeur','valeur2','valeur3')";

$requete = "SELECT * FROM tbl_nom "
            . "WHERE nom_champ2 = 'valeur'";

$requete = "DELETE FROM tbl_nom "
            . "WHERE nom_champ3 < SYSDATE - 7";
```

En cas de réussite, **les fonctions retournent un identificateur**, sinon la valeur est *false*.

L'identificateur représente le résultat produit par la requête dans la base de données en cours. La variable *$id_resultat* pourra par la suite être utilisée par d'autres fonctions afin d'exploiter les données disponibles.

```
$tab_ligne = mssql_fetch_row($id_resultat);

$tab_ligne = msql_fetch_row($id_resultat);

$tab_asso_ligne = mysql_fetch_assoc($id_resultat);

$nb_lignes = ocifetchstatement($id_resultat, $tab_lignes);

$tab_ligne = pg_fetch_row($id_resultat, $num_ligne);

$tab_ligne = sybase_fetch_array ($id_resultat);
```

Les requêtes doivent répondre à la syntaxe SQL (Structured Query Language) en général et éventuellement aux singularités des différents éditeurs de SGBDR.

Voir le cours SQL

Exemple [voir]

```
<!-- Formulaire -->
<html>
 <body>
  <form method="POST" action="traitement.php">
   <input type="text" name="nom" size="20" value="nom"><br>
   <input type="text" name="prenom" size="20" value="prenom"><br>
   <input type="text" name="email" size="20" value="email"><br>
   <input type="submit" name="soumission" value="Soumettre">
  </form>
 </body>
</html>

<?php
// fichier : traitement.php
 $id_connex = mysql_connect("localhost","root","emma")
 or die("La connexion a échoué !");

 $id_liste_bases = mysql_list_dbs($id_connex);
 $trouve = false;
 while($ligne = mysql_fetch_assoc($id_liste_bases))
 {
  if ($ligne['Database'] == 'utilisateur')
  {
   $trouve = true;
  }
 }
 if(!$trouve)
 {
  mysql_create_db("utilisateur")
     or die("La création de la base a échoué !");
 }

 $id_select = mysql_select_db("utilisateur")
     or die("La sélection de la base a échoué !");

 $id_liste_tables = mysql_list_tables('utilisateur', $id_connex);
 $i = 0;
 $trouve = false;
 while($ligne = mysql_fetch_array($id_liste_tables))
 {
  if ($ligne[$i] == 'tbl_utilisateur')
  {
   $trouve = true;
  }
  $i++;
 }
 if(!$trouve)
 {
  mysql_query("create table tbl_utilisateur "
            . "(date CHAR(30) NOT NULL, email CHAR(50) UNIQUE, "
            . "nom CHAR(50) NOT NULL)", $id_connex)
       or die("La création de la table a échoué !");
 }

 if($id_select)
 {
  mysql_query("insert into tbl_utilisateur (date, email, nom) "
            . "values ('" . date("d/m/Y H:i:s") . "', '" . $email
            .  "', '" . $prenom . " " . $nom . "')", $id_connex)
      or die("Impossible d'insérer les informations !");
 }
 else
 {
  echo "<h3>Impossible de sélectionner la table !</h3>";
 }

 $id_requete = mysql_query("select * from tbl_utilisateur", $id_connex);
 if($id_requete)
 {
  echo '<table border="0">'
      . '<tr bgcolor="#000000" style="color:#FFFFFF">'
      . '<th>Date</th><th>eMail</th><th>Nom</th></tr>';
  while($ligne = mysql_fetch_array($id_requete))
```

```
    {
      echo '<tr bgcolor ="#FFFF00">'
        . '<td>' . $ligne['date'] . '</td>'
        . '<td>' . $ligne['email'] . '</td>'
        . '<td>' . $ligne['nom'] . '</td>';
      echo '<tr>';
    }
    echo '</table>';
    mysql_free_result($id_requete);
  }
  else
  {
    echo "<h3>Impossible d'exécuter la requête de sélection !</h3>";
  }
  mysql_close();
?>
```

26.4 / L'exploitation des données

De nombreuses fonctions de bases de données permettent d'extraire des informations telles que des enregistrements ou des champs précis d'une ou plusieurs tables.

Subséquemment à l'exécution d'une requête, **des fonctions spécialisées s'occupent de l'extraction de données ciblées**.

```
$requete = "SELECT nom, prenom, adresse, cp, ville"
          . "FROM tbl_client";

// Sous Microsoft SQL Server
$id_connexion = mssql_connect("localhost", "jjfredo", "02h4Ypl3");
mssql_select_db("bd_commerce");
$id_resultat = mssql_query($requete, $id_connexion);

// Sous Oracle
$id_connexion = ocilogon("jjfredo", "02h4Ypl3", "bd_commerce");
$id_resultat = ociparse($requete);
ociexecute($id_resultat);
```

Lorsque le résultat d'une requête devient disponible, le pointeur d'enregistrement se situe sur la première ligne de la table, soit à l'index '0'.

Index	Champ_1	Champ_2	Champ_3
0	Valeur_1	Valeur_2	Valeur_3
1	Valeur_1	Valeur_2	Valeur_3
...			
N	Valeur_1	Valeur_2	Valeur_3

A partir de là, il suffit de **déplacer le pointeur par une incrémentation de l'index ou de le placer directement à un index spécifié**. Chaque ligne pourra alors délivrer la totalité de son contenu.

```
// Sous Microsoft SQL Server
echo "<table>";
// décompte du nombre d'enregistrements
for($i = 0; $i < mssql_num_rows($id_resultat); $i++)
{
  echo "<tr><th>" . $i . "</th><td>";
  // lecture de l'enregistrement
  print_r(mssql_fetch_row($id_resultat));
  echo "</td></tr>";
  // déplacement du pointeur
  mssql_data_seek($id_resultat, $i);
}
echo "</table>";

// Sous Oracle
while(ocifetch($id_resultat))
{
  // lecture de l'enregistrement
  ocifetchinto($id_resultat, $ligne);
  print_r($ligne);
}
// ou
$nb_lignes = ocifetchstatement($id_resultat, $table_resultante);
for($i = 1; $i <= $nb_lignes; $i++)
{
  $ligne = array_slice($table_resultante, $i, 1);
  print_r($ligne);
}
```

Des fonctions de déplacement du pointeur s'occupent de **désigner soit un enregistrement,**

soit un champ pour une utilisation ultérieure comme l'extraction, la modification ou encore la suppression des données.

```
// Sous Microsoft SQL Server
echo "<table>";
// décompte du nombre de champs
for($i = 0; $i < mssql_num_fields($id_resultat); $i++)
{
  echo "<tr><th>" . $i . "</th><td>";
  // lecture des informations sur le champ
  print_r(mssql_fetch_field($id_resultat));
  echo "</td></tr>";
  // déplacement du pointeur
  mssql_field_seek($id_resultat, $i);
}
echo "</table>";

// Sous Oracle
while(ocifetch($id_resultat))
{
  for($i = 1; $i <= ocinumcols($id_resultat); $i++)
  {
    echo ociresult($id_resultat, $i);
  }
}
```

Le décompte total des enregistrements ou des champs d'une table, peut être obtenu par l'intermédiaire de certaines fonctions.

```
$nb_champs = msql_num_fields($id_resultat);
$nb_lignes = msql_num_rows($id_resultat);

$nb_champs = mysql_num_fields($id_resultat);
$nb_lignes = mysql_num_rows($id_resultat);

$nb_champs = ociNumCols($id_resultat);
$nb_lignes = ociRowCount($id_resultat);

$nb_champs = pg_NumFields($id_resultat);
$nb_lignes = pg_NumRows($id_resultat);

$nb_champs = sybase_num_fields($id_resultat);
$nb_lignes = sybase_num_rows($id_resultat);
```

A partir des enregistrements, **chacun des champs devient accessible aisément** par des fonctions appropriées.

En outre, certaines fonctions sont capables d'**extraire directement un champ déterminé** avec exactitude.

```
$valeur = mssql_result($id_resultat, $num_ligne, $num_col);

$valeur = msql_result($id_resultat, $num_ligne, $num_col);

$valeur = mysql_result($id_resultat, $num_ligne, $num_col);

$valeur = pg_result($id_resultat, $num_ligne, $num_col);

$valeur = sybase_result($id_resultat, $num_ligne, $num_col);
```

Afin d'éviter une surcharge de la mémoire, **des instructions PHP permettent de libérer les ressources** suite au terme de leur utilisation.

```
mssql_free_result($id_resultat);

msql_free_result($id_resultat);

mysql_free_result($id_resultat);

sybase_free_result($id_resultat);
```

Le langage PHP prend en charge la plupart des SGBDR courants, lui procurant ainsi **un panel**

impressionnant d'outils visant à exploiter pleinement n'importe quel type de bases de données.

Exemple [voir]

```
<!-- Formulaire -->
<html>
 <body>
  <form method="POST" action="traitement.php">
   <input type="text" name="nom" size="20" value="nom"><br>
   <input type="text" name="prenom" size="20" value="prenom"><br>
   <input type="text" name="email" size="20" value="email"><br>
   <input type="submit" name="soumission" value="Soumettre">
  </form>
 </body>
</html>

<?php
// fichier : traitement.php
 $id_connex = mysql_connect("localhost","root","emma")
      or die("La connexion a échoué !");

 $id_liste_bases = mysql_list_dbs($id_connex);

 $trouve = false;
 while($ligne = mysql_fetch_assoc($id_liste_bases))
 {
  if ($ligne['Database'] == 'utilisateur')
  {
   $trouve = true;
  }
 }
 if(!$trouve)
 {
  mysql_create_db("utilisateur")
     or die("La création de la base a échoué !");
 }

 $id_select = mysql_select_db ("utilisateur")
      or die("La sélection de la base a échoué !");
 $id_liste_tables = mysql_list_tables ('utilisateur', $id_connex);

 $trouve = false;
 for($i = 0; $i < mysql_num_rows($id_liste_tables); $i++)
 {
  if (mysql_tablename($id_liste_tables, $i) == 'tbl_utilisateur')
  {
   $trouve = true;
  }
 }
 if(!$trouve)
 {
  mysql_query("create table tbl_utilisateur "
             . "(date CHAR(30) NOT NULL, email CHAR(50) UNIQUE, "
             . "nom CHAR(50) NOT NULL)", $id_connex)
     or die("La création de la table a échoué !");
 }

 if($id_select)
 {
  mysql_query("insert into tbl_utilisateur (date, email, nom) "
             . "values ('" . date("d/m/Y H:i:s") . "', '" . $email
             .  "', '" . $prenom . " " . $nom . "')", $id_connex)
   or die("Impossible d'insérer les informations !");
 }
 else
 {
  echo "<h3>Impossible de sélectionner la table !</h3>";
 }

 $id_requete = mysql_query("select * from tbl_utilisateur", $id_connex);
 if($id_requete)
 {
  echo '<table border="0">'
       . '<tr bgcolor="#000000" style="color:#FFFFFF">';

  $id_champs = mysql_list_fields("utilisateur", "tbl_utilisateur", $id_connex);
  $nb_champs = mysql_num_fields($id_requete);
  for ($i = 0; $i < $nb_champs; $i++)
```

```
    {
      echo '<th>' . mysql_field_name($id_requete, $i) . '</th>'
    }
    echo '</tr>';

    for($i = 0; $i < mysql_num_rows($id_requete); $i++)
    {
      echo '<tr bgcolor ="#FFFF00">';
      for($j = 0; $j < mysql_num_fields($id_requete); $j++)
      {
        echo '<td>' . mysql_result($id_requete, $i, $j) . '</td>';
      }
      echo '<tr>';
    }
    echo '</table>';
    mysql_free_result($id_requete);
  }
  else
  {
    echo "<h3>Impossible d'exécuter la requête de sélection !</h3>";
  }
  mysql_close();
?>
```

26.5 / Les propriétés de colonnes

Les instructions de bases de données du langage PHP, permettent de récupérer diverses informations à propos des champs d'une table résultant d'une requête SQL.

En effet, **le nom d'un champ** peut être retourné par l'intermédiaire de certaines fonctions.

```
$nom = mssql_field_name($id_resultat, $num_champ);

$nom = msql_fieldname($id_resultat, $num_champ);

$nom = mysql_field_name($id_resultat, $num_champ);

$nom = ociColumnName($id_resultat, $num_champ);

$nom = pg_FieldName($id_resultat, $num_champ);
```

De même, **le type du champ** est obtenu en utilisant les fonctions spécifiques.

```
$type_donnee = mssql_field_type($id_resultat, $num_champ);

$type_donnee = msql_fieldtype($id_resultat, $num_champ);

$type_donnee = mysql_field_type($id_resultat, $num_champ);

$type_donnee = ociColumnType($id_resultat, $num_champ);

$type_donnee = pg_FieldType($id_resultat, $num_champ);
```

Enfin, d'autres fonctions renvoyent **la longueur d'un champ**.

```
$longueur = mssql_field_length($id_resultat, $num_champ);

$longueur = msql_fieldlen($id_resultat, $num_champ);

$longueur = mysql_field_len($id_resultat, $num_champ);

$longueur = ociColumnSize($id_resultat, $num_champ);

$longueur = pg_FieldSize($id_resultat, $num_champ);
```

Pour le SGBDR *Sybase*, les propriétés pour un champ spécifié, sont regroupées dans un objet possédant les propriétés : *name* (nom de la colonne), *column_source* (nom de la table parente), *max_length* (longueur maximum pour la colonne), numeric (colonne numérique : 1 sinon -1) et *type* (type de donnée de la colonne).

```
$obj_info = sybase_fetch_field($id_resultat, $num_champ);
```

D'autres propriétés peuvent être recueillies par des instructions relatives à certains gestionnaires de bases de données

```
// nom de la table parente de la colonne
$nom_table = msql_fieldtable($id_resultat, $num_champ);
$nom_table = msql_tablename($id_resultat, $num_champ);
$nom_table = mysql_field_table($id_resultat, $num_champ);
$nom_table = mysql_tablename($id_resultat, $num_champ);

// sémaphore du champ comme NOT NULL, PRIMARY KEY
$semaphore = msql_fieldflags($id_resultat, $num_champ);
$semaphore = mysql_field_flags($id_resultat, $num_champ);

// vérification de la valeur NULL
$bool_ok = ociColumnIsNULL($id_resultat, $num_champ);
$bool_ok = pg_FieldIsNull($id_resultat, $num_champ);

// numéro du champ identifié par son nom
$num_col = pg_FieldNum($id_resultat, $nom_champ);

// taille imprimable du champ
$taille = pg_FieldPrtLen($id_resultat, $num_champ);
```

Exemple [voir]

```
<?php
 $id_connexion = mysql_connect("localhost","root","emma");
 mysql_select_db("utilisateur");
 $requete = "SELECT * FROM tbl_utilisateur";
 $id_resultat = mysql_query($requete, $id_connexion)
      or die ("La requête est invalide : ".$requete."<br>");

 $nb_champs = mysql_num_fields($id_resultat);
 $ligne = mysql_fetch_row($id_resultat);
 $type = array();
 $propriete = array();
 for ($i = 0; $i < $nb_champs; $i++)
 {
  $propriete[$i]['nom'] = mysql_field_name($id_resultat, $i);
  $propriete[$i]['type'] = mysql_field_type($id_resultat, $i);
  $propriete[$i]['longueur'] = mysql_field_len($id_resultat, $i);
 }

 for($i = 0; $i < $nb_champs; $i++)
 {
  echo "<h3>Colonne n°" . $i . "</h3>";
  foreach($propriete[$i] as $cle => $valeur)
  {
   echo "<u>" . $cle . " :</u> <b>" . $valeur . "</b><br>";
  }
 }
?>
```

26.6 / La gestion des erreurs

Les erreurs générées par la plupart des SGBDR ne sont plus traitées comme des alertes. Désormais elles sont stockées et deviennent disponibles à partir d'une fonction spécifique.

En général, deux fonctions PHP retournent respectivement **le numéro et le message de l'erreur en cours**. Certains SGBDR ne gérent que le message d'erreur tels que PostgreSQL et Sybase.

```
// message d'erreur généré par Microsoft SQL Server
$message = mssql_get_last_message();

// numéro et message d'erreur due à la dernière action pour MySQL
$num_erreur = mysql_errno([$id_connexion]);
$message = mysql_error([$id_connexion]);

// message d'erreur généré par mSQL
$message = msql_error();

// message d'erreur pour une connexion ou une requête Oracle 8
$message = ociError([$id_connexion | $id_requete]);

// message d'erreur généré par PostgreSQL
$message = pg_ErrorMessage($id_connexion);

// message d'erreur généré par Sybase
$message = sybase_get_last_message();
```

Quelques gestionnaires de bases de données autorisent **le paramétrage du niveau de sévérité des erreurs** se produisant sur le serveur ou chez le client.

```
// niveau de sévérité des erreurs ou des messages // d'erreur sur Microsoft SQL Server
mssql_min_error_severity($num_severite);
mssql_min_message_severity($num_severite);

// niveau de sévérité des erreurs du client
sybase_min_client_severity($num_severite);
sybase_min_error_severity($num_severite);
sybase_min_message_severity($num_severite);
sybase_min_server_severity($num_severite);
```

Enfin, les fonctions Oracle intègrent une instruction d'**activation ou de désactivation de l'affichage des données de debuggage**.

```
ociinternaldebug(0 | 1);
```

Exemple [voir]

```
<?php
 $id_connexion = mysql_connect("localhost", "root", "mot");
 if (!$id_connexion)
 {
  $message = "<h3>Une erreur est survenue :</h3>"
             . "<b><u>Erreur numéro " . mysql_errno()
             . ":</u> " . mysql_error() . "</b>";
  echo $message;
 }
 else
 {
  $reussite = mysql_select_db("utilsateur");
  if (!$reussite)
  {
   $message = "<h3>Une erreur est survenue :</h3>"
              . "<b><u>Erreur numéro " . mysql_errno()
              . ":</u> " . mysql_error() . "</b>";
    echo $message;
  }
  else
  {
   $id_requete = mysql_query("SELECT datte, email, nom "
                 . "FROM tbl_utilsateur");
   if (!$id_requete)
   {
    $message = "<h3>Une erreur est survenue :</h3>"
              . "<b><u>Erreur numéro " . mysql_errno()
              . ":</u> " . mysql_error() . "</b>";
    echo $message;
   }
  }
 }
?>
```

26.7 / Les fonctions DBase

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données DBase.

Le système de base de données Dbase n'est recommandée que pour l'importation et l'exportation de données en raison des limitations générées par sa structure. Les bases de données DBase sont des fichiers séquentiels dont les enregistrements sont de longueur fixe.

L'utilisation de ces fonctions nécessite l'installation de la librairie et de sa déclaration dans le fichier de configuration *php.ini*.

extension=php_dbase.dll

Fonction
Description
$ID \| False = **dbase_create**($nom, $tab_champs);
crée une base de données dBase à partir d'un tableau de champs et d'un nom.

Champ	Type	Description
L	Boolean	représente une valeur booléenne.
M	Memo	représente un champ de texte. Les Mémos ne sont pas supportés par PHP.
D	Date	représente une date au format 'AAAAMMJJ'.
N	Number	représente un nombre possèdant une longueur et un précision.
C	String	représente une chaîne de caractères.

$ID \| false = **dbase_open**($nom, $mode);
ouvre une base dBase du nom indiqué et en lecture seule '0', en écriture seule '1', et en lecture et écriture '2'.
true | false = **dbase_close**($ID);
ferme une base dBase à partir de son identifiant.
true | false = **dbase_pack**($ID);
compacte une base dBase à partir de son identifiant.
true | false = **dbase_add_record**($ID, $tab_enregistrement);
ajoute les données d'un enregistrement disposées dans un tableau, dans une base dBase.
true | false = **dbase_replace_record**($ID, $tab_enregistrement, $num_enregistrement);
remplace l'enregistrement à la ligne spécifiée par un autre dans une base dBase.
true | false = **dbase_delete_record**($ID, $num_enregistrement);
supprime l'enregistrement à la ligne spécifiée dans une base dBase.
$tab_valeur = **dbase_get_record**($ID, $num_enregistrement);
retourne un enregistrement d'une base dBase dans un tableau.
$tab_valeur = **dbase_get_record_with_names**($ID, $num_enregistrement);
retourne un enregistrement d'une base dans un tableau associatif.
$nombre = **dbase_numfields**($ID);
retourne le nombre de champs dans une base dBase.
$nombre = **dbase_numrecords**($ID);
retourne le nombre d'enregistrements dans une base dBase.

Exemple

```
<?php
 $nom_base = "fichier.dbf";
 $id_ouverture = dbase_open($nom_base, 0)
 or die("Impossible d'ouvrir la base de données !");

 $nb_enregistrements = dbase_numrecords($id_ouverture);
 $nb_champs = dbase_numfields($id_ouverture);

 echo "<table><tr>";

 for($i = 1; $i <= $nb_enregistrements; $i++)
 {
  $enregistrement = dbase_get_record_with_names($id_ouverture, $i);

  $entete = array_keys($enregistrement);
  if ($i == 1)
  {
   foreach($entete as $titre)
   {
    echo "<th>" . $titre . "<\th>";
   }
  }
  echo "</tr><tr>";
  foreach($enregistrement as $cle=>$valeur)
  {
   echo "<td>" . $valeur . "<\td>";
  }
 }

 echo "</tr></table>";

 dbase_close($id_ouverture);
?>
```

26.8 / Les fonctions Microsoft SQL Server

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données Microsoft SQL Server.

Ces fonctions ne sont disponibles que sous Windows 32 bits et avec une installation des librairies adéquates sur le serveur. Le fichier *ntwdblib.dll* disponible sur le cédérom d'installation, doit être copié dans le répertoire *\WINNT\SYSTEM32* et le fichier *php.ini* doit également comporter la ligne *extension=php_mssql.dll*.

Pour en savoir plus, la consultation des sites php.net, phpbuilder.com et de freetds.org s'impose.

Fonction
Description
true | false = **mssql_close**($id_connexion);
ferme une connexion SQL Server.
$id_connexion \| false = **mssql_connect**([$nom_serveur [, $utilisateur [, $mot_passe]]]);
ouvre une connexion à un serveur SQL server.
true | false = **mssql_data_seek**($id_connexion, $num_enregistrement);
déplace le pointeur vers un enregistrement désigné par son numéro.
$tab_valeurs \| false = **mssql_fetch_array**($id_resultat);
retourne les valeurs d'un enregistrement dans un tableau.
$obj_informations = **mssql_fetch_field**($id_resultat [, $position_champs]);
retourne les informations sur le champs dans un objet dont les propriétés sont :

Index	Description
name	représente le nom de la colonne.
column_source	représente le nom de la table d'où la colonne est originaire.
max_length	représente la taille maximale de la colonne.
numeric	retourne -1 si la colonne est numérique.

$obj_valeurs \| false = **mssql_fetch_object**($id_resultat);
retourne un objet contenant les valeurs d'un enregistrement.
$tab_valeurs \| false = **mssql_fetch_row**($id_resultat);
retourne les valeurs d'un enregistrement dans un tableau.
$longueur = **mssql_field_length**($id_resultat [, $position_champs]);
retourne la longueur d'un champs.
$nom = **mssql_field_name**($id_resultat [, $position_champs]);
retourne le nom d'un champs.
$position = **mssql_field_seek**($id_resultat, $position_champs);
positionne le pointeur sur un champs et retourne sa position.
$type = **mssql_field_type**($id_resultat [, $position_champs]);
retourne le nom d'un champs.
true | false = **mssql_free_result**($id_resultat);
libère les ressources utilisées par un résultat.
$message = **mssql_get_last_message**();
retourne un message d'erreur du serveur.
mssql_min_error_severity($niveau_severite);
détermine le niveau de sévérité des erreurs.
mssql_min_message_severity($niveau_severite);
détermine le niveau de sévérité des messages d'erreurs.
true | false = **fbsql_next_result**($id_resultat);
déplace le pointeur vers le résultat suivant.
$nb_champs = **mssql_num_fields**($id_resultat);
retourne le nombre de champs dans un résultat.

$nb_enregistrements = **mssql_num_rows**($id_resultat);
retourne le nombre d'enregistrements dans un résultat.
$id_connexion = **mssql_pconnect**([$nom_serveur [, $utilisateur [, $mot_passe]]]);
ouvre une connexion persistante à un serveur SQL Server.
$id_resultat \| false = **mssql_query**($requete [, $id_connexion]);
envoie une requête SQL au serveur.
$valeur \| false = **mssql_result**($id_resultat, $num_enregistrement, $position_champs);
retourne la valeur du champs à l'enregistrement spécifié dans un résultat.
true | false = **mssql_select_db**($nom_base, $id_connexion);
sélectionne une base de données SQL Sever.

Exemple

```
<?php
 $id_connexion= mssql_connect("localhost" , "utilisateur" , "motpasse" )
      or die("<h3>Une erreur est survenue :</h3>"
            . mssql_get_last_message() . "</b>");

 mssql_select_db("pubs");

 $id_resultat = mssql_query("select * from employee", $id_connexion);

 echo '<table border="1"><tr>';
 for($i = 0; $i < mssql_num_fields($id_resultat); $i++)
 {
  echo '<th>' . mssql_field_name($id_resultat, $i) . '</th>';
 }
 echo '</tr>';

 for($i = 0; $i < mssql_num_rows($id_resultat); $i++)
 {
  echo '<tr>';
  for($j = 0; $j < mssql_num_fields($id_resultat); $j++))
  {
   echo '<td>' . mssql_result($id_resultat, $i, $j) . '</td>';
  }
  echo '</tr>';
 }
 echo '</table>';

 mssql_close($id_connexion);
?>
```

26.9 / Les fonctions MSQL

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données mSQL.

L'utilisation de cette librairie nécessite l'**activation de l'option de configuration *--with-msql[=répertoire]***, le répertoire par défaut étant */usr/local/Hughes*.

Fonction Description
$id_resultat \| false = **msql**($nom_base, $requete, $id_connexion);
envoie une requête au serveur mSQL.
$nb_enregistrements = **msql_affected_rows**($id_resultat);
retourne le nombre d'enregistrements affectées.
true | false = **msql_close**($id_connexion);
ferme une connexion mSQL.
$id_connexion \| false = **msql_connect**([$nom_hote [, $port_hote [, $utilisateur [, $mot_passe]]]]);
ouvre une connexion mSQL et retourne un identifiant de connexion.
true | false = **msql_create_db**($nom_base, $id_connexion); true \| false = **msql_createdb**($nom_base, $id_connexion);
crée une base de données mSQL.
true | false = **msql_data_seek**($id_resultat, $num_enregistrement);
déplace le pointeur vers un enregistrement désigné par son numéro.
$nom = **msql_dbname**($id_resultat, $position);
retourne le nom de la base de données courante.
true | false = **msql_drop_db**($nom_base, $id_connexion); true \| false = **msql_dropdb**($nom_base, $id_connexion);
supprime une base de données mSQL.
$message = **msql_error**();
retourne le message d'erreur.
$tab_valeurs \| false = **msql_fetch_array**($id_resultat [, MSQL_ASSOC | MSQL_NUM | MSQL_BOTH]);
retourne les valeurs d'un enregistrement dans un tableau associatif (*MSQL_ASSOC*), indicé (*MSQL_NUM*) ou les deux (*MSQL_BOTH*).
$obj_informations = **msql_fetch_field**($id_resultat, $position_champs);
retourne les informations à propos d'un champs dans un objet dont les propriétés sont :

Propriété	Description
name	représente le nom de la colonne.
table	représente le nom de la table d'où la colonne est originaire.
not_null	retourne -1 si la colonne n'est pas NULL.
primary_key	retourne -1 si la colonne est une clé primaire.
unique	retourne -1 si la colonne est une clé unique.
type	retourne le type de la colonne.

$obj_valeurs \| false = **msql_fetch_object**($id_resultat);

retourne les valeurs d'un enregistrement dans un objet.

```
$tab_valeurs | false = msql_fetch_row($id_resultat);
```

retourne les valeurs d'un enregistrement dans un tableau indicé.

```
$nom = msql_fieldname($id_resultat, $position_champs);
```

retourne le nom d'un champs.

```
$position = msql_field_seek($id_resultat $position_champs);
```

détermine la position du pointeur de champs.

```
$nom = msql_fieldtable($id_resultat, $position_champs);
```

retourne le nom de la table d'où provient le champs.

```
$type = msql_fieldtype($id_resultat, $position_champs);
```

retourne le type de champs.

```
$semaphore = msql_fieldflags($id_resultat, $position_champs);
```

retourne le sémaphore d'un champs.

```
$longueur = msql_fieldlen($id_resultat, $position_champs);
```

retourne la longueur d'un champs.

```
true | false = msql_free_result($id_resultat);
true | false = msql_freeresult($id_resultat);
```

libère les ressources consommées par le résultat.

```
$id_liste = msql_list_fields($nom_base, $nom_table);
$id_liste = msql_listfields($nom_base, $nom_table);
```

retourne la liste des champs dans une table désignée par un identifiant.

```
$id_liste = msql_list_dbs();
$id_liste = msql_listdbs();
```

retourne une liste des bases de données mSQL sur un serveur.

```
$id_liste = msql_list_tables($nom_base);
$id_liste = msql_listtables($nom_base);
```

retourne une liste des tables mSQL sur une base de données.

```
$nb_champs = msql_num_fields($id_resultat);
$nb_champs = msql_numfields($id_resultat);
```

retourne le nombre de champs dans un résultat.

```
$nb_enregistrements = msql_num_rows($id_resultat);
$nb_enregistrements = msql_numrows($id_resultat);
```

retourne le nombre d'enregistrements dans un résultat.

```
$id_connexion | false = msql_pconnect([$nom_hote
[, $port_hote [, $utilisateur [, $mot_passe]]]]);
```

ouvre une connexion persistante à un serveur mSQL.

```
$id_resultat | false = msql_query($requete, $id_connexion);
```

envoie une requête mSQL et retourne un identifiant de résultat.

```
$chaine_preparee = msql_regcase($expression_reguliere);
```

prépare une chaîne pour une recherche par expression régulière insensible à la casse.

```
$valeur | false = msql_result($id_resultat,
$num_enregistrement , $position_champs);
```

retourne la valeur d'un champs désigné par sa position ou son nom dans un enregistrement.

```
true | false = msql_select_db($nom_base, $id_connexion);
true | false = msql_selectdb($nom_base, $id_connexion);
```

sélectionne une base de données mSQL.
$nom = **msql_tablename**($id_resultat, $position_champs);
retourne le nom de la table d'où est postionné le champs spécifié.

Exemple

```
<!-- Fichier : formulaire.html -->
<html>
 <body>
 <h3>Ajout d'un enregistrement<h3>
  <form action="traitement.php" method="POST">
   <table border="0" summary="">
    <tr>
     <td>Nom</td>
     <td><input type="text" name="nom"></td>
    </tr>
    <tr>
     <td>Prénom</td>
     <td><input type="text" name="prenom"></td>
    </tr>
    <tr>
     <td>eMail</td>
     <td><input type="text" name="email"></td>
    </tr>
    <tr>
     <td></td>
     <td><input type="submit" name="soumettre" value="Soumettre"></td>
    </tr>
   </table>
  </form>
 </body>
</html>

<!-- Fichier : traitement.html -->
<html>
 <body>
  <h3>Enregistrement ajouté</h3>
  <?php
   $id_connexion = msql_connect("serveur", 1114, "utilisateur", "mot_passe");
         or die("Une erreur s'est produite :<br>" . msql_error());

   msql_select_db($id_connexion, "bd_personnel")
         or die("Une erreur s'est produite :<br>" . msql_error());

   $date_entree = date("d/m/Y");
   $insertion = "insert into tbl_employee "
            . "values ('$date_entree', '$nom', '$prenom', '$email')";
   msql_query($sock, $insertion)
         or die("Une erreur s'est produite :<br>" . msql_error());

   $resultat = msql_query($sock, "select * from tbl_employee")
         or die("Une erreur s'est produite :<br>" . msql_error());

   $i = 0;
   echo "<table><tr>";
   while($enregistrement = msql_fetch_array($resultat, MSQL_ASSOC))
   {
    $entete = array_keys($enregistrement);
    $contenu = array_values($enregistrement);

    if($i == 0)
    {
     while($cle = array_shift($entete))
     {
      echo "<th>" . $cle . "</th>";
     }
    }

    echo "</tr><tr>";

    while($valeur = array_shift($contenu))
    {
     echo "<td>" . $valeur . "</td>";
    }
    $i++;
   }
   echo "</tr></table>";

   msql_free_result($resultat);
```

```
    msql_close($id_connexion);
  %>
 </body>
</html>
```

26.10 / Les fonctions MySQL

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données MySQL.

L'utilisation de ces fonctions nécessite une **compilation de PHP avec le support MySQL en activant l'option *--with-mysql=[répertoire]***. Le répertoire indiqué correspond au chemin où est localisé les librairies du SGBDR MySQL.

Pour en savoir plus sur MySQL et sa configuration, une consultation du site officiel de MySQL semble s'imposer inévitablement.

Fonction
Description
$nb_enregistrements = **mysql_affected_rows**($id_resultat);
retourne le nombre denregistrements affectés lors d'une requête SQL.
true | false = **mysql_change_user**($utilisateur, $mot_passe [, $nom_base [, $id_connexion]]);
modifie le nom de session de l'utilisateur actif.
true | false = **mysql_close**($id_connexion);
ferme la connexion MySQL.
$id_connexion \| false = **mysql_connect**([$nom_hote [, $utilisateur [, $mot_passe]]]);
ouvre une connexion sur un serveur MySQL.
true | false = **mysql_create_db**($nom_base, $id_connexion);
crée une base de données MySQL.
true | false = **mysql_data_seek**($id_resultat, $num_enregistrement);
déplace le pointeur vers l'enregisrement spécifié.
$nom \| false = **mysql_db_name**($id_resultat, $num_enregistrements, $position_champs);
retourne le nom de la base de données par rapport à l'enregistrement et au champs spécifiés.
$id_resultat \| false = **mysql_db_query**($nom_base, $requete [, $id_connexion]);
envoie une requête SQL à un serveur MySQL et retourne un identifiant.
true | false = **mysql_drop_db**($nom_base [, $id_connexion]);
supprime une base de données MySQL.
$num_erreur = **mysql_errno**([$id_connexion]);
retourne le numéro de l'erreur MySQL.
$message = **mysql_error**([$id_connexion]);
retourne le message de l'erreur MySQL.
$nouvelle_chaine = **mysql_escape_string**($chaine);
échappe une chaîne pour la passer à mysql_query.
$tab_valeurs \| false = **mysql_fetch_array**($id_resultat [MYSQL_ASSOC | MYSQL_NUM | MYSQL_BOTH]);
retourne les valeurs d'un enregistrement dans un tableau associatif (*MYSQL_ASSOC*), indicé (*MYSQL_NUM*) ou les deux (*MYSQL_BOTH*).
$tab_valeurs \| false = **mysql_fetch_assoc**($id_resultat);
retourne les valeurs d'un enregistrement dans un tableau associatif.

$obj_informations | false = **mysql_fetch_field**($id_resultat, $position_champs);

retourne les informations à propos du champs spécifié dans un objet dont les propriétés sont :

Propriété	Description
name	représente le nom de la colonne.
table	représente le nom de la table d'où la colonne est originaire.
max_length	représente la taille maximum de la colonne.
not_null	retourne -1 si la colonne n'est pas NULL.
multiple_key	retourne -1 si la colonne est une clé non-unique.
primary_key	retourne -1 si la colonne est une clé primaire.
unique_key	retourne -1 si la colonne est une clé unique.
type	retourne le type de la colonne.
blob	retourne -1 si la colonne est de type BLOB.
numeric	retourne -1 si la colonne est numérique.
unsigned	retourne -1 si la colonne est non-signée.
zerofill	retourne -1 si la colonne est complétée par des zéros.

$tab_tailles = **mysql_fetch_lengths**($id_resultat);

retourne la taille de chaque colonne d'une ligne de résultat.

$obj_valeurs = **mysql_fetch_object**($id_resultat);

retourne les lignes résultats sous la forme d'un objet.

$tab_valeurs | false = **mysql_fetch_row**($id_resultat);

retourne les valeurs d'un enregistrement dans'un tableau.

$semaphore = **mysql_field_flags**($id_resultat, $position_champs);

retourne le sémaphore associé au champs spécifié dans le résultat.

$nom = **mysql_field_name**($id_resultat, $position_champs);

retourne le nom d'un champ.

$longueur = **mysql_field_len**($id_resultat, $position_champs);

retourne la longueur d'un champ.

$position = **mysql_field_seek**($id_resultat, $position_champs);

déplace le pointeur vers un champ spécifié.

$nom = **mysql_field_table**($id_resultat, $position_champs);

retourne le nom de la table où se trouve le champ.

$type = **mysql_field_type**($id_resultat, $position_champs);

retourne le type du champ spécifié.

true | false = **mysql_free_result**($id_resultat);

libére les ressources utilisées par le résultat.

$chaine = **mysql_get_client_info**();

retourne le numéro de version du client MySQL.

$chaine = **mysql_get_host_info**([$id_connexion]);

retourne une chaîne de caractères indiquant le type de connexion et le nom du serveur hôte.

$chaine = **mysql_get_proto_info**([$id_connexion]);

retourne une chaîne de caractères indiquant la version du protocole utilisée par la connexion.
$chaine = **mysql_get_server_info**([$id_connexion]);
retourne une chaîne de caractères indiquant la version du serveur utilisée par la connexion.
$id_insertion = **mysql_insert_id**([$id_connexion]);
retourne l'identifiant généré par la dernière requête d'insertion INSERT.
$id_liste = **mysql_list_dbs**([$id_connexion]);
retourne la liste des bases de données sur le serveur MySQL.
$id_liste = **mysql_list_fields**($nom_base, $nom_table [, $id_resultat]);
retourne la liste des champs d'un résultat MySQL.
$id_liste = **mysql_list_tables**($nom_base [, $id_connexion]);
retourne la liste des tables d'une base de données.
$nb_champs = **mysql_num_fields**([$id_resultat]);
retourne le nombre de champs d'un résultat.
nb_enregistrements = **mysql_num_rows**([$id_resultat]);
retourne le nombre d'enregistrements dans un résultat.
$id_connexion \| false = **mysql_pconnect**([$nom_hote [, $utilisateur [, $mot_passe]]]);
ouvre une connexion persistante à un serveur MySQL.
$id_resultat\| false = **mysql_unbuffered_query**($requete [, $id_connexion [, $mode]]);
exécute une requête SQL en évitant de consommer de la mémoire.
$id_resultat \| false = **mysql_query**($requete [, $id_connexion]);
exécute une requête SQL sur un serveur MySQL.
$valeur = **mysql_result**($id_resultat, $num_enregistrement [, $champ]);
retourne la valeur d'un champ désigné par sa position ou son nom dans un enregistrement indiqué.
true | false = **mysql_select_db**($nom_base [, $id_connexion]);
sélectionne une base de données MySQL.
$nom = **mysql_tablename**($id_resultat, $position_champ);
retourne le nom de la table d'où le champ spécifié provient.

Exemple [voir]

```php
<?php
 function gestion_erreur($identifiant)
 {
  echo "Une erreur est survenue :<br>"
        . mysql_errno() . " : " . mysql_error();
  mysql_close($identifiant);
  exit;
 }

 $id_connexion = mysql_connect("localhost","root","mpt");
 if(!$id_connexion) gestion_erreur($id_connexion);

 $id_liste_bases = mysql_list_dbs($id_connexion);
 if(!$id_liste_bases) gestion_erreur($id_liste_bases);

 $nb_bases = mysql_num_rows($id_liste_bases);

 echo '<table width="640" border="1">';
 for($i = 0; $i < $nb_bases; $i++)
 {
  $nom_base = mysql_db_name($id_liste_bases, $i);

  $id_liste_tables = mysql_list_tables($nom_base, $id_connexion);
  if(!$id_liste_tables) gestion_erreur($id_liste_tables);

  $nb_tables = mysql_num_rows($id_liste_tables);

  echo '<tr><th valign="top" width="140"><u>' . $nom_base
          . '</u></th><td valign="top" width="500">';
  for($j = 0; $j < $nb_tables; $j++)
  {
   $nom_table = mysql_tablename($id_liste_tables, $j);

   $id_liste_champs =
              mysql_list_fields($nom_base, $nom_table, $id_connexion);
   if(!$id_liste_champs) gestion_erreur($id_liste_champs);

   $nb_champs = mysql_num_fields($id_liste_champs);

   echo '<table width="500" border="1"><tr><th valign="top" width="200">'
        . $nom_table . '</th><td valign="top" width="300">';
   for($k = 0; $k < $nb_champs; $k++)
   {
    $nom_champ = mysql_field_name($id_liste_champs, $k);
    $type_champ = mysql_field_type($id_liste_champs, $k);
    $longueur_champ = mysql_field_len($id_liste_champs, $k);
    $semaphore_champ = mysql_field_flags($id_liste_champs, $k);

    echo '<u>' . $nom_champ . '</u>('
          . $type_champ . ' '
          . $longueur_champ . ' '
          . $semaphore_champ . ')<br>';
   }
   echo '</td></tr></table>';
  }
  echo '</td></tr>';
 }
 echo '</table>';

 mysql_free_result($id_liste_bases);
 mysql_free_result($id_liste_tables);
 mysql_free_result($id_liste_champs);

 mysql_close($id_connexion);
?>
```

26.11 / Les fonctions ODBC

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données ODBC (Open DataBase Connectivity).

Une compilation de PHP avec un support iODBC disponible sur iODBC.org, UnixODBC ou encore sur OpenLink, permet d'utiliser cette librairie de fonction avec de nombreux gestionnaires de bases de données compatibles ODBC.

Fonction
Description
true | false = **odbc_autocommit**($id_connexion [, 0 | 1]);
active (*1*) ou désactive (*0*) le mode d'auto-validation.
= **odbc_binmode**($id_resultat, $mode);
modifie la gestion des colonnes de données binaires selon un mode donné.

Mode	Description
ODBC_BINMODE_PASSTHRU	représente le mode Passthru.
ODBC_BINMODE_RETURN	retourne les données telles quelles.
ODBC_BINMODE_CONVERT	retourne les données converties en caractères.

odbc_close($id_connexion);
ferme une connexion ODBC.
odbc_close_all();
ferme toutes les connexions ODBC.
true | false = **odbc_commit**($id_connexion);
valide une transaction ODBC.
$id_connexion = **odbc_connect**($DSN, $utilisateur, $mot_passe [, $type_curseur]);
ouvre une connexion à une source de données.

Mode	Description
SQL_CUR_USE_IF_NEEDED	représente le mode Passthru.
SQL_CUR_USE_ODBC	retourne les données telles quelles.
SQL_CUR_USE_DRIVER	retourne les données converties en caractères.
SQL_CUR_DEFAULT	retourne les données converties en caractères.

$chaine = **odbc_cursor**($id_resultat);
retourne le pointeur de la fiche courante (*cursorname*).
$chaine = **odbc_do**($id_connexion, $requete);
Synonyme de odbc_exec().
$code_erreur = **odbc_error**([$id_connexion]);
retourne le code de l'erreur.
$message = **odbc_errormsg**([$id_connexion]);
retourne le message de l'erreur.
$id_resultat \| false = **odbc_exec**($id_connexion, $requete);
prépare et exécute une requête SQL puis retourne un identifiant de résultat.
true | false = **odbc_execute**($id_connexion [, $tab_parametres]);
exécute une requête SQL préparée par *odbc_prepare*.
$nb_champs \| false = **odbc_fetch_into**($id_resultat [, $num_enregistrement, $tab_resultats]);
retourne le nombre de champs de l'enregistrement spécifié et place ses valeurs dans le tableau de résultats.
true | false = **odbc_fetch_row**($id_resultat [, $num_enregistrement]);
lit un enregistrement spécifié.

$nom | false = **odbc_field_name**($id_resultat, $position_champ);

retourne le nom du champ désigné.

$num_champs | false = **odbc_field_num**($id_resultat, $nom_champ);

retourne le numéro du champ.

$type | false = **odbc_field_type**($id_resultat, $position_champ);

retourne le type de données d'un champ.

$longueur | false = **odbc_field_len**($id_resultat, $position_champ);
$longueur | false = **odbc_field_precision**($id_resultat, $position_champ);

retourne la longueur d'un champ.

$echelle = **odbc_field_scale**($id_resultat, $position_champ);

retourne l'échelle d'un champ.

true = **odbc_free_result**($id_resultat);

libère les ressources utilisées par un résultat.

$nombre = **odbc_longreadlen**($id_resultat [, $nb_octets]);

détermine le nombre d'octets pour des champs de type LONG et LONGVARBINARY.

$nb_champs = **odbc_num_fields**($id_resultat);

retourne le nombre de champs dans un résultat.

$id_connexion = **odbc_pconnect**($DSN, $utilisateur, $mot_passe
[, $type_curseur]);

ouvre une connexion persistante à une source de données.

$id_preparation = **odbc_prepare**($id_connexion, $requete);

prépare une requête pour une exécution par *odbc_execute*.

$nb_enregistrements = **odbc_num_rows**($id_resultat);

retourne le nombre d'enregistrements dans un résultat.

$valeur = **odbc_result**($id_resultat, $champ);

retourne la valeur d'un champ désigné par son nom ou sa position.

$nb_enregistrements | false = **odbc_result_all**($id_resultat [, $format]);

affiche le résultat sous la forme d'une table HTML et retourne le nombre d'enregistrements.
L'argument *$format* est une chaîne de caractères définissant le format d'affichage des valeurs.

true | false = **odbc_rollback**($id_connexion);

annule une transaction.

$nombre = **odbc_setoption**($identifiant,
$num_fonction, $num_option, $num_param);

modifie les paramètres ODBC de la connexion ou d'un résultat de requête.

$id_resultat = **odbc_tables**(id_connexion
[, $qualificateur [, $proprietaire [, $nom [, $types]]]]);

liste les tables d'une source selon différents paramètres.

$id_resultat = **odbc_tableprivileges**($id_connexion
[, $qualificateur [, $proprietaire [, $nom]]]);

liste les tables et leurs privilèges selon différents paramètres.

$id_resultat = **odbc_columns**($id_connexion
[, $qualificateur [, $proprietaire [, $nom_table [, $nom_colonne]]]]);

liste les colonnes d'une table selon différents paramètres.

$id_resultat = **odbc_columnprivileges**($id_connexion
[, $qualificateur [, $proprietaire [, $nom_table [, $nom_colonne]]]]);

liste les colonnes et leurs droits associés selon différents paramètres.

```
$id_resultat = odbc_gettypeinfo($id_connexion [, $type_donnee]);
```

liste les types de données supportés par une source selon différents paramètres.

```
$id_resultat = odbc_primarykeys($id_connexion,
$qualificateur, $proprietaire, $nom_table);
```

liste les colonnes utilisées dans une clé primaire selon différents paramètres.

```
$id_resultat = odbc_foreignkeys($id_connexion,
$pk_qualificateur, $pk_proprietaire, $pk_nom_table,
$fk_qualificateur, $fk_proprietaire, $fk_nom_table);
```

liste les clés étrangères (*fk*) selon différents paramètres.

```
$id_resultat = odbc_procedures($id_connexion
[, $qualificateur [, $proprietaire [, $nom]]]);
```

liste les procédure stockées selon différents paramètres.

```
$id_resultat = odbc_procedurecolumns($id_connexion
[, $qualificateur [, $proprietaire [, $nom_procedure [, $nom_colonne]]]]);
```

liste les paramètres des procédures selon différents paramètres.

```
$id_resultat = odbc_specialcolumns(id_connexion, $qualificateur,
$proprietaire, $nom_table, $num_etendue, $num_annulable);
```

retourne l'ensemble optimal de champs permettant de définir uniquement un enregistrement dans une table.

```
$id_resultat = odbc_statistics($id_connexion, $qualificateur,
$proprietaire, $nom_table, $num_unique, $num_exactitude);
```

calcule des statistiques sur une table.

Exemple

```php
<?php
// Fichier : commun.inc
 function gestion_erreur($identifiant)
 {
  echo "Une erreur est survenue :<br>"
       . odbc_error() . " : " . odbc_errormsg();
  odbc_close($identifiant);
  exit;
 }

 function affichage_base($identifiant)
 {
  $resultat = odbc_exec($identifiant, "select * from tbl_personnel");
  if (!$resultat) gestion_erreur($resultat);

  $nb_enregistrements = 0;
  $nb_champs = odbc_num_fields($resultat);

  echo '<table border="1"><tr>';
  for($i = 1; $i <= $nb_champs; $i++)
  {
   echo '<th>' . odbc_field_name($identifiant, $i) . '</th>';
  }
  echo '</tr><tr>';
  while(odbc_fetch_row($resultat))
  {
   for($i = 1; $i < $nb_champs; $i++)
   {
    echo '<td>' . odbc_result($resultat, $i) . '</td>';
   }
   $nb_enregistrements++;
  }

  echo '</tr><tr>'
       . '<td colspan="' . $nb_champs . '">'
       . $nb_enregistrements . ' enregistrements</td>'
       . '</tr></table>';
 }
?>

<?php
// Fichier : traitement.php
 include ("commun.inc");

 function ajout_entree($tab_valeurs)
 {
  $i = 0;
  $nb = sizeof($tab_valeurs);
  $requete_sql = "Insert Into tbl_personnel (";
  while($cle = array_shift(array_keys($tab_valeurs)))
  {
   $fin = $i < $nb - 1 ? ", " : ") ";
   $requete_sql .= $cle . $fin;
   $i++;
  }
  $requete_sql = "Values ('";
  foreach($i = 0; $i < $tab_valeurs; $i++)
  {
   $fin = $i < $nb - 1 ? "', " : "')";
   $requete_sql .= "'" . $tab_valeur[$i] . $fin;
  }

  $resultat = odbc_exec($id_connexion, $requete_sql );
  if (!$resultat) gestion_erreur($resultat);

  return "<h3La mise à jour de la base de données a été réussie !</h3>";
 }

 $id_connexion = odbc_connect("serveur", 'utilisateur', 'mpoatsse');
 if (!$id_connexion) gestion_erreur($id_connexion);

 $informations = array("nom"=>$nom,
                       "prenom"=>$prenom,
```

```
                    "email"=>$email);

  echo ajout_entree($informations);
  affichage_base($id_connexion);

  odbc_close($id_connexion);
?>
<!-- Fichier : formulaire.php -->
<?php include ("commun.inc"); ?>
<html>
  <body>
  <h2>Enregistrements de la base de données<h2>
  <?php
     $id_connexion = odbc_connect("serveur", 'utilisateur', 'mpoatsse');
     if (!$id_connexion) gestion_erreur($id_connexion);

     affichage_base();

     odbc_close($id_connexion);
  ?>
  <h3>Ajout d'un enregistrement<h3>
  <form action="traitement.php" method="POST">
     <table border="0">
       <tr>
       <td>Nom</td>
       <td><input type="text" name="nom"></td>
       </tr>
       <tr>
       <td>Prénom</td>
       <td><input type="text" name="prenom"></td>
       </tr>
       <tr>
       <td>eMail</td>
       <td><input type="text" name="email"></td>
       </tr>
       <tr>
       <td></td>
       <td><input type="submit" name="soumettre" value="Soumettre"></td>
       </tr>
     </table>
  </form>
  </body>
</html>
```

26.12 / Les fonctions Oracle

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données Oracle version 7 et 8.

La configuration de PHP pour Oracle est longuement expliquée sur le site PHPBuilder.com ou sur tldp.org.

Fonction
Description
true | false = **Ora_Bind**($id_curseur, $variable_php, $parametre_sql, $longueur [, $type]);
réalise une liaison entre une variable PHP et un paramètre Oracle.
true | false = **Ora_Close**($id_curseur);
ferme un pointeur Oracle.
$nom = **Ora_ColumnName**($id_curseur, $position_champ);
retourne le nom du champ du résultat.
$taille = **Ora_ColumnSize**($id_curseur, $position_champ);
retourne la taille du champ d'un résultat.
$type = **Ora_ColumnType**($id_curseur, $position_champ);
retourne le type du champ d'un résultat.
true | false = **Ora_Commit**($id_connexion);
valide une transaction Oracle.
true | false = **Ora_CommitOff**($id_connexion);
désactive la validation automatique.
true | false = **Ora_CommitOn**($id_connexion);
active la validation automatique.
true | false = **Ora_Do**($id_connexion, $requete);
prépare, exécute une requête et lit le premier enregistrement du résultat.
$message = **Ora_Error**($id_connexion);
Retourne le message d'erreur Oracle.
$code_erreur = **Ora_ErrorCode**($id_connexion);
Retourne le code d'erreur Oracle.
true | false = **Ora_Exec**($id_curseur);
exécute une requête analysée sur un pointeur Oracle.
true | false = **Ora_Fetch**($id_curseur);
retourne un enregistrement d'un résultat.
true | false = **Ora_Fetch_Into**($id_curseur, $tab_resultat [, $semaphore]);
retourne un enregistrement dans le tableau de résultat.
$valeur = **Ora_GetColumn**($id_curseur, $champ);
retourne la valeur d'un champ désigné par son nom ou sa position.
true | false = **Ora_Logoff**($id_connexion);
ferme une connexion Oracle.
$id_connexion = **Ora_Logon**($utilisateur, $mot_passe);
ouvre une connexion Oracle.
$id_connexion = **Ora_pLogon**($utilisateur, $mot_passe);
ouvre une connexion persistante à Oracle.
$nb_champs = **Ora_Numcols**($id_curseur);
retourne le nombre de champs.
$nb_enregistrements = **Ora_Numrows**($id_curseur);

retourne le nombre d'enregistrements.

$id_curseur = **Ora_Open**($id_connexion);

ouvre un pointeur Oracle.

true | false = **Ora_Parse**($id_curseur, $requete, $nb_report);

analyse une requête SQL.

true | false = **Ora_Rollback**($id_connexion);

annule une transaction.

Fonctions Oracle 8

$nombre = **ociDefineByName**($id_resultat, $nom_champ, $variable [, $type]);

utilise une variable PHP pour la phase de définition, dans une commande SELECT.

$nombre = **ociBindByName**($id_resultat, $nom_oracle,
$variable_php, $longueur [, $type]);

utilise une variable PHP pour la phase de définition dans un SELECT.

$id_connexion | false = **ociLogon**($utilisateur, $mot_passe [, $nom_base]);

ouvre une connexion à un serveur Oracle.

$id_connexion | false = **ociPLogon**($utilisateur, $mot_passe [, $nom_base]);

ouvre une connecxion persistante à un serveur Oracle.

$id_connexion | false = **ociNLogon**($utilisateur, $mot_passe [, $nom_base]);

ouvre une nouvelle connexion à un serveur Oracle.

true | false = **ociLogOff**($id_connexion);

ferme une connexion à un serveur Oracle.

= **ociexecute**($id_resultat [oci_COMMIT_ON_SUCCESS | oci_DEFAULT]);

exécute une requête SQL préparée par *ociparse*.

true | false = **ociCommit**($id_connexion);

valide les transactions en cours.

true | false = **ociRollback**($id_connexion);

annule les transactions en cours.

$id_descripteur = **ociNewDescriptor**($id_connexion
[, oci_D_FILE | oci_D_LOB | oci_D_ROWID]);

initialise un nouveau pointeur vide de LOB/FILE.

$nb_enregistrements = **ociRowCount**($id_resultat);

retourne le nombre d'enregistrements dans un résultat.

$nb_champs = **ociNumCols**($id_resultat);

retourne le nombre de champs dans un résultat.

$valeur = **ociResult**($id_resultat, $champ);

retourne la valeur d'un champ désigné par son nom ou sa position dans la ligne courante d'un résultat.

$position = **ociFetch**($id_resultat);

place le prochain enregistrement dans le pointeur de résultat.

true | false = **ociFetchInto**($id_resultat, $tab_resultat, $mode);

retourne la ligne suivante dans un tableau.

Mode	Description
oci_ASSOC	retourne un tableau associatif.

oci_NUM	retourne un tableau indicé.
oci_RETURN_NULLS	retourne les colonnes vides.
oci_RETURN_LOBS	retourne la valeur des objets LOB plutôt que leur descripteur.

true | false = **ociFetchStatement**($id_resultat, $tab_variables);

retourne toutes les lignes d'un résultat dans le tableau de variables.

true | false = **ociColumnIsNULL**($id_resultat, $champ);

vérifie si la valeur d'un champ est égale à NULL.

$nom = **ociColumnName**($id_resultat, $position_champ);

retourne le nom d'un champ.

$taille = **ociColumnSize**($id_resultat, $champ);

retourne la taille d'un champ.

$type = **ociColumnType**($id_resultat, $champ);

retourne le type de données d'un champ.

$version = **ociServerVersion**($id_connexion);

retourne une chaîne de caractères contenant le numéro de version du serveur.

$chaine = **ociStatementType**($id_resultat);

retourne le type de commande SQL (*select*, *UPDATE*, *INSERT*, etc.).

$id_curseur = **ociNewCursor**($id_connexion);

retourne un nouveau pointeur à utiliser pour lier les pointeurs de références.

true | false = **ociFreeStatement**($id_resultat);

libère toutes les ressources occupées par une commande.

true | false = **ociFreeCursor**($id_curseur);

libère toutes les ressources occupées par un pointeur.

true | false = **ociFreeDesc**($id_descripteur);

supprime un descripteur de LOB.

true | false = **ociparse**($id_connexion, $requete);

analyse une requête SQL.

$tableau | false = **ociError**($identificateur);

retourne la dernière erreur d'une connexion, d'un résultat ou globale.

ociinternaldebug(0 | 1);

active ou désactive l'affichage des données de deboguage.

true | false = **OCICancel**($id_resultat);

détruit les ressources liées au dernier résultat.

true | false = **ocisetprefetch**($id_resultat, $nb_enregistrements);

indique le nombre d'enregistrements devant être pré-lues.

OCIWriteLobToFile($obj_lob [, $fichier [, $nb_depart [, $longueur]]]);

écrit un objet LOB dans un fichier.

$chaine = **OCISaveLobFile**($obj_lob);

retourne un objet LOB dans une chaîne de caractères.

$chaine = **OCISaveLob**($obj_lob);

retourne un objet LOB dans une chaîne de caractères.

$chaine = **OCILoadLob**($obj_lob);
charge un objet LOB.
$echelle = **OCIColumnScale**($id_resultat, $num_champ);
retourne l'échelle d'un champ numérique.
$precision = **OCIColumnPrecision**($id_resultat, $num_champ);
retourne la précision d'un champ numérique.
$valeur = **OCIColumnTypeRaw**($id_resultat, $num_champ);
retourne le type brut d'un champ.
$chaine = **OCINewCollection**($id_connexion, $tdo, $schema);
crée une nouvelle collection.
$chaine = **OCIFreeCollection**($obj_lob);
libère une collection.
$chaine = **OCICollAssign**($obj_collection, $objet);
assigne un objet à une collection.
$valeur = **OCICollAssignElem**($obj_collection, $ndx, $valeur);
assigne un élément à une collection.
$chaine = **OCICollGetElem**($obj_collection, $ndx);
retourne la valeur d'un élément d'une collection.
$chaine = **OCICollMax**($obj_collection);
retourne la longueur maximum de la collection.
$chaine = **OCICollSize**($obj_collection);
retourne la taille de la collection.
$chaine = **OCICollTrim**($obj_collection, $nombre);
enlève les espaces de début et de fin de la collection.

Exemple

```php
<?php
 $id_connexion = ociLogon("utilisateur" , "motpasse" "bd_personnel")
      or die("<h3>Une erreur est survenue :</h3>"
            . ociError($id_connexion) . "</b>");

 $preparation = ociParse($id_connexion, "select * from employee")
 $id_resultat = ociExecute($preparation, oci_DEFAULT);

 echo '<table border="1"><tr>';
 for($i = 0; $i < ociNumCols($id_resultat); $i++)
 {
  echo '<th>' . ociColumnName($id_resultat, $i)
          . '<br>' . ociColumType($id_resultat, $i) . '('
          . ociColumnSize($id_resultat, $i) . ')</th>';
 }
 echo '</tr>';

 for($i = 0; $i < ociRowsCount($id_resultat); $i++)
 {
  echo '<tr>';
  for($j = 0; $j < ociNumCols($id_resultat); $j++))
  {
   echo '<td>' . ociResult($id_resultat, $i, $j) . '</td>';
  }
  echo '</tr>';
 }
 echo '</table>';

 ociCancel($id_resultat);

 ociLogoff($id_connexion);
?>
```

26.13 / Les fonctions PostgreSQL

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données PostgreSQL.

Fonction
Description
true | false = **pg_close**($id_connexion);
ferme une connexion PostgreSQL.
$nb_instances \| false = **pg_affected_rows**($id_resultats);
retourne le nombre d'instances (tuples) affectées. (PHP < 4.2 : **pg_cmdTuples**)
$id_connex = **pg_connect**({$nom_hote, $port_hote [, $options [, $tty [, $nom_base]]]} | $chaine_connection);
ouvre une connexion à un serveur PostgreSQL.
$nom = **pg_dbname**($id_connexion);
retourne le nom de la base de données.
true | false = **pg_end_copy**($id_connexion);
synchronise le client avec le serveur PostgreSQL.
$message = **pg_ErrorMessage**($id_connexion);
retourne un message d'erreur.
$id_resultat = **pg_query**($id_connexion, $requete);
exécute une requête sur le serveur et retourne un identifiant de résultat. (PHP < 4.2 : pg_exec)
$tab_valeurs \| false = **pg_fetch_array**($id_resultat, num_enregistrement [, $type]);
retourne les valeurs d'un enregistrement dans un tableau d'un type associatif (*PGSQL_ASSOC*), indicé (*PGSQL_NUM*) ou des deux (*PGSQL_BOTH*).
$obj_valeurs = **pg_fetch_object**($id_resultat, $num_enregistrement);
retourne les valeurs d'un enregistrement dans un objet.
$tab_valeurs = **pg_fetch_row**($id_resultat, $num_enregistrement);
retourne les valeurs d'un enregistrement dans un tableau indicé.
true | false = **pg_field_is_null**($id_resultat, $num_enregistrement, $position_champ);
vérifie si un champ possède une valeur égale à NULL. (PHP < 4.2 : pg_FieldIsNull)
$chaine = **pg_field_name**($id_resultat, $position_champ);
retourne le nom d'un champ dans un résultat. (PHP < 4.2 : pg_FieldName)
$position \| -1 = **pg_field_num**($id_resultat, $nom_champ);
retourne la position d'un champ dans un résultat. (PHP < 4.2 : pg_FieldNum)
$taille \| -1 = **pg_field_prtlen**($id_resultat, $num_enregistrement, $position_champ);
retourne la taille imprimée du champ d'un enregistrement dans un résultat. (PHP < 4.2 : pg_FieldPrtLen)
$taille\| false = **pg_field_size**($id_resultat, $position_champ);
retourne la taille en octets d'un champ dans un résultat. (PHP < 4.2 : pg_FieldSize)
$type \| false = **pg_field_type**($id_resultat, $position_champ);

retourne le type d'un champ dans un résultat. Si la taille est variable, la valeur *-1* sera retournée. (PHP < 4.2 : pg_FieldType)

```
true | false = pg_free_result($id_resultat);
```

libère les ressources consommées par un résultat. (PHP < 4.2 : pg_FreeResult

```
$identifiant = pg_last_oid($id_resultat);
```

retourne le dernier identifiant d'objet. (PHP < 4.2 : pg_GetLastOid)

```
$nom_hote = pg_host($id_connexion);
```

retourne le nom de l'hôte pour la connexion spécifiée.

```
pg_lo_close($id_lo);
```

ferme un objet de grande taille (LO : Large Object). (PHP < 4.2 : pg_loclose)

```
$id_lo = pg_lo_create($id_connexion);
```

crée un objet de grande taille. (PHP < 4.2 : pg_locreate)

```
true | false = pg_lo_export($id_lo, $id_fichier [, $id_connexion]);
```

exporte un objet de grande taille vers un fichier. (PHP < 4.2 : pg_loexport)

```
true | false = pg_loi_mport($id_fichier [, $id_connexion]);
```

importe un objet de grande taille à partir d'un fichier. (PHP < 4.2 : pg_loimport)

```
$id_lo = pg_lo_open($id_connexion, $obj_lo, {r | w | rw});
```

ouvre un objet de grande taille. (PHP < 4.2 : pg_loopen)

```
$chaine = pg_lo_read($id_lo, $longueur);
```

retourne un objet de grande taille dans une chaîne de caractères. (PHP < 4.2 : pg_loread)

```
pg_lo_read_all($id_lo);
```

lit entièrement un objet de grande taille. (PHP < 4.2 : pg_loreadall)

```
= pg_lo_unlink($id_connexion, $id_lo);
```

supprime un objet de grande taille (PHP < 4.2 : pg_lounlink).

```
$nb_octets | false = pg_lo_write($id_lo, $donnee);
```

écrit des données dans un objet de grande taille. (PHP < 4.2 : pg_lowrite)

```
$nb_champs | -1 = pg_num_fields($id_resultat);
```

retourne le nombre de champs dans un résultat. (PHP < 4.2 : pg_NumFields)

```
$nb_enregistrements | -1 = pg_num_rows($id_resultat);
```

retourne le nombre d'enregistrements dans un résultat. (PHP < 4.2 : pg_NumRows)

```
$option = pg_options($id_connexion);
```

retourne les options d'une connexion SQL.

```
$id_connexion = pg_pconnect($chaine_connexion);
```

ouvre une connexion persistante sur un serveur PostgreSQL.

```
$num_port = pg_port($id_connexion);
```

retourne le numéro de port pour la connexion indiquée.

```
true | false = pg_put_line($id_connexion, $chaine);
```

envoie une chaîne de caractères au serveur PostgreSQL.

```
$valeur = pg_fetch_result($id_resultat, $num_enregistrement, $position_champ);
```

retourne la valeur d'un champ dans un enregistrement d'un résultat. (PHP < 4.2 : pg_result)

```
true | false = pg_set_client_encoding($id_connexion, $encodage);
```

détermine l'encodage du client (SQL_ASCII, UNICODE, LATINX, WIN, BIG5, WIN1250).

$encodage = **pg_client_encoding**($id_connexion);
retourne l'encodage du client.
true | false = **pg_trace**($fichier [, $mode [, $id_connexion]]);
active le suivi d'une connexion PostgreSQL.
$chaine = **pg_tty**($id_connexion);
retourne le nom tty de la connexion.
true | false = **pg_untrace**($id_connexion);
arrête le suivi d'une connexion PostgreSQL.

Exemple

```
<?php
 function gestion_erreur($identifiant)
 {
  echo "Une erreur s'est produite : " . pg_errormessage($id_connexion);
  exit;
 }

 $id_connexion = pg_pconnect("user=utilisateur "
                             . "password=essapmot "
                             . "dbname=tbl_personnel");
 if (!$id_connexion) gestion_erreur($id_connexion);

 $date = date("d/m/Y");
 $insertion = pg_exec($id_connexion,
                      "INSERT INTO tbl_personnel (date, nom, prenom, email) "
                    . "VALUES (" . $date . "'" . $nom . "', '"
                    . $prenom . "', '" . $email . "')");

 $selection = pg_exec($id_connexion, "SELECT * FROM tbl_personnel");

 $i = 0;
 echo "<table border=\"1\"><tr>";
 while($enregistrement = pg_fetch_object($id_connexion, $i))
 {
  $informations = get_object_vars($enregistrement);
  $entete = array_keys($informations);
  $contenu = array_values($informations);

  if($i == 0)
  {
   while($cle = array_shift($entete))
   {
    echo "<th>" . $cle . "</th>";
   }
  }

  echo "</tr><tr>";

  while($valeur = array_shift($contenu))
  {
   echo "<td>" . $valeur . "</td>";
  }
  $i++;
 }
 echo "</tr></table>";
?>
```

26.14 / Les fonctions Sybase

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données Sybase.

Fonction
Description
$nb_enregistrements \| false = **sybase_affected_rows**([$id_connexion]);
retourne le nombre d'enregistrements affectés par une requête.
true | false = **sybase_close**($id_connexion);
ferme une connexion Sybase.
$id_connexion = **sybase_connect**($nom_hote, $utilisateur, $mot_passe);
ouvre une connexion à un serveur Sybase.
true | false = **sybase_data_seek**($id_connexion, $num_enregistrement);
déplace le pointeur d'enregistrements à une position indiquée.
$tab_valeurs = **sybase_fetch_array**($id_resultat);
retourne les valeurs d'un enregistrement dans un tableau.
$obj_informations = **sybase_fetch_field**($id_connexion, $position_champ);
retourne les informations à propos du champ indiqué dans un objet dont les propriétés sont :

Propriété	Description
name	représente le nom de la colonne.
column_source	représente le nom de la table d'où la colonne est originaire.
max_length	retourne la taille maximum d'une colonne.
numeric	retourne -1 si la colonne est numérique.
type	retourne le type de la colonne.

$obj_valeurs \| false = **sybase_fetch_object**($id_resultat);
retourne les valeurs d'un enregistrement dans un objet.
$tab_valeurs \| false = **sybase_fetch_row**($id_resultat);
retourne les valeurs d'un enregistrement dans un tableau indicé.
$position \| false = **sybase_field_seek**($id_resultat, $position_champ);
déplace le pointeur de champs à la position spécifiée.
true | false = **sybase_free_result**($id_resultat);
libère les ressources consommées par un résultat.
$message = **sybase_get_last_message**();
retourne le message d'erreur du serveur Sybase.
sybase_min_client_severity($niveau_severite);
détermine le niveau de sévérité minimum du client.
sybase_min_error_severity($niveau_severite);
détermine le niveau de sévérité minimum du client pour les erreurs.
= **sybase_min_message_severity**($niveau_severite);
détermine le niveau de sévérité minimum du client pour les messages.
= **sybase_min_server_severity**($niveau_severite);
détermine le niveau de sévérité minimum du client pour le serveur.
$nb_champs = **sybase_num_fields**($id_resultat);
retourne le nombre de champ dans un résultat.
$nb_enregistrements = **sybase_num_rows**($id_resultat);

retourne le nombre d'enregistrements dans un résultat.
$id_connexion \| false = **sybase_pconnect**($nom_hote, $utilisateur, $mot_passe);
ouvre une connexion persistante à un serveur Sybase.
$id_resultat \| false = **sybase_query**($requete, $id_connexion);
envoie une requête à une base Sybase.
$valeur = **sybase_result**($id_resultat, $num_enregistrement, $position_champ);
retourne la valeur d'un champ dans un enregistrement spécifié.
true | false = **sybase_select_db**($nom_base, $id_connexion);
sélectionne une base de données Sybase.

Exemple

```
<?php
 $id_connexion= sybase_connect("localhost" , "utilisateur" , "motpasse" )
      or die("<h3>Une erreur est survenue :</h3>"
            . sybase_get_last_message() . "</b>");

 sybase_select_db("pubs");

 $id_resultat = sybase_query("select * from employee", $id_connexion);

 echo '<table border="1"><tr>';
 for($i = 0; $i < sybase_num_fields($id_resultat); $i++)
 {
  echo '<th>' . sybase_field_name($id_resultat, $i) . '</th>';
 }
 echo '</tr>';

 for($i = 0; $i < sybase_num_rows($id_resultat); $i++)
 {
  echo '<tr>';
  for($j = 0; $j < sybase_num_fields($id_resultat); $j++))
  {
   echo '<td>' . sybase_result($id_resultat, $i, $j) . '</td>';
  }
  echo '</tr>';
 }
 echo '</table>';

 sybase_free_result($resultat);

 sybase_close($id_connexion);
?>
```

27 / Le langage XML

Le langage XML est en passe de devenir un support de stockage de données indispensable pour la mise en oeuvre d'un site Web dynamique. Le langage PHP intègre déjà de nombreux outils permettant de travailler en conjonction avec des documents XML.

Un document XML possède des informations parfaitement ordonnées, et partant, exploitables par une application PHP.

```
<?xml version="1.0"?>
<!DOCTYPE racine SYSTEM "http://adresse.web.com/dtD">
<racine>
  <element attribut="Valeur"/>
  <element_2>Contenu_Textuel</element_2>
  <[CDATA[Texte]]>
</racine>
```

Il est possible de **manipuler tous les composants d'une arborescence XML**, c'est-à-dire, de parcourir les différents noeuds, d'extraire les informations contenues dans ces derniers, d'en insérer de nouveaux, de mettre à jour leur contenu ou encore d'en supprimer.

Le langage PHP dispose de plusieurs librairies de fonctions permettant de manipuler des documents XML.

27.1 / La manipulation XML par les fonctions Expat

Les fonctions XML supportant de la librairie *expat* de James Clark, permettent d'analyser et non de valider des documents XML subséquemment à la création d'analyseurs XML et à la définition de points d'entrée pour chaque événement XML.

Les données XML peuvent provenir soit d'une chaîne de caractères interne au script lui-même, soit d'un fichier externe.

```
$donnee = '<?xml version="1.0">'
          '<element_racine>'
          '<element_enfant>Valeur</element_enfant>'
          ...
          '</element_racine>';
```

Dans ce dernier cas, il faut ouvrir le fichier par l'intermédaire de la fonction *fopen*.

```
<?php
 $fichier = "fichier.xml";

 $id_fichier = fopen($fichier);
 $donnee_XML = fread($id_fichier, filesize($fichier));
?>
```

Les informations XML désormais disponibles à partir d'une variable peuvent être traitées par un analyseur XML préalablement créé par la fonction *xml_parser_create*.

```
$id_analyseur = xml_parser_create();
```

Suite à la création d'un analyseur XML, il devient possible de **lancer l'analyse de données XML**.

```
xml_parse($id_analyseur, $donnee_XML))
```

L'analyse d'un document XML peut s'effectuer par morceaux en accord avec la fonction *fread*. Le dernier argument devra alors indiquer par une valeur *TRUE* la fin de ce document.

```
$donnee_XML = fread($id_fichier, 4096));
xml_parse($id_analyseur, $donnee_XML, feof($id_fichier)));
```

L'analyse des informations XML provoque l'**appel des gestionnaires d'événements adaptés à chaque fois qu'est rencontré un composant XML** comme un élément, du texte ou encore une instruction de traitement.

Auparavant, il est nécessaire de **notifier aux gestionnaires d'événement des fonctions destinées à exécuter des tâches précises**.

```
function gestionnaire_debut($analyseur,
                            $nom_element, $tab_attributs)
{ # instructions... }

function gestionnaire_fin($analyseur, $nom_element)
{ # instructions... }

xml_set_element_handler($id_analyseur,
                        "gestionnaire_debut", "gestionnaire_fin");
```

Seul, **le gestionnaire d'événement relatif aux éléments XML fait appel à deux fonctions.** Ces fonctions se lancent respectivement lorsque l'analyseur rencontre une balise ouvrante ou une balise de fermeture.

```
<balise attribut="valeur">
    ==> entraine l'appel de la fonction gestionnaire_debut.
  Valeur textuelle
      ==> entraine l'appel de la fonction gestionnaire_texte.
</balise>
    ==> entraine l'appel de la fonction gestionnaire_fin.
```

La fonction dénommée *gestionnaire_texte* pourra être sollicitée si le gestionnaire

d'événement *xml_set_character_data_handler* est lui-même appelé dans le script.

```
function gestionnaire_texte($analyseur, $nom_element)
{ # instructions... }

xml_set_character_data_handler($id_analyseur,
                               "gestionnaire_texte");
```

Il existe **sept gestionnaires d'événements chargés d'activer des fonctions** lorsque l'analyseur XML trouve des éléments, des données textuelles, des instructions de traitement, des entités non analysées, des références d'entités, des notations, etc..

La fonction *gestionnaire_debut* permet de récupérer d'une part **le nom de l'élément rencontré** pour les stocker dans une chaîne de caractères (second argument) et d'autre part **le nom des attributs et leur valeur**, lesquels sont placés dans un tableau associatif (troisième argument).

```
function gestionnaire_debut($id_analyseur,
                            $nom_element, $tab_attributs)
{
  echo $nom_element . "<br>";
  foreach($tab_attributs as $cle => $valeur)
  {
    echo "  * " . $cle . " = " . $valeur . "<br>";
  }
}
```

Les données textuelles comprises au sein des éléments sont obtenues par le second argument de la fonction *gestionnaire_texte*.

```
function gestionnaire_texte($id_analyseur, $texte)
{
  echo $texte . "<br>";
}
```

Tous les caractères, y compris les espaces blancs tels que les tabulations et les retours de lignes, sont analysés et retournés dans l'argument *$texte*.

Les informations relatives aux erreurs XML peuvent être reueillies par l'intermédiaire de fonctions spéciales *xml_get_error_code*, *xml_error_string*, *xml_get_current_line_number* et *xml_get_current_column_number* retournant respectivement son numéro, son libellé et le numéro de la ligne et de la colonne où elle s'est produite dans le document XML.

```
sprintf("Erreur XML n°%d "
              . "Message : %s "
              . "Ligne  : %d "
              . "Colonne : %d",
      xml_get_error_code($analyseur_xml),
      xml_error_string(xml_get_error_code($analyseur_xml)),
      xml_get_current_line_number($analyseur_xml),
      xml_get_current_column_number($analyseur_xml)
);
```

Enfin, **la libération des ressources consommées par l'analyseur est réalisée par la fonction *xml_parser_free***.

```
xml_parser_free($id_analyseur);
```

Exemple [voir]

```php
<?php
 function gestionnaire_debut($analyseur, $nom, $attribut)
 {
  global $nb;
  print "<b><u>Elément (niveau $nb) :</u></b> ";
  for ($i = 0; $i < $nb; $i++)
  {
   print "    ";
  }
  print "<b>" . $nom . "</b>";
  $nb++;
  if(sizeof($attribut))
  {
   foreach($attribut as $cle => $valeur)
   {
    echo ' (' . $cle . ' : ' . $valeur . ')<br>';
   }
  }
  else echo "<br>";
 }

 function gestionnaire_fin($analyseur, $nom)
 {
  global $nb;
  $nb--;
 }

 function gestionnaire_texte($analyseur, $texte)
 {
  global $nb;
  print "Texte (niveau $nb) :";
  for ($i = 0; $i < $nb; $i++)
  {
   print "      ";
  }
  print $texte . "<br>";
 }

 $fichier_xml = "recueil.xml";
 $nb = 0;

 $analyseur_xml = xml_parser_create();

 xml_set_element_handler($analyseur_xml,
            "gestionnaire_debut", "gestionnaire_fin");
 xml_set_character_data_handler($analyseur_xml,
                            "gestionnaire_texte");

 if (!($id_fichier = fopen($fichier_xml, "r")))
 {
  die("Impossible d'ouvrir le fichier XML !");
 }

 while ($donnee = fread($id_fichier, filesize($fichier_xml)))
 {
  if (!xml_parse($analyseur_xml, $donnee, feof($id_fichier)))
  {
   die(sprintf("Une erreur XML %s s'est produite "
               . "à la ligne %d et à la colonne %d.",
            xml_error_string(xml_get_error_code($analyseur_xml)),
            xml_get_current_line_number($analyseur_xml),
            xml_get_current_column_number($analyseur_xml)));
  }
 }
 xml_parser_free($analyseur_xml);
?>
```

27.1.1 / Les erreurs XML

L'analyseur XML retourne certaines constantes suite à une erreur dans le traitement des données XML.

Constante d'erreur
Description
XML_ERROR_NONE
représente une erreur nulle.
XML_ERROR_NO_MEMORY
représente une erreur de manque de ressource.
XML_ERROR_SYNTAX
représente une erreur de syntaxe XML.
XML_ERROR_NO_ELEMENTS
représente une erreur due à une carence d'éléments.
XML_ERROR_INVALID_TOKEN
représente une erreur due à une entrée d'octet ne pouvant être correctement assignée à un caractère.
XML_ERROR_UNCLOSED_TOKEN
représente une erreur due à une balise non fermée.
XML_ERROR_PARTIAL_CHAR
représente une erreur due à un caractère partiel.
XML_ERROR_TAG_MISMATCH
représente une erreur due à un balisage disparate.
XML_ERROR_DUPLICATE_ATTRIBUTE
représente une erreur due à un double d'un attribut dans un élément.
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
représente une erreur due à des déchets après l'élément document comme des caractères non autorisés. Seuls les espaces blancs sont permis dans cette zone.
XML_ERROR_PARAM_ENTITY_REF
représente une erreur due à une référence d'entité paramètre trouvée à un endroit non autorisé.
XML_ERROR_UNDEFINED_ENTITY
représente une erreur due à une entité non définie.
XML_ERROR_RECURSIVE_ENTITY_REF
représente une erreur due à une référence d'entité récursive.
XML_ERROR_ASYNC_ENTITY
représente une erreur due à une entité asynchrone.
XML_ERROR_BAD_CHAR_REF
représente une erreur due à une mauvaise référence de caractères.
XML_ERROR_BINARY_ENTITY_REF
représente une erreur due à une référence d'entité se référant à une entité déclarée par une notation.
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
représente une erreur due à une mauvaise attribution d'une référence d'entité externe.
XML_ERROR_MISPLACED_XML_PI
représente une erreur due à une instruction de traitement placée ailleurs qu'au début du document.

XML_ERROR_UNKNOWN_ENCODING
représente une erreur due à un encodage inconnu.
XML_ERROR_INCORRECT_ENCODING
représente une erreur due à un encodage incorrect.
XML_ERROR_UNCLOSED_CDATA_SECTION
représente une erreur due à une section CDATA non fermée.
XML_ERROR_EXTERNAL_ENTITY_HANDLING
représente une erreur due à un traitement d'entité externe.

27.1.2 / Les fonctions d'analyseur XML

Le langage PHP dispose de nombreuses fonctions permettant de travailler avec les analyseurs XML.

Fonction
Description
$chaine_ISO-8859 = **utf8_decode**($donnee_UTF-8);
convertit une chaîne UTF-8 en ISO-8859.
$chaine_UTF-8**utf8_encode**($donnee_ISO-8859);
convertit une chaîne ISO-8859-1 en UTF-8.
$chaine = **xml_error_string**($nombre);
retourne le message d'erreur de l'analyseur XML par l'intermédiaire du nombre entier provenant de *xml_get_error_code*.
$index = **xml_get_current_byte_index**($reference);
retourne l'index de l'octet en cours d'un analyseur XML.
$nombre = **xml_get_current_column_number**($reference);
retourne le nombre de la colonne en cours d'un analyseur XML.
$nombre = **xml_get_current_line_number**($reference);
retourne le numéro de la ligne en cours d'un analyseur XML.
$nombre = **xml_get_error_code**($reference);
retourne le nombre courant de colonne d'un analyseur XML.
$nombre = **xml_parse**($reference, $donnee, true|false);
analyse un fichier XML par morceau dont le dernier devra être à *true*.
$nombre = **xml_parse_into_struct**($reference, $donnee, $tab_valeurs, $tab_index);
analyse des données XML et les placent dans les tableaux spécifiés.
$reference = **xml_parser_create**([$encodage]);
crée un analyseur XML et retourne une référence. L'encodage peut être : ISO-8859-1 (par défaut), US-ASCII, UTF-8 ou UTF-16.
$reference = **xml_parser_create_ns**([$encodage[, $sep]]);
crée un analyseur XML avec un espace de noms. L'agument *sep* est un caractère faisant parti de l'espace de noms concerné.
true | false = **xml_parser_free**($reference);
détruit un analyseur XML.
$valeur = **xml_parser_get_option**($reference, $option);
retourne la valeur des options d'un analyseur XML.
$nombre= **xml_parser_set_option**($referene, $option, valeur);
modifie les options d'un analyseur XML. Les options peuvent être *XML_OPTION_CASE_FOLDING* contrôlant la gestion de la casse des balises ou *XML_OPTION_TARGET_ENCODING* modifiant l'encodage (ISO-8859-1, US-ASCII ou UTF-8) à la cible utilisé par cet analyseur XML. Respectivement, le dernier argument sera soit un nombre entier, soit une chaîne de caractères.
xml_set_default_handler($reference, gestionnaire);
affecte le gestionnaire par défaut.

$nombre = **xml_set_character_data_handler**($reference, gestionnaire);
affecte les gestionnaires de caractère bruts.
$nombre = **xml_set_element_handler**($reference, gestionnaire_départ, gestionnaire_fin);
affecte les gestionnaires de début et de fin.
$nombre = **xml_set_notation_decl_handler**($reference, gestionnaire);
affecte les gestionnaires de notation.
xml_set_processing_instruction_handler($reference, gestionnaire);
affecte les gestionnaires d'instructions exécutables.
xml_set_end_namespace_decl_handler($reference, gestionnaire);
affecte le gestionnaire de fin d'espace de noms.
xml_set_start_namespace_decl_handler($reference, gestionnaire);
affecte le gestionnaire de démarrage d'espace de noms.
$nombre = **xml_set_external_entity_ref_handler**($reference, gestionnaire);
modifie le gestionnaire de référence externes.
xml_set_unparsed_entity_decl_handler($reference, gestionnaire);
affecte les gestionnaires d'entité non déclaré.
xml_set_object($reference, &$objet);
utilise un analyseur XML à l'intérieur d'un objet.

27.2 / Le DOM de PHP 5

L'extension DOM de PHP 5 permet de manipuler des documents XML avec une collection d'objets et leurs méthodes et propriétés associées.

L'extension DOM de la version 5 de PHP respecte assez fidèlement les spécifications XML (Document Object Model Level 2) du W3C.

Les interfaces fondamentales et étendues proposées par le W3C sont toutes présentes dans le modéle objet de l'extension DOM.

Objet PHP	Interface W3C	Description
DOMDocument	Document	représente un document XML.
DOMElement	Element	représente un élément XML.
DOMNode	Node	représente un noeud XML.
DOMAttr	Attr	représente un attribut XML.
DOMCharacterData	CharacterData	représente une section de caractères.
DOMText	Text	représente le texte d'un noeud XML.
DOMComment	Comment	représente un commentaire XML.
DOMNodeList	NodeList	représente une liste de noeuds.
DOMNamedNodeMap	NamedNodeMap	représente une collection d'attributs ordonnés par paire nom/valeur.
DOMDocumentFragment	DocumentFragment	représente une portion d'un document XML.
DOMImplementation	DOMImplementation	représente l'implémentation utilisé pour exploiter un document XML.
DOMException	DOMException	représente une exception du DOM qui pourra être lancée lors de l'analyse ou l'exploitation du document XML.
	ExceptionCode	constitue une liste de constantes représentant chacune un type d'exception.
DOMCDATASection	CDATASection	représente une section de caractères non-analysables (section CDATA)
DOMProcessingInstruction	ProcessingInstruction	représente une instruction de traitement.
DOMDocumentType	DocumentType	représente une définition de type de document.
DOMNotation	Notation	représente une notation XML.
DOMEntity	Entity	représente une entité.
DOMEntityReference	EntityReference	représente une référence d'entité.

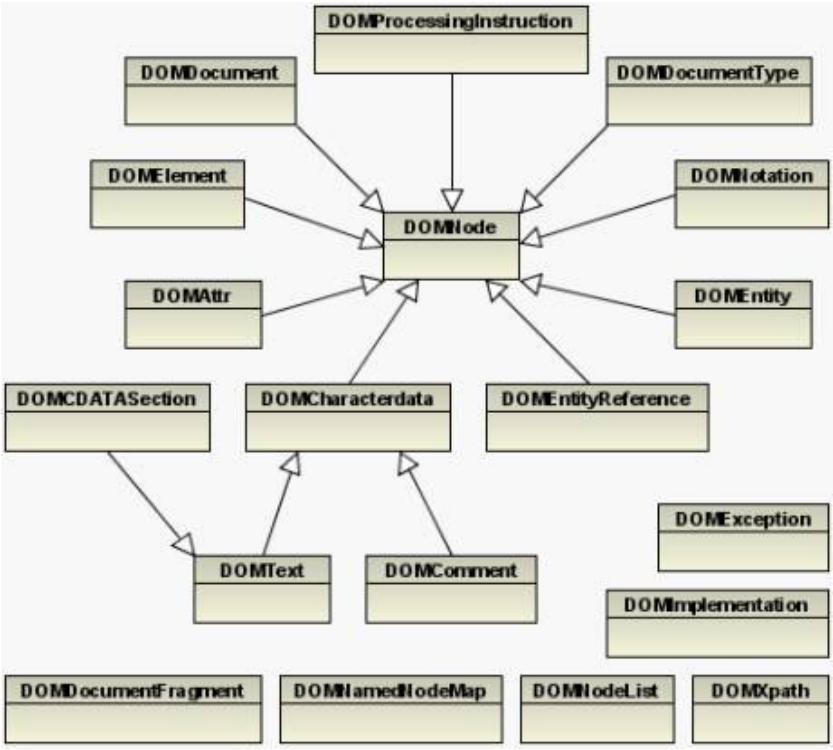
Chaque objet du modèle d'objet de document de PHP 5, possède un constructeur, des méthodes et des attributs.

```
//Instanciation d'un objet DOMDocument
$doc_xml = new DOMDocument();
//Modification de l'attribut de validation du document par sa DTD
$doc_xml->validateOnParse = true;
//Invocation de la méthode de chargement d'un fichier XML
$doc_xml->load('source.xml');
```

La classe *DOMNode* est héritée par de nombreuses autres classe du DOM. En effet, *DOMDocument*, *DOMElement*, *DOMAttr*, *DOMCharacterData*, *DOMDocumentType*, *DOMEntity*, *DOMEntityReference*, *DOMNotation* et *DOMProcessingInstruction* sont des classes dérivées de

DOMNode. Cela signifie qu'**elles héritent toutes des méthodes et attributs de la classe *DOMNode***.

Les classes *DOMComment* et *DOMText* sont des sous classes de *DOMCharacterData*. Ainsi, les deux sous-classes héritent des méthodes et attributs de la classe *DOMCharacterData* et par le truchement de cette dernière de ceux de *DOMNode* également.



27.2.1 / Utiliser le DOM

L'utilisation du DOM de PHP 5 n'est pas particulièrement compliqué mais diffère des versions précédentes.

La création d'un document XML est le préalable obligatoire à l'exploitation du modèle d'objet.

```
$doc = new DOMDocument();
```

Le numéro de version XML et l'encodage de caractères peuvent être précisés lors de l'instanciation de la classe *DOMDocument*.

```
$doc = new DOMDocument('1.0', 'ISO-8859-1');
echo $doc->saveXML(); //Affiche le prologue
```

Désormais, **l'objet *DOMDocument*** est prêt à :

- soit être **construit de toutes pièces** avec des éléments, attributs et textes,
- soit être **chargé à partir d'une source XML** provenant d'un fichier ou d'une variable textuelle.

```
$doc->load('fichier.xml');
//ou
$xml = '<racine><element>texte</element></racine>';
$doc->loadXML($xml);
```

Un document HTML peut être chargé de la même façon avec les méthodes *loadHTMLFile()* et *loadHTML()*.

L'exploration du document XML s'effectue en se déplaçant dans l'arborescence, de noeud en noeud, à l'aide des nombreuses méthodes et attributs proposées par le DOM.

Le premier noeud est obtenu par l'attribut *documentElement* de l'objet *DOMDocument*.

```
$racine = $doc->documentElement;
```

L'élément racine étant un descendant (héritage) d'un noeud, les méthodes et attributs de l'objet *DOMNode* sont utilisables pour récupérer des informations sur le noeud et sa descendance.

```
$nom = $racine->nodeName;
$valeur = $racine->nodeValue;
$type = $racine->nodeType;
$pere = $racine->nodeParent;
$liste_enfants = $racine->childNodes;
$attributs = $racine->attributes;
$document = $racine->ownerDocument;
$texte = $racine->textContent;
```

D'autres propriétés permettent d'accéder aux noeuds environnants.

```
$noeud = $liste_enfants->item(3);
$premier_enfant = $noeud->firstChild;
$dernier_enfant = $noeud->lastChild;
$grand_frere = $noeud->previousSibling;
$petit_frere = $noeud->nextSibling;
```

L'attribut *childNodes* est particulièrement utile pour parcourir rapidement et facilement une arborescence XML. Il suffit d'employer une boucle pour accéder à chaque enfant de la liste de noeuds obtenus.

```
$doc_xml = new DOMDocument();
$doc_xml->loadXML($source);
$racine = $doc_xml->documentElement;
echo '<h1>ELEMENT RACINE</h1>';
afficherInfos($racine);
$liste = $racine->childNodes;

echo '<h2>ENFANTS DE L\'ELEMENT RACINE</h2>';
```

```
foreach($liste as $noeud){
  afficherInfos($noeud);
}

function afficherInfos($noeud){
  $nom = $noeud->nodeName;
  $valeur = $noeud->nodeValue;
  $type = $noeud->nodeType;
  echo '<b>' . $nom . '</b> (' . $type . ') = ' . $valeur . '
';
}
```

Une boucle *for* pourrait également permettre de parcourir la liste en faisant appel à la méthode *item()* de l'objet *DOMNodeList*.

```
for($i = 0; $i < $liste.length; $i++){
  afficherInfos($liste.item($i));
}
```

Le parcours complet de l'arborescence demande une fonction récursive afin que chaque élément rencontré puisse être exploré à son tour, s'il possède des noeuds enfants (*hasChildNodes()*).

```
$doc_xml = new DOMDocument();
$doc_xml->loadXML($source);
$racine = $doc_xml->documentElement;
echo '<h1>ARBORESCENCE DE L\'ELEMENT RACINE</h1>';
parcourir($racine);

function afficherInfos($noeud){
  $nom = $noeud->nodeName;
  $valeur = $noeud->nodeValue;
  $type = $noeud->nodeType;
  echo $nom . '(' . $type . ') = "' . $valeur . '"<br/>';
}
function parcourir($noeud, $p = 0){
  if($p > 0) indenter($p);
  afficherInfos($noeud);
  if($noeud->nodeType == XML_ELEMENT_NODE
  && $noeud->hasChildNodes()){
    $p++;
    $enfants = $noeud->childNodes;
    foreach($enfants as $enfant){
      parcourir($enfant, $p);
    }
  }
}
function indenter($n){
  $tab = '    ';
  for($i = 0; $i < $n; $i++){
    echo $tab;
  }
  echo ' - ';
}
```

Les attributs peuvent être récupérés de la même façon. **Seuls les éléments XML contiennent des attributs.** Fort de cette constatation, il suffit de tester si le noeud rencontré est un élément et qu'il possède des attributs (*hasAttributes()*). Puis avec l'attribut *attributes* de l'objet *DOMNode*, il faut récupérer la collection *DOMNamedNodeMap* et la parcourir.

```
function parcourir($noeud, $p = 0){
  if($p > 0) indenter($p);
  afficherInfos($noeud);
  if($noeud->nodeType == XML_ELEMENT_NODE
  && $noeud->hasChildNodes()){
    if($noeud->hasAttributes()){
      $attributs = $noeud->attributes;
      indenter($p + 3);
      echo '[ ';
      foreach($attributs as $nom=>$valeur){
        echo '<b>'. $nom . '</b>="' . $valeur->value . '" ';
      }
      echo ']<br/>';
```

```
    }
    $p++;
    $enfants = $noeud->childNodes;
    foreach($enfants as $enfant){
      parcourir($enfant, $p);
    }
  }
}
```

Le noeud *$valeur* est un objet *DOMAttr*, c'est pourquoi il est nécessaire d'appeler l'attribut *value* pour obtenir la valeur du noeud.

Par ailleurs, plusieurs méthodes de l'objet *DOMElement* permettent de récupérer un attribut ou sa valeur, à l'aide de son nom et éventuellement son espace de noms. Il s'agît des méthodes *getAttribute()* et *getAttributeNS()* ou *getAttributeNode()* et *getAttributeNodeNS()* qui retournent respectivement la valeur de l'attribut ou un objet *DOMAttr*.

```
$noeuds = $doc_xml->getElementsByTagName('nom');
foreach($noeuds as $noeud){
  $langue = $noeud->getAttributeNode('langue');
  afficherInfos($langue);
  $nom = 'systeme_exploitation';
  $os = $noeud->getAttribute($nom);
  echo '<b>' . $nom .'</b> = "' . $os . '"';
}
```

L'objet *DOMXpath* et une expression XPath soumis à cet objet, fournit un moyen efficace pour acccéder directement à un noeud précis dans l'arborescence d'un document XML.

```
$doc_xml = new DOMDocument();
$doc_xml->loadXML($source);
$racine = $doc_xml->documentElement;
$xpath = new DOMXPath($doc_xml);
$noeuds = $xpath->query('//logiciel[1]/nom', $racine);
$noeud = $noeuds->item(0);
echo $noeud->nodeName . '="' . $noeud->nodeValue . '"';
```

Le second argument de la méthode *query()* détermine un noeud contextuel à partir duquel s'appliquera l'expression XPath spécifiée. Par défaut, l'expression XPath s'applique à la racine du document XML.

```
$logiciels = $doc_xml->getElementsByTagName('logiciel');
$xpath = new DOMXPath($doc_xml);
foreach($logiciels as $logiciel){
  $noeuds = $xpath->query('nom', $logiciel);
  $noeud = $noeuds->item(0);
  echo $noeud->nodeName . '="' . $noeud->nodeValue . '"';
}
```

La méthode *getElementsByTagName()* des objets *DOMDocument* et *DOMElement* retourne une liste de noeuds répondant au nom passé en argument.

```
$logiciels = $doc_xml->getElementsByTagName('logiciel');
foreach($logiciels as $logiciel){
  $nom = $logiciel->getElementsByTagName('nom');
  $commentaire = $logiciel->getElementsByTagName('commentaire');
  $editeur = $logiciel->getElementsByTagName('editeur');
  $prix = $logiciel->getElementsByTagName('prix');
}
```

Si le document possède des attributs ID permettant d'identifier sans ambiguités des éléments, il est possible d'**employer la méthode *getElementById()*** afin de récupérer l'élément correspondant à un identificateur passé en argument. Toutefois, **le document XML doit être validé** par une DTD, un schéma XML ou un schéma Relax NG.

```
$doc_xml = new DOMDocument();
$doc_xml->loadXML($source);
$doc_xml->validateOnParse = true;
$logiciel = $doc_xml->getElementById('XSY325684');
afficherInfos($logiciel);
```

Si vous supprimez la déclaration *DOCTYPE* de la source XML, la méthode *getElementById()*

deviendra alors inopérante.

Lors du parcours des attributs d'un élément XML, la méthode *DOMAttr->isId()* permet de déterminer si l'attribut en cours est ou n'est pas un attribut identificateur. A l'instar de la méthode *getElementById()*, le document doit être validé.

```
$noeuds = $element->doc->getElementsByTagName('logiciel');
foreach($noeuds as $noeud){
  $id = $noeud->getAttributeNode('id');
  if($id->isId()){
    echo 'L'attribut est bien un identificateur !';
  }
}

<?php //Fichier contenant la source XML
$source = <<<XML
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE liste [
<!ELEMENT commentaire (#PCDATA)>
<!ELEMENT editeur (#PCDATA)>
<!ATTLIST editeur adresse CDATA #REQUIRED>
<!ELEMENT liste (logiciel+)>
<!ELEMENT logiciel (nom, commentaire, editeur, prix)>
<!ATTLIST logiciel id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ATTLIST nom
     langue CDATA #REQUIRED
     systeme_exploitation CDATA #REQUIRED>
<!ELEMENT prix (#PCDATA)>
<!ATTLIST prix monnaie CDATA #REQUIRED>
]>
<liste>
  <logiciel id="CTP0124555">
    <nom langue="US" systeme_exploitation="Win">
      Cooktop 2.200
    </nom>
    <commentaire>
      Un editeur XML, XSLT, XPath et DTD puissant et totalement gratuit.
    </commentaire>
    <editeur adresse="http://xmleverywhere.com/cooktop/">
      XML Everywhere
    </editeur>
    <prix monnaie="$US">00.00</prix>
  </logiciel>
  <logiciel id="XSY325684">
    <nom langue="US" systeme_exploitation="Win">
      XML Spy 4.1
    </nom>
    <commentaire>
      Un editeur XML desormais mature.
    </commentaire>
    <editeur adresse="http://www.xmlspy.com/default.html">
      Altova Inc.
    </editeur>
    <prix monnaie="$US">199,00</prix>
  </logiciel>
  <logiciel id="XSY210356">
    <nom langue="US" systeme_exploitation="Win">
      XML Spy 4.1 B2B Server
    </nom>
    <commentaire>
      La version 4 en version Business to business.
    </commentaire>
    <editeur adresse="http://www.xmlspy.com/default.html">
      Altova Inc.
    </editeur>
    <prix monnaie="$US">1 999,00</prix>
  </logiciel>
  <logiciel id="XWR387795">
    <nom langue="US" systeme_exploitation="Win">
      XMLwriter v1.21
    </nom>
    <commentaire>
      Permet de creer des documents XML.
```

```
    </commentaire>
    <editeur adresse="http://xmlwriter.net/">
      Wattle Software
    </editeur>
    <prix monnaie="$US">75,00</prix>
  </logiciel>
</liste>
XML;
?>
```

27.2.2 / Créer des noeuds

La création ou la modification d'un document XML existant s'accomplit aisément avec les outils proposés par le DOM.

La création d'objets du DOM s'obtient soit par l'instanciation des objets, soit par les méthodes de création de l'objet *DOMDocument*.

```
$attribut = $doc_xml->createAttribute('unAttribut');
$attr_ns = $doc_xml->createAttributeNS(
                'http://www.laltruiste.com/', 'lal:unAttribut');
$cdata =  $doc_xml->createCDATASection('Un texte non analysable...');
$commentaire = $doc_xml->createComment('Un commentaire...');
$element = $doc_xml->createElement('unElement', 'Une valeur...');
$elt_ns = $doc_xml->createElementNS(
                'http://www.laltruiste.com/', 'lal:unElement', 'Une valeur...');
$entite = $doc_xml->createEntityReference('&amp;');
$pi = $doc_xml->createProcessingInstruction(
                'xml-stylesheet', 'href="style.xsl" type="text/xsl"');
$texte = $doc_xml->createTextNode('Un texte...');

$attribut = new DOMAttr('unAttribut', 'Une valeur');
$attr_ns = new DOMAttr('unAttribut', 'Une valeur');
$attr_ns->namespaceURI = 'http://www.laltruiste.com/';
$attr_ns->prefix = 'lal';
$cdata =  new DOMCDATASection('Un texte non analysable...');
$commentaire = new DOMComment('Un commentaire...');
$element = new DOMElement('unElement', 'Une valeur...');
$elt_ns = new DOMElement(
          'lal:unElement',
          'Une valeur...',
          'http://www.laltruiste.com/');
$entite = new DOMEntity('&á');
$pi = new DOMProcessingInstruction(
                'xml-stylesheet', 'href="style.xsl" type="text/xsl"');
$texte = new DOMText('Un texte...');
```

Désormais, il faut pouvoir insérer ces objets au sein d'un document XML.

Les noeuds éléments ou textuels se greffent dans un document par l'intermédiaire de la méthode *appendChild()* de l'objet *DOMNode*.

```
$doc_xml = new DOMDocument('1.0', 'ISO-8859-1');
$racine = new DOMElement('racine');
$doc_xml->appendChild($racine);
echo $doc_xml->saveXML();
/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <racine/> */
```

L'ajout d'un élément ou d'un texte à un élément parent est identique. **Le noeud cible de l'ajout doit invoquer la méthode *appendChild()* pour qu'un nouveau noeud soit ajouter en son sein et à la fin de ses enfants éventuels.**

```
$enfant = new DOMElement('enfant');
$racine->appenChild($enfant);
$cdata = new DOMCDATASection('<balise/> -> Balise vide');
$noeud_ajoute = $enfant->appendChild($cdata);
echo $doc_xml->saveXML();

/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <racine> <enfant>
<![CDATA[<balise/> -> Balise vide]]> </enfant> </racine> */
```

La méthode *appendChild()* retourne le noeud nouvellement ajouté. En outre, **cette méthode est susceptible de lancer des exceptions**.

- **DOM_NO_MODIFICATION_ALLOWED_ERR** est lancée si le noeud est en lecture seul ou si le parent précédent le noeud à ajouter est en lecture seul.
- **DOM_HIERARCHY_REQUEST_ERR**est lancée si le noeud est d'un type qui n'autorise pas d'enfant du type du nouveau noeud, ou si le noeud à ajouter est un des noeuds ancêtres ou ce noeud lui-même.

- **DOM_WRONG_DOCUMENT_ERR** est lancée si le nouveau noeud a été créé à partir d'un document différent que celui qui a créé ce noeud.

Les attributs s'ajoutent aux éléments d'une manière différente. En effet, il faut passer **par les méthodes d'ajout d'attribut de l'objet *DOMElement***.

```
$element = new DOMElement('elt');
$reussi = $element->setAttribute('attribut', 'valeur');
$attribut = new DOMAttr('attribut', 'valeur');
$reussi = $element->setAttributeNode($attribut);
$attr_ns = $element->setAttributeNS(
                'http://www.laltruiste.com/',
                'lal:attr_ns',
                'Valeur de l\'attribut');

$attribut_ns = new DOMAttr('lal:attribut_ns');
$attribut_ns->nodeValue = 'Une valeur d\'attribut';
$nouv_attr = $element->setAttributeNodeNS($attribut_ns);
if($attribut_ns->isSameNode($attribut_ns))
  echo 'Le noeud a bien été ajouté puisqu\'il sont identiques !';
echo $doc_xml->saveXML();

/*Affiche : Le noeud a bien été ajouté puisqu'il sont identiques ! <?xml version="1.0"
encoding="ISO-8859-1"?> <racine> <enfant> <![CDATA[<balise/> -> Balise vide]]> <elt
xmlns:lal="../" attribut="valeur" lal:attr_ns="Valeur de l'attribut" lal:attribut_ns="Une valeur
d'attribut"/> </enfant> </racine> */
```

L'ajout ou la modification d'une valeur textuelle à un attribut s'effectue par l'intermédiaire de la propriété *nodeValue*.

Le clonage des noeuds se réalise avec la méthode *cloneNode()* de l'objet *DOMNode*.

```
$clone = $enfant->clonNode(true);
$racine->appendChild($clone);
echo $doc_xml->saveXML();

/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <racine> <enfant>
<![CDATA[<balise/> -> Balise vide]]> <elt xmlns:lal="../" attribut="valeur" lal:attr_ns="Valeur
de l'attribut" lal:attribut_ns="Une valeur d'attribut"/> </enfant> <enfant> <![CDATA[<balise/>
-> Balise vide]]> <elt xmlns:lal="../" attribut="valeur" lal:attr_ns="Valeur de l'attribut"
lal:attribut_ns="Une valeur d'attribut"/> </enfant> </racine> */
```

Si ce genre d'ajout avait été tenté sans clonage :

- soit la commande aurait été inopérante puisque le noeud a déjà été ajouté,
- soit, dans le cas d'une insertion dans un autre noeud que le parent du noeud courant, une exception *Hierarchy Request Error* aurait été levée puisque le noeud était déjà utilisé ailleurs.

La méthode *importNode()* de l'objet *DOMDocument* fournit un moyen d'importation d'un noeud avec éventuellement toute sa descendance au sein d'un autre document que le propriétaire du noeud importé.

```
$nouv_doc = new DOMDocument('1.0', 'ISO-8859-1');
$import = $nouv_doc->importNode($enfant, true);
$nouv_doc->appendChild($import);
echo $nouv_doc->saveXML();

/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <enfant> <![CDATA[<balise/> ->
Balise vide]]> <elt xmlns:lal="../" attribut="valeur" lal:attr_ns="Valeur de l'attribut"
lal:attribut_ns="Une valeur d'attribut"/> </enfant> */
```

Les références d'entité peuvent être ajoutées directement dans un élément avec la méthode *appendChild()*.

```
$doc_xml = new DOMDocument('1.0', 'ISO-8859-1');
$element = $doc_xml->createElement('zoneTexte');
$ref = $doc_xml->createEntityReference('eacute');
$texte1 = $doc_xml->createtextNode('La r');
$texte2 = $doc_xml->createtextNode('f');
$texte3 = $doc_xml->createtextNode('rence d\'entit');
$elt->appendChild($texte1);
```

```
$element->appendChild($ref);
$element->appendChild($texte2);
$element->appendChild($ref->cloneNode());
$element->appendChild($texte3);
$element->appendChild($ref->cloneNode());
$doc_xml->appendChild($element);
echo $doc_xml->saveXML();

/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <zoneTexte> La référence d'entité
</zoneTexte> */
```

Les chaînes de caractères peuvent être ajoutées ou insérées dans un objet textuel à l'aide des méthodes *appendData()* et *insertData()*, voire même *replaceData()* de l'objet *DOMCharacterData*.

```
$doc_xml = new DOMDocument('1.0', 'ISO-8859-1');
$element = $doc_xml->createElement('citation');
$texte = new DOMCDATASection('On n\'est jamais si hereu');
$texte->appendData('ni si malheureux qu\'on s\'imagine.');
$texte->appendData('La Rochefoucauld, Maximes.');
$texte->replaceData(19, 5, 'heureux');
$texte->insertData(26, ' ');
$texte->insertData(60, '<br/>(');
$texte->insertData(0, '<p>');
$texte->appendData(')</p>');
$element->appendChild($texte);
$doc_xml->appendChild($element);
echo $doc_xml->saveXML();

/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <citation> <![CDATA[ <p>On n'est
jamais si heureux ni si malheureux qu'on s'imagine. </br>(La Rochefoucauld,
Maximes.)</p>]]> </citation> */
```

27.2.3 / Modifier les noeuds

Un document XML peut nécessiter une mise à jour de sa structure et de ses données. **Le DOM dispose de méthodes capables de remplacer et de supprimer des noeuds et de modifier des données textuelles.**

La méthode *removeChild()* de l'objet *DOMNode* supprime le noeud spécifié de la liste des enfants.

```
$doc_xml = new DOMDocument();
$doc_xml->validateOnParse = true;
$doc_xml->loadXML($source); //voir La création
$element = $doc_xml->getElementById('XSY210356');
$racine = $doc_xml->documentElement;
$noeud_sup = $racine->removeChild($element);

echo $doc_xml->saveXML();
/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE liste [...]> <liste>
<logiciel id="CTP0124555"> ... </logiciel> <logiciel id="XSY325684"> ... </logiciel> <logiciel
id="XWR387795"> ... </logiciel> </liste>*/
```

De cette manière n'importe quel noeud dans l'arborescence XML peut être supprimé. Pour un noeud textuel, il suffit de cibler l'objet contenant le texte, puis de soumettre l'enfant de ce dernier à la méthode *removeChild()*.

```
$commentaires = $doc_xml->getElementsByTagName('commentaire');
foreach($commentaires as $commentaire){
  $texte = $commentaire->childNodes->item(0);
  $commentaire->removeChild($texte);
}

/*Tous les éléments commentaire deviennent des éléments vides <commentaire/> */
```

Après suppression un noeud *DOMElement* peut devenir un élément vide.

```
for($i = $noeuds->length - 1; $i >= 0; $i--){
  $noeud = $noeuds->item($i);
  $noeud_sup = $element->removeChild($noeud);
  afficherInfos($noeud_sup);
}
/*L'arborescence de l'élément logiciel avec id="XSY210356" devient un élément vide
<logiciel id="XSY210356"/> */
```

La méthode *removeChild()* est susceptible de lancer deux exceptions :

- **DOM_NO_MODIFICATION_ALLOWED_ERR** est lancée si le noeud est en lecture seul.
- **DOM_NOT_FOUND** est lancés si le noeud à supprimer n'est pas un enfant de ce noeud.

Les attributs peuvent être supprimés d'un élément en utilisant les méthodes de suppression d'attributs de l'objet *DOMElement*.

```
$xpath = new DOMXPath($doc_xml);
$elements = $xpath->query(
                    '//logiciel[1]/nom',
                    $doc_xml->documentElement);

$element = $elements->item(0);
if($element->removeAttribute('systeme_exploitation'))
  echo 'L\'attribut systeme_exploitation a été supprimé !';

$attribut = $element->attributes->item(0);
if($element->removeAttributeNode($attribut))
  echo 'L\'attribut ' . $attribut->nodeName . ' a été supprimé !';

echo $doc_xml->saveXML();
/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE liste [...]> <liste>
<logiciel id="CTP0124555"> <nom>Cooktop 2.200</nom> ... </logiciel> <logiciel
id="XSY325684"> ... </logiciel> <logiciel id="XSY210356"> ... </logiciel> <logiciel
id="XWR387795"> ... </logiciel> </liste>*/
```

Une autre méthode de suppression existe, mais elle doit être employée pour un attribut situé dans un espace de noms. Il s'agît de la méthode *DOMElement->removeAttributeNS()*.

La méthode *replaceChild()* remplace un noeud par un autre.

```
xpath = new DOMXPath($doc_xml);
$elements = $xpath->query('//logiciel/commentaire');
$element1 = $elements->item(0);
$element2 = $elements->item($elements->length - 1);
$texte1 = $element1->firstChild->cloneNode();
$texte2 = $element2->firstChild->cloneNode();
$nouv_noeud1 = $element1->replaceChild(
                    $texte2,
                    $element1->firstChild);
$nouv_noeud2 = $element2->replaceChild(
                    $texte1,
                    $element2->firstChild);
echo $doc_xml->saveXML();
/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE liste [...]> <liste>
<logiciel id="CTP0124555"> ... <commentaire> Permet de creer des documents XML.
</commentaire> ... </logiciel> <logiciel id="XSY325684"> ... </logiciel> <logiciel
id="XSY210356"> ... </logiciel> <logiciel id="XWR387795"> ... <commentaire> Un editeur
XML, XSLT, XPath et DTD puissant et totalement gratuit. </commentaire> </logiciel> ...
</liste>*/
```

Cette méthode peut lancer plusieurs exceptions :

- **DOM_NO_MODIFICATION_ALLOWED_ERR** est lancée si le noeud est en lecture seule ou si le parent précédent du noeud à insérer est en lecture seule.
- **DOM_HIERARCHY_REQUEST_ERR** est lancée si le noeud est d'un type qui n'autorise pas les enfants du type du nouveau noeud, ou si le noeud à insérer est un des ancêtres de ce noeud ou ce noeud lui-même.
- **DOM_WRONG_DOCUMENT_ERR** est lancée si le nouveau noeud provient d'un document différent que celui qui l'a créé.
- **DOM_NOT_FOUND** est lancée si l'ancien noeud n'est pas un enfant de ce noeud.

Les chaînes de caractères présentes dans les noeuds textuels ou dans les attributs peuvent être modifiées complètement ou partiellement. La classe *DOMCharacterData* propose des méthodes de modification du contenu textuel d'un noeud.

```
$doc_xml->loadXML($source);
$logiciels = $doc_xml->getElementsByTagName('logiciel');
foreach($logiciels as $logiciel){
  $noms = $logiciel->getElementsByTagName('nom');
  $nom = $noms->item(0)->firstChild;
  $commentaires = $logiciel->getElementsByTagName('commentaire');
  $commentaire = $commentaires->item(0)->firstChild;
  //Insertion du nom devant le commentaire
  $commentaire->insertData(0, ' : ');
  $commentaire->insertData(0, $nom->nodeValue);
  $colprix = $logiciel->getElementsByTagName('prix');
  $prix = $colprix->item(0)->firstChild;
  $pos = strrpos($commentaire->nodeValue, '.');
  $taille = strlen($commentaire->nodeValue);
  //Suppression du point terminal et des espaces blancs
  $commentaire->deleteData($pos , $taille - $pos);
  //Ajout du prix entre parenthèses
  $commentaire->appendData('(');
  $commentaire->appendData($prix->nodeValue);
  $commentaire->appendData(').');
}

echo $doc_xml->saveXML();
/*Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE liste [...]> <liste>
<logiciel id="CTP0124555"> ... <commentaire> Cooktop 2.200 : Un editeur XML, XSLT,
XPath et DTD puissant et totalement gratuit (00.00). </commentaire> ... </logiciel> <logiciel
id="XSY325684"> ... <commentaire> XML Spy 4.1 : Un editeur XML desormais mature
(199,00). </commentaire> ... </logiciel> <logiciel id="XSY210356"> ... <commentaire> XML
Spy 4.1 B2B Server : La version 4 en version Business to business (1 999,00).
</commentaire> ... </logiciel> <logiciel id="XWR387795"> ... <commentaire> XMLwriter
v1.21 : Permet de creer des documents XML (75,00). </commentaire> ... </logiciel> </liste>
*/
```

```
foreach($logiciels as $logiciel){
  $noms = $logiciel->getElementsByTagName('nom');
  $nom = $noms->item(0)->firstChild;
  $commentaires = $logiciel->getElementsByTagName('commentaire');
  $commentaire = $commentaires->item(0)->firstChild;
  $pos = strpos($commentaire->nodeValue, ':');
  //Suppression du nom du logiciel
  $commentaire->deleteData(0, $pos + 1);
  $pos = strpos($commentaire->nodeValue, '(');
  $taille = strlen($commentaire->nodeValue);
  //Remplacement du prix entre parenthèses par un point terminal
  $commentaire->replaceData($pos, $taille - $pos, '.');
}

echo $doc_xml->saveXML();
//Retour à la source d'origine
```

Les méthodes *appendData()*, *deleteData()*, *insertData()*, *replaceData()* et *substringData()* s'appliquent à tous les noeuds héritant de la classe *DOMCharacterData*, en l'occurrence *DOMText*, *DOMComment* et *DOMCDataSection*.

27.2.4 / Sauvegarde d'un document

La sauvegarde d'un document XML permet de conserver les éventuelles modifications de valeurs ou de structure pratiquées sur ce document.

Il est possible soit d'enregistrer le document XML dans un fichier ou dans une chaîne de caractères.

Les méthodes *saveXML()* ou *saveHTML()* de l'objet *DOMDocument* permettent de récupérer un document sous la forme d'une chaîne de caractères.

```
$doc_xml = new DOMDocument('1.0', 'ISO-8859-1');
$doc_xml->formatOutput = true;

$racine = $doc_xml->createElement('racine');
$doc_xml->appendChild($racine);

$element = $doc_xml->createElement('element');
$racine->appendChild($element);

$element->setAttribute('attribut', 'valeur');

$texte = $doc_xml->createTextNode('Un texte quelconque...');
$element->appendChild($texte);

echo $doc_xml->saveXML();
/* Affiche : <?xml version="1.0" encoding="ISO-8859-1"?> <racine> <element
attribut="valeur">Un texte quelconque</element> </racine> */
```

L'attribut *formatOutput* indique que le document doit être correctement formaté.

La méthode *saveHTML()* ne peut que sauvegarder la totalité de l'arborescence interne du document. Tandis que la méthode *saveXML()* est capable de fournir une représentation textuelle d'un noeud particulier du document.

```
echo $doc_xml->saveXML($element);

/* Affiche : <element attribut="valeur">Un texte quelconque</element> */
```

Les méthodes *save()* et *saveHTMLFile()* enregistre un document au sein d'un fichier.

Les méthodes retournent le nombre d'octets écrits dans le fichier cible ou la valeur booléenne *false* en cas d'échec.

```
$source = 'source.html';
$doc_html = new DOMDocument();
$doc_html->loadHTMLFile($source);
$titres = $doc_html->getElementByTagName('title');
$titres->item(0)->nodeValue = "Un nouveau titre";

$nb_octets = $doc_html->saveHTMLFile($source);
if($nb_octets){
  echo 'Le fichier ' . $source . ' a bien été sauvegardé !';
}
else {
  echo 'Le fichier ' . $source . ' n'a pu être sauvegardé !';
}
```

La normalisation d'un document XML peut s'opérer au moyen de la méthode *normalize()*. Cette opération peut être utile pour stocker un document normalisé dans une représentation qui sera identique lors de sauvegarde et rechargement. Elle est également utile lorsque des opérations spécifiques, dépendant directement de la structure de l'arborescence spécifique à un document, doivent être pratiquées.

La méthode *normalize()* exprime le document sous une forme "normale". En fait, elle **met à jour les références d'entité** en fonction du dictionnaire d'entités de la DTD et **normalise des noeuds *Text***, c'est-à-dire place tous les noeuds *Text* dans toute la profondeur de la sous arborescence en dessous du noeud courant, dans une forme "normale" incluant les attributs, où seulement le balisage (éléments, commentaires, instructions de traitement, sections CDATA et

références d'entité) sépare les noeuds *Text*, c'est-à-dire qu'il ne doit pas y avoir de noeuds de type *Text* adjacents ou vides.

27.2.5 / La validation

L'extension DOM met à disposition plusieurs moyens de validation d'un document XML. Contrairement à ces prédécesseurs, PHP 5 introduit **la validation par des schémas W3C ou Relax NG** en plus de la validation traditionnelle par la définition de type du document (DTD).

```
//Inidique si le document doit être (true) //ou ne pas être (false) validé par sa DTD
$doc_xml->validateOnParse = true;
//Validation d'un document XML en fonction de sa DTD
$booleen = $doc_xml->validate();
//Valide un document XML en fonction //du fichier contenant le schéma XSD
$booleen = $doc_xml->schemaValidate('schema.xsd');
//Valide un document XML en fonction de //la chaîne de caractères contenant le schéma
XSD
$booleen = $doc_xml->schemaValidateSource($sourceXSD);
//Valide un document XML en fonction //du fichier contenant le schéma Relax NG
$booleen = $doc_xml->relaxNGValidate('schema.rng');
//Valide un document XML en fonction de //la chaîne de caractères contenant le schéma
XSD
$booleen = $doc_xml->relaxNGValidateSource($sourceRNG);
```

La validation par DTD nécessite que le document XML contienne au moins la déclaration *DOCTYPE* afin que ce document puisse être validé par la méthode *validate()* ou directement au chargement à l'aide de l'attribut *validateOnParse*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : employes.xml -->
<!DOCTYPE employes SYSTEM "definition.dtd">
<employes>
  <!-- ... -->
</employes>

<?php
  $doc_xml = new DOMDocument();
  $doc_xml->validateOnParse = true;
  $doc_xml->load('employes.xml');
  if($doc_xml->validate();)
    echo 'Le document XML est valide';
?>
```

La méthode *validate()* permet de revalider le document XML suite à une modification de son arborescence.

La validation par des schémas W3C et Relax NG demande simplement de fournir :

- soit le nom du fichier contenant le schéma XSD ou RNG,
- soit la source textuelle contenant le schéma XSD ou RNG.

```
<?php
  $doc_xml = new DOMDocument();
  $doc_xml->load('source.xml');

  if($doc_xml->validateSchema('schema.xsd');)
    echo 'Le document XML est valide par rapport au schéma W3C';

  if($doc_xml->relaxNGValidate('schema.rng');)
    echo 'Le document XML est valide par rapport au schéma Relax NG';
?>
```

Si le document XML comporte déjà une déclaration de type de document, cela ne pose aucun problème pour tenter des validations avec d'autres schémas.

```
<php
 $fichier = 'employes';
 $doc_xml = new DOMDocument();
 $doc_xml->validateOnParse = true;
 if($doc_xml->load($fichier . '.xml')){
  echo '<h3>Le document XML '
       . $employe .
       'n\'a été chargé !</h3>';
  if($doc_xml->validate())
   echo '<p>Le document XML est valide '
      . 'par rapport à la DTD</p>';

  if($doc_xml->schemaValidate($fichier . '.xsd'))
   echo '<p>Le document XML est valide '
      . 'par rapport au schéma W3C</p>';

  if($doc_xml->relaxNGValidate($fichier . '.rng'))
   echo '<p>Le document XML est valide '
      . 'par rapport au schéma Relax NG</p>';
 }
 else
  echo '<p style="color:red">Le document XML '
       . $employe .
       'n\'a pu être chargé !</p>';
?>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : employes.xml -->
<employes>
 <employe id="RJ1002" service="DG001">
  <nom>Robierre</nom>
  <prenom>Jean</prenom>
 </employe>
 <employe id="LA1012" service="DG001">
  <nom>Lardut</nom>
  <prenom>Anne</prenom>
 </employe>
 <employe id="GA1013" service="ST001">
  <nom>Guilde</nom>
  <prenom>Angélique</prenom>
 </employe>
 <employe id="HP1022" service="SC001">
  <nom>Henry</nom>
  <prenom>Paul</prenom>
 </employe>
 <employe id="MM1045" service="RH001">
  <nom>Mortier</nom>
  <prenom>Marc</prenom>
 </employe>
 <employe id="LS1102" service="SQ001">
  <nom>Lebreton</nom>
  <prenom>Sophie</prenom>
 </employe>
 <employe id="JM1095" service="RD001">
  <nom>Jolie</nom>
  <prenom>Martine</prenom>
 </employe>
 <employe id="MT1036" service="SC101">
  <nom>Marcelin</nom>
  <prenom>Tania</prenom>
 </employe>
 <employe id="LL1029" service="SC101">
  <nom>Léger</nom>
  <prenom>Laurence</prenom>
 </employe>
 <employe id="DM1052" service="SC001">
  <nom>Duroi</nom>
  <prenom>Maxime</prenom>
 </employe>
</employes>

<?xml version="1.0" encoding="UTF-8"?>
<!-- Fichier : definition.dtd -->
```

```
<!ELEMENT employe (nom, prenom)>
<!ATTLIST employe
        id ID #REQUIRED
        service NMTOKEN #REQUIRED
>
<!ELEMENT employes (employe+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : schema.xsd -->
<xsd:schema
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         elementFormDefault="qualified">
 <xsd:element name="employe">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="nom"/>
    <xsd:element ref="prenom"/>
   </xsd:sequence>
   <xsd:attribute name="id" type="xsd:ID" use="required"/>
   <xsd:attribute name="service" type="xsd:NMTOKEN" use="required"/>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="employes">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="employe" maxOccurs="unbounded"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="nom" type="xsd:string"/>
 <xsd:element name="prenom" type="xsd:string"/>
</xsd:schema>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : schema.rng -->
<element
         name="employes"
         xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
 <oneOrMore>
  <element name="employe">
  <attribute name="id">
   <data type="ID"/>
  </attribute>
  <attribute name="service">
   <data type="NMTOKEN"/>
  </attribute>
   <element name="nom">
    <text/>
   </element>
   <element name="prenom">
    <text/>
   </element>
  </element>
 </oneOrMore>
</element>
```

27.2.6 / L'objet *DOMImplementation*

L'objet *DOMImplementation* procure un ensemble de méthodes permettant de créer un document et une délaration de type de document. Etant donné qu'il n'est pas possible d'affecter une DTD à un objet *DOMDocument* déjà chargé, les méthodes *createDocumentType()* et *createDocument()* pallient à ce problème.

```
$impl = new DOMImplementation();
$dtd = $doc_xml->createDocumentType(
                'html',
                '-//W3C//DTD XHTML 1.0 Strict//EN',
                'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd');
$doc = $impl->createDocument(
                 'http://www.w3.org/1999/xhtml',
                 'html',
                 $dtd);
```

En premier lieu, une instance de *DOMImplementation* doit être créée. Puis, à partir de l'objet obtenu, la méthode *createDocumentType()* construit une déclaration de type de document avec le nom de l'élément racine et éventuellement les identificateurs public et système. Ce dernier peut être un chemin vers une définition DTD locale. Enfin, un objet *DOMDocument* est conçu avec la méthode *createDocument()* à laquelle on spécifie l'bojet *DOMDocumentType* et éventuellement une URI d'espace de noms et un nom qualifié.

Finalement, il ne reste plus qu'à importer l'élément racine du document dénué de DTD, et l'ajouter dans le nouveau document XML.

```
<?php
 $fichier = 'societe';
 $doc_xml = new DOMDocument();
 if($doc_xml->load($fichier . '.xml')){
  echo '<h3>Le document XML '
       . $fichier .
       ' a été chargé !</h3>';

  $impl = new DomImplementation();
  $dtd = $impl->createDocumentType(
                    $fichier, '', $fichier . '.dtd');
  $doc = $impl->createDocument('', '',$dtd);
  $doc->version = '1.0';
  $doc->encoding = 'ISO-8859-1';
  $doc->standalone = 'no';
  $doc->validateOnParse = true;
  $racine = $doc->importNode($doc_xml->documentElement, true);
  $doc->appendChild($racine);
  echo '<pre>'
     . str_replace('<', '<', $doc->saveXML())
     . '</pre>';
  if($doc->validate())
    echo '<p>Le document XML est valide '
      . 'par rapport à la DTD</p>';
 }
 else
  echo '<p style="color:red">Le document XML '
       . $fichier .
       ' n\'a pu être chargé !</p>';
?>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : societe.xml -->
<societe>
 <services>
  <service id="DG001">
   <designation>Direction générale</designation>
  </service>
  <service id="ST001">
   <designation>Service technique</designation>
  </service>
  <service id="SC001">
   <designation>Service commercial</designation>
```

```
    </service>
    <service id="SC101">
      <designation>Service clientèle</designation>
    </service>
    <service id="RH001">
      <designation>Ressources humaines</designation>
    </service>
    <service id="SQ001">
      <designation>Service qualité</designation>
    </service>
    <service id="SP001">
      <designation>Service production</designation>
    </service>
    <service id="RD001">
      <designation>Recherche et développement</designation>
    </service>
  </services>
  <divisions>
    <division id="A001" services="DG001 RH001"/>
    <division id="B001" services="SC001 SC101"/>
    <division id="C001" services="SP001 SQ001 RD001"/>
  </divisions>
  <employes>
    <employe id="RJ1002" service="DG001">
      <nom>Robierre</nom>
      <prenom>Jean</prenom>
    </employe>
    <employe id="LA1012" service="DG001">
      <nom>Lardut</nom>
      <prenom>Anne</prenom>
    </employe>
    <employe id="GA1013" service="ST001">
      <nom>Guilde</nom>
      <prenom>Angelique</prenom>
    </employe>
    <employe id="HP1022" service="SC001">
      <nom>Henry</nom>
      <prenom>Paul</prenom>
    </employe>
    <employe id="MM1045" service="RH001">
      <nom>Mortier</nom>
      <prenom>Marc</prenom>
    </employe>
    <employe id="LS1102" service="SQ001">
      <nom>Lebreton</nom>
      <prenom>Sophie</prenom>
    </employe>
    <employe id="JM1095" service="RD001">
      <nom>Jolie</nom>
      <prenom>Martine</prenom>
    </employe>
    <employe id="MT1036" service="SC101">
      <nom>Marcelin</nom>
      <prenom>Tania</prenom>
    </employe>
    <employe id="LL1029" service="SC101">
      <nom>Leger</nom>
      <prenom>Laurence</prenom>
    </employe>
    <employe id="DM1052" service="SC001">
      <nom>Duroi</nom>
      <prenom>Maxime</prenom>
    </employe>
  </employes>
</societe>

<?xml version="1.0" encoding="UTF-8"?>
<!-- Fichier : societe.dtd -->
<!ELEMENT designation (#PCDATA)>
<!ELEMENT division EMPTY>
<!ATTLIST division
   id ID #REQUIRED
   services IDREFS #REQUIRED
>
<!ELEMENT divisions (division+)>
```

```
<!ELEMENT employe (nom, prenom)>
<!ATTLIST employe
   id ID #REQUIRED
   service IDREF #REQUIRED
>
<!ELEMENT employes (employe+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT service (designation)>
<!ATTLIST service id ID #REQUIRED>
<!ELEMENT services (service+)>
<!ELEMENT societe (services, divisions, employes)>
```

27.2.7 / La gestion des inclusions

La méthode *xinclude()* indique à PHP de remplacer les déclarations d'inclusion XML par le contenu des fichiers qui leurs sont rattachés.

```
<?php
 $fichier = 'societe';
 $doc_xml = new DOMDocument();
 if($doc_xml->load($fichier . '.xml')){
  echo '<h3>Le document XML '
       . $fichier .
       ' a été chargé !</h3>';
  $doc_xml->xinclude();
  echo '<pre>'
     . str_replace('<', '<', $doc_xml->saveXML())
     . '</pre>';
 }
 else
  echo '<p style="color:red">Le document XML '
       . $fichier .
       ' n\'a pu être chargé !</p>';
?>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : societe.xml -->
<societe xmlns:xi="http://www.w3.org/2001/XInclude">
 <xi:include href="services.xml">
  <xi:fallback>
   <services>
    <service id="id">
     <designation/>
    </service>
   </services>
  </xi:fallback>
 </xi:include>
 <xi:include href="divisions.xml">
  <xi:fallback>
   <divisions>
    <division id="id" services="services"/>
   </divisions>
  </xi:fallback>
 </xi:include>
 <xi:include href="employes.xml">
  <xi:fallback>
   <employes>
    <employe id="id" service="service">
     <nom/>
     <prenom/>
    </employe>
   </employes>
  </xi:fallback>
 </xi:include>
</societe>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : services.dtd -->
<services>
 <service id="DG001">
  <designation>Direction générale</designation>
 </service>
 <service id="ST001">
  <designation>Service technique</designation>
 </service>
 <service id="SC001">
  <designation>Service commercial</designation>
 </service>
 <service id="SC101">
  <designation>Service clientèle</designation>
 </service>
 <service id="RH001">
  <designation>Ressources humaines</designation>
 </service>
```

```
  <service id="SQ001">
    <designation>Service qualité</designation>
  </service>
  <service id="SP001">
    <designation>Service production</designation>
  </service>
  <service id="RD001">
    <designation>Recherche et développement</designation>
  </service>
</services>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : divisions.dtd -->
<divisions>
  <division id="A001" services="DG001 RH001"/>
  <division id="B001" services="SC001 SC101"/>
  <division id="C001" services="SP001 SQ001 RD001"/>
</divisions>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Fichier : employes.dtd -->
<employes>
  <employe id="RJ1002" service="DG001">
    <nom>Robierre</nom>
    <prenom>Jean</prenom>
  </employe>
  <employe id="LA1012" service="DG001">
    <nom>Lardut</nom>
    <prenom>Anne</prenom>
  </employe>
  <employe id="GA1013" service="ST001">
    <nom>Guilde</nom>
    <prenom>Angelique</prenom>
  </employe>
  <employe id="HP1022" service="SC001">
    <nom>Henry</nom>
    <prenom>Paul</prenom>
  </employe>
  <employe id="MM1045" service="RH001">
    <nom>Mortier</nom>
    <prenom>Marc</prenom>
  </employe>
  <employe id="LS1102" service="SQ001">
    <nom>Lebreton</nom>
    <prenom>Sophie</prenom>
  </employe>
  <employe id="JM1095" service="RD001">
    <nom>Jolie</nom>
    <prenom>Martine</prenom>
  </employe>
  <employe id="MT1036" service="SC101">
    <nom>Marcelin</nom>
    <prenom>Tania</prenom>
  </employe>
  <employe id="LL1029" service="SC101">
    <nom>Leger</nom>
    <prenom>Laurence</prenom>
  </employe>
  <employe id="DM1052" service="SC001">
    <nom>Duroi</nom>
    <prenom>Maxime</prenom>
  </employe>
</employes>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Fichier : societe.dtd -->
<!ELEMENT designation (#PCDATA)>
<!ELEMENT division EMPTY>
<!ATTLIST division
   id ID #REQUIRED
   services IDREFS #REQUIRED
>
<!ELEMENT divisions (division+)>
<!ELEMENT employe (nom, prenom)>
<!ATTLIST employe
```

```
    id ID #REQUIRED
    service IDREF #REQUIRED
>
<!ELEMENT employes (employe+)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT service (designation)>
<!ATTLIST service id ID #REQUIRED>
<!ELEMENT services (service+)>
<!ELEMENT societe (services, divisions, employes)>
<!ATTLIST societe
    xmlns:xi CDATA #FIXED "http://www.w3.org/2001/XInclude"
>
```

27.2.8 / Les fonctions

Le langage PHP 5 ajoute de nombreuses fonctions et modifient le nom des méthodes par rappor aux versions précédentes, permettant de travailler sur le modèle d'objet de document XML (eXtended Markup Language).

L'activation du module DOM XML/XSLT sur les machines équipées de MS Windows, nécessite de copier les librairies *php_domxml.dll* et *php_xslt.dll* présentes dans le dossier *DLL* du répertoire d'installation de PHP, dans le dossier système *SYSTEM32* du répertoire d'installation du système d'exploitation Windows.

Fonction
Description
$attr = **$DOMAttr->__construct(string nom, string valeur)**;
crée un nouvel attribut avec l'opérateur *new* (ex.: *$attr = new DOMAttr('attr', 'attrvalue')*). Cette méthode ne doit pas être appelée directement.;
$booleen = **$DOMAttr->isId()**;
indique si l'attribut est un identifiant défini.
$cdata = **$DOMCDATASection->__construct(string texte)**;
crée une nouvelle section CDATA avec l'opérateur *new* (ex.: *$cdata = new DOMAttr('Un texte non analysable...')*). Cette méthode ne doit pas être appelée directement.;
$DOMCharacterData->appendData(string chaine);
ajoute la chaîne de caractères à la fin des données textuelles dans le noeud.
$DOMCharacterData->deleteData(int debut, int n);
supprime une sous-chaîne de caractères commençant à l'indice *début* et sur *n* caractères, dans le noeud.
$DOMCharacterData->insertData(int debut, string chaine);
insère une chaîne de caractères à la position *debut* d'unité 16-bit.
$DOMCharacterData->replaceData(int debut, int n, string chaine);
remplace une sous-chaîne de caractères délimitée par la position *debut* jusqu'à *n* caractères, par la chaîne de caractères passée en argument, dans le noeud *DOMCharacterData*.
$chaine = **$DOMCharacterData->substringData(int debut, int n)**;
retourne une sous-chaîne de caractères extrait du noeud courant à partir d'une position de début et jusqu'à *n* caractères.
$commentaire = **$DOMComment->__construct(string valeur)**;
crée un nouveau commentaire à l'aide d'une instanciation de la classe *DOMComment* (ex. : *$commentaire = new DOMComment('commentaire...');*).
$document = **$DOMDocument->__construct(string version[, string encodage])**;
crée un nouveau document XML à l'aide d'une instanciation de la classe *DOMDocument* (ex. : *$document = new DOMDocument('1.0', 'UTF-16');*).
$attribut\|false = **$DOMDocument->createAttribute(string nom)**;
crée un nouvel attribut portant le nom spécifié.
$attribut\|false = **$DOMDocument->createAttributeNS(string URI_espace_noms, string nom)**;
crée un nouvel attribut avec un espace de noms associé et portant le nom qualifié spécifié (préfixe:nom).
$sectionCDATA\|false = **$DOMDocument->createCDATASection(string valeur)**;

crée une nouvelle section CDATA avec une chaîne de caractères non analysée par le parser XML.

$commentaire|false = **$DOMDocument->createComment(string commentaire)**;

crée un nouveau commentaire avec le texte spécifié.

$fragment|false = **$DOMDocument->createDocumentFragment()**;

crée un fragment de document.

$element|false = **$DOMDocument->createElement(string nom[, string valeur])**;

crée un nouvel élément portant le nom spécifié et contenant éventuellement une valeur textuelle.

$element|false = **$DOMDocument->createElementNS(**
string URI_espace_noms, string nom_qualifie[, string valeur]);

crée un nouvel élément avec un espace de noms associé, portant le nom qualifié (préfixe:nom) spécifié et contenant éventuellement une valeur textuelle.

$ref_entite**$DOMDocument->createEntityReference(string nom)**;

crée une nouvelle référence d'entité portant le nom spécifié (ex.: &).

$pi = **$DOMDocument->createProcessingInstruction(**
string cible[, string expression]);

crée une nouvelle instruction de traitement (ex.: *$pi = $dom->createProcessingInstruction('xml-stylesheet', 'href="style.xsl" type="text/xsl"');*).

$texte = **$DOMDocument->createTextNode(string texte)**;

crée un nouveau noeud textuel contenant le texte spécifié.

$element = **$DOMDocument->getElementById(string identifiant)**;

retourne l'élément portant un attribut *id* ayant la valeur identifiante spécifiée.

$liste_noeuds = **$DOMDocument->getElementsByTagName(string nom)**;

retourne une liste de noeuds portant le nom passé en argument.

$liste_noeuds = **$DOMDocument->getElementsByTagNameNS(**
string URI_espace_noms, string nom);

retourne une liste de noeuds portant le nom passé en argument et dans l'espace de noms spécifié.

$noeud = **$DOMDocument->importNode(DOMNode noeud[, boolean recursif])**;

importe un noeud et éventuellement sa descendance (*recursif = true*) dans le document courant.

$booleen = **$DOMDocument->load(string fichier[, int options])**;

charge un document XML préalablement créé, à partir du fichier spécifié.

$booleen = **$DOMDocument->loadHTML(string source)**;

charge un document HTML préalablement créé, à partir d'une source textuelle passée en argument.

$booleen = **$DOMDocument->loadHTMLFile(string fichier)**;

charge un document HTML préalablement créé, à partir du fichier spécifié.

$booleen = **$DOMDocument->loadXML(string source[, int options])**;

charge un document XML préalablement créé, à partir d'une source textuelle spécifiée.

$DOMDocument->normalize();

normalise un document XML.

$booleen = **$DOMDocument->relaxNGValidate(string fichier)**;

vérifie la validité du document XML en fonction du fichier contenant un schéma Relax NG.

$booleen = **$DOMDocument->relaxNGValidateSource(string source)**;

vérifie la validité du document XML en fonction de la source contenant un schéma Relax NG.

$nb_octets|false = **$DOMDocument->save(string fichier)**;

sauvegarde le document XML dans le fichier spécifié.

$source|false = **$DOMDocument->saveHTML()**;

sauvegarde le document HTML dans une chaîne de caractères.

$nb_octets|false = **$DOMDocument->saveHTMLFile(string fichier)**;

sauvegarde un document HTML dans le fichier spécifié.

$source|false = **$DOMDocument->saveXML(DOMNode noeud)**;

sauvegarde le noeud XML dans une chaîne de caractères.

$booleen = **$DOMDocument->schemaValidate(string fichier)**;

vérifie la validité du document XML en se fondant sur le fichier contenant un schéma XML.

$DOMDocument->schemaValidateSource();

vérifie la validité du document XML en se fondant sur la source textuelle contenant un schéma XML.

$booleen = **$DOMDocument->validate()**;

vérifie la validité du document XML en se fondant sur sa définition de type de document (DTD).

$nb_inclusions = **$DOMDocument->xinclude([int options])**;

remplace les inclusions XML (XInclude) dans un document XML et éventuellement en focntion des paramètres Libxml.

$element = **$DOMElement->__construct(**
string nom[, string valeur[, string URI_espace_noms]]);

crée un nouvel élément à l'aide d'une instanciation de classe (ex.:
$element = new DOMElement('cours', 'Le langage PHP');).

$val_attribut = **$DOMElement->getAttribute(string nom)**;

retourne la valeur de l'attribut portant le nom spécifié.

$attribut = **$DOMElement->getAttributeNode(string nom)**;

retourne l'attribut portant le nom spécifié.

$attribut = **$DOMElement->getAttributeNodeNS(**
string URI_espace_noms, string nom);

retourne l'attribut portant le nom spécifié et dans l'espace de noms passé en argument.

$val_attribut = **$DOMElement->getAttributeNS(**
string URI_espace_noms, string nom);

retourne la valeur de l'attribut portant le nom spécifié et inclus dans l'espace de noms passé en argument.

$liste_elements = **$DOMElement->getElementsByTagName(string nom)**;

retourne une liste de noeuds portant le nom spécifié.

$liste_elements = **$DOMElement->getElementsByTagNameNS(**
string URI_espace_noms, string nom);

retourne une liste de noeuds portant le nom spécifié et inclus dans l'espace de noms passé en argument.

$booleen = **$DOMElement->hasAttribute(string nom)**;

vérifie si un attribut portant le nom donné, existe dans l'élément courant.

$booleen = **$DOMElement->hasAttributeNS(**
string URI_espace_noms, string nom);

vérifie si un attribut portant le nom donné et inclus dans l'espace de noms indiqué, existe dans l'élément courant.

$DOMElement->removeAttribute(string nom);

supprime l'attribut portant le nom spécifié.

$booleen = **$DOMElement->removeAttributeNode(DOMAttr attribut)**;

supprime l'attribut spécifié.

$booleen = **$DOMElement->removeAttributeNS(**
string URI_espace_noms, string nom);

supprime l'attribut portant le nom spécifié et inclus dans l'espace de noms donné.

$booleen = **$DOMElement->setAttribute(string nom, string valeur)**;

ajoute un nouvel attribut portant le nom indiqué et contenant la valeur fournie.

$attribut|NULL = **$DOMElement->setAttributeNode(DOMAttr attribut)**;

ajoute un nouvel attribut à l'élément courant.

$attribut = **$DOMElement->setAttributeNodeNS(**
string URI_espace_noms, DOMAttr attribut);

ajoute un nouvel attribut portant le nom spécifié et à inclure dans l'espace de noms indiqué, à l'élément

$DOMElement->setAttributeNS(
string URI_espace_noms, string nom, string valeur);

ajoute un nouvel attribut portant le nom spécifié, contenant la valeur donnée et à inclure dans l'espace de noms indiqué, à l'élément

$attribut = **$DOMAttr->__construct(string nom)**;

crée une nouvelle référence d'entité à l'aide d'une instanciation de classe (ex.: *$ref_entite = new DOMEntityReference('é');*).

$impl_dom = **$DOMImplementation->__construct()**;

crée une nouvelle implémentation DOM à l'aide d'une instanciation de classe (ex.: *$impl_DOM = new DOMImplementation();*).

Crée un nouvel objet DOMImplementation

$document|NULL = **$DOMImplementation->createDocument(**
[string URI_espace_noms[, string nom_qualifie[, DOMDocumentType dtd]]]);

crée un document XML avec éventuellement un espace de noms, un nom qualifié pour la racine du document, et une DTD.

$type_document = **$DOMImplementation->createDocumentType(**
[string nom_qualifie[, string publicId[, string systemId]]]);

crée une déclaration du type de document avec éventuellement un nom qualifié pour l'élément racine du document, un identifiant public externe et un identifiant système externe.

$booleen = **$DOMImplementation->hasFeature(**
string caracteristique, string version);

vérifie si l'implémentation DOM implémente une fonctionnalité spécifique.

$noeud = **$DOMNamedNodeMap->getNamedItem(string nom)**;

retourne le noeud nommé par le nom spécifié.

$noeud = **$DOMNamedNodeMap->getNamedItemNS(**
string URI_espace_noms, string nom);

retourne le noeud portant le nom spécifié et inclus dans l'espace de noms indiqué.

$noeud = **$DOMNamedNodeMap->item(int index)**;

retourne le noeud positionné à l'index fourni.

$noeud = **$DOMNode->appendChild(DOMNode fils)**;

ajoute un nouveau noeud à la fin des noeuds de l'objet *DOMNode*.

$noeud = **$DOMNode->cloneNode([boolean recursif])**;

clone un noeud et éventuellement l'ensemble de sa descendance (*recursif = true*).

$booleen = **$DOMNode->hasAttributes()**;

vérifie si le noeud possède des attributs.

$booleen = **$DOMNode->hasChildNodes()**;

vérifie si le noeud possède des noeuds enfants.

$noeud = **$DOMNode->insertBefore(DOMNode fils[, DOMNode reference])**;

Ajoute un nouveau fils à la fin des enfants du noeud courant.

$booleen = **$DOMNode->isSameNode(DOMNode noeud)**;

vérifie si deux noeuds sont identiques.

$booleen = **$DOMNode->isSupported(string caracteristique, string version)**;

vérifie si la fonctionnalité fournie est disponible pour la version spécifiée.

$URI_espace_noms = **$DOMNode->lookupNamespaceURI(string prefixe)**;

retourne l'URI de l'espace de noms correspondant au préfixe indiqué.

$prefixe = **$DOMNode->lookupPrefix(string URI_espace_noms)**;

retourne le préfixe correspondant à l'URI de l'espace de noms spécifiée.

$DOMNode->normalize();

normalise le noeud XML.

$noeud = **$DOMNode->removeChild(DOMNode ancien_noeud)**;

supprime le noeud spécifié de la liste des enfants du noeud courant.

$noeud = **$DOMNode->replaceChild(**
DOMNode nouv_node, DOMNode anc_noeud);

remplace un ancien noeud par un nouveau au sein des enfant du noeud courant.

$noeud|NULL = **$DOMNodelist->item(int index)**;

retourne le noeud positionné à l'index spécifié.

$pi = **$DOMProcessingInstruction->__construct(string nom[, string valeur])**;

crée une nouvelle instruction de traitement portant le nom spécifié et contenant la valeur indiquée. Pour utiliser ce constructeur, il faut utiliser une instanciation de classe (ex.: *$pi = new DOMProcessingInstruction('xml-stylesheet', 'type="text/xsl" href="feuille.xsl"');*).

$noeud_texte = **$DOMText->__construct([string valeur])**;

crée un nouvel objet *DOMText* à l'aide d'une instanciation de classe (ex.: *$noeud_texte = new DOMText('Un texte...');*).

$booleen = **$DOMText->isWhitespaceInElementContent()**;

vérifie si le noeud de texte courant contient des espaces blancs (espace, tabulation, fin de ligne, etc.).

$noeud_texte = **$DOMText->splitText(int position)**;

découpe le noeud textuel courant en deux noeuds à l'endroit spécifié.

$obj_xpath = **$DOMXPath->__construct(DOMDocument document)**;

crée une nouvelle expression XPath à l'aide d'une instanciation de classe (ex.: *$noeud_texte = new DOMXPath('/racine/enfant[@id = "OCK1254"]');*).

$chaine|$liste_noeuds = **$DOMXPath->evaluate(**
string expression[, DOMNode contexte]);

évalue l'expression XPath fournie et retourne un contenu textuel ou une liste de noeuds. Le noeud contextuel peut être fournie pour exécuter des requêtes XPath relative à ce noeud plutôt qu'à l'élément racine du document XML.

$liste_noeuds = **$DOMXPath->query(string expression[, DOMNode contexte])**;

exécute la requête XPath spécifiée et retourne une liste de noeuds.

$booleen = **$DOMXPath->registerNamespace(**
string prefixe, string URI_espace_noms);

enregistre l'espace de noms et le préfixe spécifiés avec l'objet *DOMXPath*.

$element|false = **dom_import_simplexml(SimpleXMLElement noeud)**;

transforme un objet *SimpleXMLElement* en un objet *DOMElement*.

27.2.9 / Les exceptions

Les exceptions interrompent l'exécution normale d'un programme, en raison d'une erreur qu'aurait rencontré PHP.

Certaines méthodes des objets du DOM sont susceptibles de lancer une exception suite à une erreur que rencontre PHP dans la tentative d'accomplir l'opération requise. Par exemple, la méthode *importNode()* de l'objet *DOMDocument* peut être interrompu durant son exécution par une exception si le noeud ne peut être importé.

Il est possible de **capturer une exception avant qu'elle n'interrompt définitivement le programme**. Pour cela, il suffit de placer le bloc de codes sensible dans le gestionnaire d'exception *try...catch*.

```
try {
  $noeud = $doc->importNode($nouv_noeud);
}
catch(DOMException $e){
  //Gestion de l'exception
  echo $e->getMessage();
}
```

Les codes d'exception précisent la nature de l'erreur qui se serait produite durant le déroulement du programme.

Constante	Valeur	Description
Méthodes susceptibles de produire l'exception		
DOM_INDEX_SIZE_ERR	1	indique que l'index ou la taille est négatif ou plus grand que la valeur autorisée.
DOMCharacterData->deleteData(), DOMCharacterData->insertData(), DOMCharacterData->replaceData(), DOMCharacterData->substringData()		
DOMSTRING_SIZE_ERR	2	indique que le texte fourni ne tient pas dans le type *DOMString*.
DOM_HIERARCHY_REQUEST_ERR	3	indique qu'un noeud est inséré à un endroit non autorisé.
DOMNode->appendChild(), DOMNode->insertBefore(), DOMNode->replaceChild()		
DOM_WRONG_DOCUMENT_ERR	4	indique qu'un noeud est utilisé dans un document différent de celui qui l'a créé.
DOMDocument->saveXML(), DOMImplementation->createDocument(), DOMNode->appendChild(), DOMNode->insertBefore(), DOMNode->replaceChild()		
DOM_INVALID_CHARACTER_ERR	5	indique qu'un caractère invalide ou non autorisé est actuellement utilisé.
DOMDocument->createAttribute(), DOMDocument->createAttributeNS(), DOMDocument->createElement(), DOMDocument->createElementNS(), DOMDocument->createEntityReference(), DOMDocument->createProcessingInstruction()		
DOM_NO_DATA_ALLOWED_ERR	6	indique que des données sont employées dans un noeud qui n'en supporte aucunes.

DOM_NO_MODIFICATION_ALLOWED_ERR	7	indique qu'une tentative de modification a été opérée dans un objet qui n'en supportent aucunes.
DOMElement->removeAttribute(), DOMElement->removeAttributeNode(), DOMElement->removeAttributeNS(), DOMElement->setAttribute(), DOMElement->setAttributeNode(), DOMElement->setAttributeNodeNS(), DOMElement->setAttributeNS(), DOMNode->appendChild(), DOMNode->insertBefore(), DOMNode->removeChild(), DOMNode->replaceChild()		
DOM_NOT_FOUND_ERR	8	indique qu'une tentative de référencer un noeud a été opérée alors qu'il n'existe pas dans ce contexte.
DOMElement->removeAttributeNode(), DOMNode->removeChild(), DOMNode->insertBefore(), DOMNode->replaceChild()		
DOM_NOT_SUPPORTED_ERR	9	indique que l'implémentation ne supporte pas le type de l'objet ou de l'opération requis.
DOM_INUSE_ATTRIBUTE_ERR	10	indique qu'une tentative d'ajout d'un attribut a été effectuée alors que cet attribut a déjà été utilisé ailleurs.
DOM_INVALID_STATE_ERR	11	indique qu'une tentative d'utilisation d'un objet inexistant ou inutilisable a été effectuée.
DOM_SYNTAX_ERR	12	indique qu'une chaîne de caractères invalide ou illégale a été utilisée.
DOM_INVALID_MODIFICATION_ERR	13	indique qu'une tentative est de modification du type d'un objet fondamental a été opérée.
DOM_NAMESPACE_ERR	14	indique qu'une tentative de création ou de modification d'un objet a été opérée alors que ce dernier n'est pas conforme à l'espace de noms en vigueur.
DOMDocument->createAttributeNS(), DOMDocument->createElementNS(), DOMElement->setAttributeNS(), DOMImplementation->createDocument(), DOMImplementation->createDocumentType()		
DOM_INVALID_ACCESS_ERR	15	indique qu'un paramètre ou une opération n'est pas supportée par l'objet fondamental.
DOM_VALIDATION_ERR	16	indique qu'une opération d'ajout ou de modification dun noeud risque de rendre le document invalide.

27.2.10 / Les contantes

Les constantes XML permettent de distinguer le type d'un noeud XML.

L'attribut *nodeType* de l'objet *DOMNode* contient une valeur égale à l'une des constantes XML.

```
if($noeud->nodeType == XML_ELEMENT_NODE)
  echo 'Ce noeud est un élément XML';
else if($noeud->nodeType == XML_CDATA_SECTION_NODE)
  echo 'Ce noeud est une section CDATA';
else if($noeud->nodeType == XML_TEXT_NODE)
  echo 'Ce noeud est un texte';
```

Constante	Valeur	Description
XML_ELEMENT_NODE	1	Le noeud est du type *DOMElement*.
XML_ATTRIBUTE_NODE	2	Le noeud est du type *DOMAttr*.
XML_TEXT_NODE	3	Le noeud est du type *DOMText*.
XML_CDATA_SECTION_NODE	4	Le noeud est du type *DOMCharacterData*.
XML_ENTITY_REF_NODE	5	Le noeud est du type *DOMEntityReference*.
XML_ENTITY_NODE	6	Le noeud est du type *DOMEntity*
XML_PI_NODE	7	Le noeud est du type *DOMProcessingInstruction*.
XML_COMMENT_NODE	8	Le noeud est du type *DOMComment*.
XML_DOCUMENT_NODE	9	Le noeud est du type *DOMDocument*.
XML_DOCUMENT_TYPE_NODE	10	Le noeud est du type *DOMDocumentType*.
XML_DOCUMENT_FRAG_NODE	11	Le noeud est du type *DOMDocumentFragment*.
XML_NOTATION_NODE	12	Le noeud est du type *DOMNotation*.
XML_HTML_DOCUMENT_NODE	13	Le noeud est un document HTML.
XML_DTD_NODE	14	Le noeud est du type *DOMDocumentType*.
XML_ELEMENT_DECL_NODE	15	Le noeud est l'élément de la déclaration de type de document.
XML_ATTRIBUTE_DECL_NODE	16	Le noeud est un attribut de la déclaration de type de document.
XML_ENTITY_DECL_NODE	17	Le noeud est une entité de la déclaration de type de document..
XML_NAMESPACE_DECL_NODE	18	Le noeud est un espace de noms de la déclaration de type de document.
XML_ATTRIBUTE_CDATA	1	
XML_ATTRIBUTE_ID	2	Le noeud est un attribut d'identification ID.
XML_ATTRIBUTE_IDREF	3	Le noeud est un attribut de référence à un identificateur ID.
XML_ATTRIBUTE_IDREFS	4	Le noeud est un attribut de liste de références d'identificateur séparés par un espace blanc.
XML_ATTRIBUTE_ENTITY	5	Le noeud est une entité.
XML_ATTRIBUTE_NMTOKEN	7	Le noeud est un attribut contenant un nom symbolique appelée *token*.
XML_ATTRIBUTE_NMTOKENS	8	Le noeud est un attribut contenant une liste de noms symboliques séparés par un espace blanc.

XML_ATTRIBUTE_ENUMERATION	9	Le noeud est une énumération de valeurs dans une déclaration d'attribut.
XML_ATTRIBUTE_NOTATION	10	Le noeud est du type *DOMNotation*.

Ces constantes ne sont disponibles que si l'extension a été compilée avec PHP, ou bien chargée au moment de l'exécution.

27.3 / L'extension SimpleXML

L'extension SimpleXML propose des fonctionnalités élémentaires pour manipuler des documents XML.

La manipulation d'un document XML par cette extension permet le chargement d'une source XML à partir d'un fichier ou d'un texte, l'extraction des attributs des éléments, l'exploration du document XML, l'itération des enfants d'un noeud élément.

Il est même possible de modifier le document XML à partir de cette extension. En effet, il suffit d'accéder à un élément SimpleXML par l'intermédiaire des champs de l'objet SimpleXML constitués suite au chargement d'une source XML (voir exemple).

Fonction
Description
$chaine = **$SimpleXMLElement->asXML()**
retourne une représentation XML de l'élément SimpleXML.
$element = **$SimpleXMLElement->attributes([string donnee])**;
identifie les attributs d'un élément.
$element = **$SimpleXMLElement->children([string prefixe])**;
retourne l'élément enfant du noeud courant.
$elements = **$SimpleXMLElement->xpath(string requete)**;
exécute une requête XPath sur l'élément XML courant et retourne un tableau d'objets *SimpleXMLElement*.
$element|false = **simplexml_import_dom(DOMNode noeud[, string nom_classe])**;
transforme un objet *DOMElement* en un objet *SimpleXMLElement*.
$element|false = **simplexml_load_file(** **string fichier[, string nom_classe[, int options]])**;
charge le fichier spécifié et le transforme en un élément SimpleXML.
$element|false = **simplexml_load_string(** **string fichier[, string nom_classe[, int options]])**;
charge la source XML spécifiée et la transforme en un élément SimpleXML.

Exemple [voir]

```
<?php
$source = <<<XML
<?xml version="1.0" standalone="yes" encoding="iso-8859-1"?>
<liste>
  <logiciel>
    <nom langue="US" systeme_exploitation="Win">Cooktop 2.200</nom>
    <commentaire>Un editeur XML, XSLT, XPath et
                  DTD puissant et totalement gratuit.</commentaire>
    <editeur adresse="http://xmleverywhere.com/cooktop/">XML Everywhere</editeur>
    <prix monnaie="$US">00.00</prix>
  </logiciel>
  <logiciel>
    <nom langue="US" systeme_exploitation="Win">XML Spy 4.1</nom>
    <commentaire>Un editeur XML desormais mature.</commentaire>
    <editeur adresse="http://www.xmlspy.com/default.html">Altova Inc.</editeur>
    <prix monnaie="$US">199,00</prix>
  </logiciel>
  <logiciel>
    <nom langue="US" systeme_exploitation="Win">XML Spy 4.1 B2B Server</nom>
    <commentaire>La version 4 en version Business to business.</commentaire>
    <editeur adresse="http://www.xmlspy.com/default.html">Altova Inc.</editeur>
    <prix monnaie="$US">1 999,00</prix>
  </logiciel>
  <logiciel>
    <nom langue="US" systeme_exploitation="Win">XMLwriter v1.21</nom>
    <commentaire>Permet de creer des documents XML.</commentaire>
    <editeur adresse="http://xmlwriter.net/">Wattle Software</editeur>
    <prix monnaie="$US">75,00</prix>
  </logiciel>
</liste>
XML;
?>
<html>
<body>
<?php
$xml = simplexml_load_string($source);
var_dump($xml);

/*Affiche : object(SimpleXMLElement)#1 (1) { ["logiciel"]=> array(4) { [0]=> object(SimpleXMLElement)#2
(4) { ["nom"]=> string(13) "Cooktop 2.200" ["commentaire"]=> string(66) "Un editeur XML, XSLT, XPath et
DTD puissant et totalement gratuit." ["editeur"]=> string(14) "XML Everywhere" ["prix"]=> string(5) "00.00"
} [1]=> object(SimpleXMLElement)#3 (4) { ["nom"]=> string(11) "XML Spy 4.1" ["commentaire"]=>
string(32) "Un editeur XML desormais mature." ["editeur"]=> string(11) "Altova Inc." ["prix"]=> string(6)
"199,00" } [2]=> object(SimpleXMLElement)#4 (4) { ["nom"]=> string(22) "XML Spy 4.1 B2B Server"
["commentaire"]=> string(45) "La version 4 en version Business to business." ["editeur"]=> string(11)
"Altova Inc." ["prix"]=> string(8) "1 999,00" } [3]=> object(SimpleXMLElement)#5 (4) { ["nom"]=> string(15)
"XMLwriter v1.21" ["commentaire"]=> string(34) "Permet de creer des documents XML." ["editeur"]=>
string(15) "Wattle Software" ["prix"]=> string(5) "75,00" } } } */

//Modification de la valeur d'un élément
$xml->logiciel[0]->nom = 'Cooktop';
var_dump($xml);

/*Affiche : object(SimpleXMLElement)#1 (1) { ["logiciel"]=> array(4) { [0]=> object(SimpleXMLElement)#2
(4) { ["nom"]=> string(13) "Cooktop" ["commentaire"]=> string(66) "Un editeur XML, XSLT, XPath et
DTD puissant et totalement gratuit." ["editeur"]=> string(14) "XML Everywhere" ["prix"]=> string(5) "00.00"
} [1]=> object(SimpleXMLElement)#3 (4) { ["nom"]=> string(11) "XML Spy 4.1" ["commentaire"]=>
string(32) "Un editeur XML desormais mature." ["editeur"]=> string(11) "Altova Inc." ["prix"]=> string(6)
"199,00" } [2]=> object(SimpleXMLElement)#4 (4) { ["nom"]=> string(22) "XML Spy 4.1 B2B Server"
["commentaire"]=> string(45) "La version 4 en version Business to business." ["editeur"]=> string(11)
"Altova Inc." ["prix"]=> string(8) "1 999,00" } [3]=> object(SimpleXMLElement)#5 (4) { ["nom"]=> string(15)
"XMLwriter v1.21" ["commentaire"]=> string(34) "Permet de creer des documents XML." ["editeur"]=>
string(15) "Wattle Software" ["prix"]=> string(5) "75,00" } } } */

//Sauvegarde des modifications
$contenu = $xml->asXML();
$res = fopen('fichier_sauv.xml', 'x+');
fwrite($res, $contenu)
fclose($res);

//Utilisation de XPath
var_dump($xml->xpath('/liste/logiciel/nom'));
/*Affiche : array(4) { [0]=> object(SimpleXMLElement)#7 (1) { [0]=> string(7) "Cooktop" } [1]=>
```

```
object(SimpleXMLElement)#6 (1) { [0]=> string(11) "XML Spy 4.1" } [2]=> object(SimpleXMLElement)#8
(1) { [0]=> string(22) "XML Spy 4.1 B2B Server" } [3]=> object(SimpleXMLElement)#9 (1) { [0]=>
string(15) "XMLwriter v1.21" } }

$element = $xml->xpath('/liste/logiciel[2]/editeur');
var_dump($element);
//Affiche : array(1) { [0]=> object(SimpleXMLElement)#7 (1) { [0]=> string(11) "Altova Inc." } }

//Accès aux attributs d'un élément XML
foreach($element[0]->attributes() as $nom => $valeur){
  echo '<p>' . $nom . ' = ' . $valeur . '</p>';
}
//Affiche : adresse = http://www.xmlspy.com/default.html

//Chargement d'un fichier XML
$sxml = simplexml_load_file('fichier.xml');
parcourir($sxml);
function parcourir($sxml){
  $i = 0;
  //Accès aux enfants de l'élément courant
  foreach($sxml->children() as $element){
     echo $i++ . ' ';
     var_dump($element);
     echo '<br>';
     parcourir($element);
  }
  echo '<br>';
}

/*Affiche : 0 object(SimpleXMLElement)#4 (4) { ... } 0 object(SimpleXMLElement)#7 (1) { [0]=> string(13)
"Cooktop 2.200" } 1 object(SimpleXMLElement)#9 (1) { [0]=> string(66) "Un editeur XML, XSLT, XPath et
DTD puissant et totalement gratuit." } 2 object(SimpleXMLElement)#8 (1) { [0]=> string(14) "XML
Everywhere" } 3 object(SimpleXMLElement)#7 (1) { [0]=> string(5) "00.00" } 1
object(SimpleXMLElement)#7 (4) { ... } 0 object(SimpleXMLElement)#5 (1) { [0]=> string(11) "XML Spy
4.1" } 1 object(SimpleXMLElement)#8 (1) { [0]=> string(32) "Un editeur XML desormais mature." } 2
object(SimpleXMLElement)#9 (1) { [0]=> string(11) "Altova Inc." } 3 object(SimpleXMLElement)#5 (1) {
[0]=> string(6) "199,00" } 2 object(SimpleXMLElement)#5 (4) { ... } 0 object(SimpleXMLElement)#4 (1) {
[0]=> string(22) "XML Spy 4.1 B2B Server" } 1 object(SimpleXMLElement)#9 (1) { [0]=> string(45) "La
version 4 en version Business to business." } 2 object(SimpleXMLElement)#8 (1) { [0]=> string(11)
"Altova Inc." } 3 object(SimpleXMLElement)#4 (1) { [0]=> string(8) "1 999,00" } 3
object(SimpleXMLElement)#4 (4) { ... } 0 object(SimpleXMLElement)#7 (1) { [0]=> string(15) "XMLwriter
v1.21" } 1 object(SimpleXMLElement)#8 (1) { [0]=> string(34) "Permet de creer des documents XML." } 2
object(SimpleXMLElement)#9 (1) { [0]=> string(15) "Wattle Software" } 3 object(SimpleXMLElement)#7
(1) { [0]=> string(5) "75,00" } */
?>
</body>
</html>
```

27.4 / L'extension XSL de PHP 5

L'extension XSL se base sur les spécifications XSLT (XSL Transformations 1.0) édictées par le W3C.

L'extension XSL permet d'**effectuer des transformations d'un document XML selon une feuille de style XSLT**.

L'extension XSL s'appuie sur la bibliothèque libxslt.

Par défaut, l'extension XSL est inclut dans PHP5. Nénanmoins, il peut être nécessaire de l'activer en ajoutant l'argument *--with-xsl[=répertoire_libxslt]* dans les directives de compilation. Pour Windows, il faut placer le fichier *php_xsl.dll* dans le répertoire SYSTEM[32] et suprimer le point-virgule désactivant l'extension *extension=php_xsl.dll* dans le fichier de configuration *php.ini*.

Fonction
Description
$procXSL = **$XSLTProcessor->__construct()**;
crée un nouvel objet *XSLTProcessor* à l'aide d'une instanciation de classe (ex.: *$proc = new XSLTProcessor();*).
$param = **XSLTProcessor->getParameter(string URI_espace_noms, string nom_local)**;
retourne la valeur d'un paramètre précédemment défini par la fonction *XSLTProcessor->setParameter()*.
$booleen = **XSLTProcessor->hasExsltSupport()**;
indique si PHP utilise l'extension EXSLT.
XSLTProcessor->importStylesheet(DOMDocument styleXSL);
importe une feuille de style XSLT.
XSLTProcessor->registerPHPFunctions([mixed restriction]);
active l'utilisation de fonctions PHP en tant que fonctions XSLT dans les feuilles de styles XSLT.
$booleen = **XSLTProcessor->removeParameter(string URI_espace_noms, string nom_local)**;
Efface un paramètre
$booleen = **XSLTProcessor->setParameter(string URI_espace_noms, (string nom, string valeur)|array options)**;
définit la ou les valeurs d'un ou plusieurs paramètres pour être utilisé lors du processus de transformation. Le tableau *options* doit contenir des paires nom/valeur.
$document = **XSLTProcessor->transformToDoc(DOMNode noeud)**;
transforme le noeud spécifié en un objet *DOMDocument*.
$nb_octets|false = **XSLTProcessor->transformToURI(DOMDocument doc, string uri)**;
transforme le document en une URI en y appliquand la feuille de style donnée par la méthode *XSLTProcessor->importStylesheet()*.
$resultat = **XSLTProcessor->transformToXML(DOMDocument doc)**;
transforme le document XML en y appliquand une feuille de style donnée par la méthode *XSLTProcessor->importStylesheet()*.

Constante	Valeur	Description
XSL_CLONE_AUTO	0	

XSL_CLONE_NEVER	-1	
XSL_CLONE_ALWAYS	1	

```
<?php
// Chargement du document XML
$xml = new DOMDocument;
$xml->load('logitheque.xml');

// Chargement de la feuille de style
$xsl = new DOMDocument;
$xsl->load('param.xsl');

// Création du processeur XSLT
$proc = new XSLTProcessor;

//Affectation de la feuille de style
$proc->importStyleSheet($xsl);

// Transformation du document XML selon la feuille XSL
echo $proc->transformToXML($xml);
?>
```

27.4.1 / Transformation XSLT

L'extension XSL permet d'effectuer des transformations XSLT (XML Stylesheet Language Transformation), c'est-à-dire, appliquer à un document XML, un ensemble de règles de restructuration des données, dans le but d'obtenir un document formaté en HTML ou dans un autre balisage XML (WML, RDF, RSS, etc.).

La transformation nécessite un document XML, une feuille de style XSLT et un processeur XSLT capable de mettre en relation les deux documents précités afin d'en générer un troisième qui contiendra le résultat de la transformation.

Le processeur XSLT est créé à partir d'une instanciation de la classe *XSLTProcessor*.

```
$proc_xsl = new XSLTProcessor();
```

La feuille de style XSLT doit être importée dans l'objet *XSLTProcessor*, après avoir été rendue disponible dans le programme par un chargement des règles de style dans un objet *DOMDocument*. Ceci est possible, puisque le langage XSLT n'est autre qu'un des nombreux dialectes dérivant du langage XML.

```
$doc_xsl = new DOMDocument();
$doc_xsl->load('regles.xsl');

$proc_xsl->importStyleSheet($doc_xsl);
```

Le document XML est chargé également dans un objet *DOMDocument*.

```
$doc_xml = new DOMDocument();
$doc_xml->load('document.xml');
```

Désormais, il ne reste plus qu'à opérer la transformation du document XML en suivant les règles de style du document XSLT. L'extension XSL propose trois méthodes de transformation à invoquer sur l'objet *XSLTProcessor*.

La méthode *transformToXML()* transforme selon la feuille de style importée, l'objet *DOMDocument* spécifié en une chaîne de caractères.

```
$resultat = $proc_xsl->transformToXML($doc_xml);
```

La méthode *transformToURI()* transforme l'objet *DOMDocument* en fonction des règles de style, puis stocke le résultat de la transformation dans un fichier situé à l'adresse URI spécifiée.

```
$nb = $proc_xsl->transformToURI($doc_xml, 'file:///resultat.html');
```

La méthode *transformToDoc()* applique les règles de style au noeud XML donné, et place le résultat de la transformation dans un objet *DOMDocument*.

```
$doc_res = $proc->transformToDoc($doc_xml);
```

Cette dernière méthode peut prendre un noeud du document comme argument, mais la transformation se fera malgré cela sur la totalité d'un document XML. En effet, La méthode remonte automatiquement sur le noeud *DOMDocument* si cela est possible.

```
$noeud = $doc_xml->getElementById('JF001LDP');
$doc_res = $proc->transformToDoc($noeud);
//équivaut
$doc_res = $proc->transformToDoc($noeud->ownerDocument);

<?php
  $docxml = new DOMDocument();
  $docxml->validateOnParse = true;
  $docxml->loadXML($xml);

  $docxsl = new DOMDocument();
  $docxsl->loadXML($xsl);

  $proc = new XsltProcessor();
  $proc->importStylesheet($docxsl);
```

```
  $doc = $proc->transformToDoc($docxml);
  echo '<h3>transformToDoc</h3><pre>'
     . htmlEntities($doc->saveXML())
     . '</pre>';

  $nb = $proc->transformToURI($docxml, 'resultat.html');
  echo '<h3>transformToURI</h3><p>'
     . 'Nombre d'octets écrits : ' . $nb . '</p>';

  echo '<h3>transformToXML</h3><pre>'
     . htmlEntities($proc->transformToXML($docxml))
     . '</pre>';
?>

<?xml version="1.0" encoding="iso-8859-1"?>
<!-- Fichier : recueil.xml -->
<!DOCTYPE recueil [
  <!ELEMENT poesie (titre, texte, auteur)>
  <!ATTLIST poesie id ID #REQUIRED>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT texte (#PCDATA)>
  <!ELEMENT auteur (#PCDATA)>
  <!ELEMENT recueil (poesie+)>
]>
<recueil>
  <poesie id="JF001LDP">
    <titre>Locution des pierrots</titre>
    <texte>
      Je ne suis qu'un viveur lunaire
      Qui fait des ronds dans le bassin
      Et cela, sans autre dessein
      Que de devenir légendaire.

      Retroussant d'un air de défin
      Mes manches de Mandarin pâle,
      J'arrondis ma bouche et - j'exhale
      Des conseils doux de Crucifix

      Ah! oui, devenir légendaire,
      Au seuil des siècles charlatans !
      Mais où sont les Lunes d'antan ?
      Et que Dieu n'est-il à refaire ?
    </texte>
    <auteur>Jules Laforgue</auteur>
  </poesie>
  <!-- ... -->
  <poesie id="PV002OTA">
    <titre>Ô triste, triste était mon âme</titre>
    <texte>
      Ô triste, triste était mon âme
      À cause, à cause d'une femme.
      Je ne me suis pas consolé
      Bien que mon coeur s'en soit allé,
      Bien que mon coeur, bien que mon âme
      Eussent fui loin de cette femme.
      Je ne me suis pas consolé
      Bien que mon coeur s'en soit allé.
      Et mon coeur, mon coeur trop sensible
      Dit à mon âme : Est-il possible,
      Est-il possible, - le fût-il -
      Ce fier exil, ce triste exil ?
      Mon âme dit à mon coeur: Sais-je
      Moi-même que nous veut ce piège
      D'être présents bien qu'exilés,
      Encore que loin en allés ?
    </texte>
    <auteur>Paul Verlaine</auteur>
  </poesie>
</recueil>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : style.xsl -->
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/TR/xhtml1/strict">
```

```
  <xsl:output method="html" encoding="ISO-8859-1"/>
  <xsl:template match="/">
    <html>
      <body>
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="poesie">
    <h2 align="center">
      <xsl:value-of select="titre"/>
    </h2>
    <div align="center">
      <pre>
        <xsl:value-of select="texte"/>
      </pre>
    </div>
    <h4 align="center">
      <xsl:value-of select="auteur"/>
    </h4>
  </xsl:template>
</xsl:stylesheet>
```

27.4.2 / Passage de paramètres

Les feuilles de style XSLT peuvent déclarer des paramètres utilisables dans des règles de style. Par dce moyen, il devient possible de passer des informations d'un programme PHP vers une feuille de style.

```
<xsl:param name="nom">Valeur</xsl:param>
```

L'extension XSL permet de gérer ce genre de paramètres.

La méthode *setParameter()* désigne les paramètres et leur associe à chacun une valeur. Deux solutions sont possibles pour spécifier les noms et valeurs des paramètres.

- Spécifier simplement un nom et une valeur.

```
$proc->setParameter('', 'param', 'valeur');
```

- Fournir un tableau associatif contenant des paires nom/valeur.

```
$params = array(
            'param1' => 'valeur1',
            ...,
            'paramN', 'valeurN');
$proc->setParameter('', $params);
```

La méthode *removeParameter()* supprime le paramètre correspondant au nom local passé en argument.

```
if($proc->removeParameter('', 'nom');)
  echo 'Le paramètre "nom" a été supprimé !';
```

Le premier argument des méthodes *setParameter()* et *removeParameter()* est une adresse URI d'espace de noms du paramètre XSLT.

Si un paramètre existe dans une feuille de style et qu'aucune définition de sa valeur n'a été pratiquée (*setParameter()*) ou que la définition a été effacée (*removeParameter()*), alors le paramètre conservera sa valeur par défaut ou s'il n'en possède pas provoquera une erreur lors du processus de transformation.

```
<?php
  $doc_xml = new DOMDocument();
  $doc_xml->validateOnParse = true;
  $doc_xml->load('logitheque.xml');

  $doc_xsl = new DOMDocument();
  $doc_xsl->load('stylesheet.xsl');

  $proc = new XsltProcessor();
  $proc->importStylesheet($doc_xsl);

  $proc->setParameter('', 'valeur', 'Edition Web');

  $nb = $proc->transformToURI($doc_xml, 'resultat.html');
```

```
  echo '

     transformToURI

'
     . 'Nombre d\'octets écrits : ' . $nb . '

';

  echo $proc->transformToXML($doc_xml);
?>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : stylesheet.xsl -->
<xsl:stylesheet
              version="1.0"
              xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output
          method="html"
          media-type="text/html; charset=ISO-8859-1"/>
  <xsl:param name="valeur">XML et XSL</xsl:param>
  <xsl:template match="/">
    <html>
      <head>
        <title>
          La logithèque : la catégorie
          <xsl:value-of select="$valeur"/>
        </title>
      </head>
      <body>
        <h2>
          La logithèque : la catégorie
          <xsl:value-of select="$valeur"/>
        </h2>
        <xsl:for-each
              select="logitheque/categorie[@nom=$valeur]/logiciel">
          <xsl:variable name="url" select="editeur/@lien"/>
          <h3>
            <xsl:value-of select="nom"/>
              (<xsl:value-of select="langue"/>)
          </h3>
          <p><xsl:value-of select="commentaire"/></p>
          <h4>
            <a href="{$url}">
              <xsl:value-of select="editeur"/>
            </a>
          </h4>
          <u>Prix : </u>
          <p>
            <xsl:value-of select="prix"/>
              <xsl:value-of select="prix/@monnaie"/>
          </p>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Fichier : logitheque.xml -->
<!DOCTYPE logitheque SYSTEM "logitheque.dtd">
<logitheque>
  <categorie nom="Edition Web">
    <logiciel code="13404391">
      <nom>HomeSite 4.5</nom>
      <commentaire>Un éditeur HTML Professionnel.</commentaire>
      <editeur lien="http://www.allaire.com/products/">Allaire</editeur>
      <langue>FR</langue>
      <plateforme>Win</plateforme>
      <prix monnaie="FRF">960,00</prix>
    </logiciel>
    <logiciel code="13402703">
      <nom>NetObjects Fusion 5.0</nom>
      <commentaire>Le logiciel le plus complet de création de site.</commentaire>
```

```
  <editeur lien="http://www.netobjects.com/">NetObjects</editeur>
  <langue>US</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">1 239,00</prix>
</logiciel>
<logiciel code="13414170">
  <nom>ShopFactory Pro v4.5</nom>
  <commentaire>Le commerce électronique facile !!!</commentaire>
  <editeur lien="http://www.3d3.com/enter.html?target=Products.html">3D3</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">1 490,00</prix>
</logiciel>
<logiciel code="13404433">
  <nom>UltraEdit 32 version 8.0</nom>
  <commentaire>(Code de débridage) L'éditeur HTML multi usages (1 à 9 postes).</commentaire>
  <editeur lien="http://www.3d3.com/enter.html?target=Products.html">3D3</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">290,00</prix>
</logiciel>
<logiciel code="">
  <nom>BBEdit 6.1</nom>
  <commentaire>Un logiciel d'édition de pages Web</commentaire>
  <editeur lien="http://www.barebones.com/products.html">Bare Bones Software</editeur>
  <langue>FR</langue>
  <plateforme>Mac</plateforme>
  <prix monnaie="$US">119,00</prix>
</logiciel>
<logiciel code="">
  <nom>CoffeeCup HTML Editor 8.9</nom>
  <commentaire>Un logiciel d'édition de pages Web</commentaire>
  <editeur lien="http://www.coffeecup.com/editor/">CoffeeCup Inc.</editeur>
  <langue>US</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="$US">49,00</prix>
</logiciel>
<logiciel code="245306183">
  <nom>GoLive 5.0</nom>
  <commentaire>Création, production et gestion de sites Web.</commentaire>
  <editeur lien="http://www.adobe.com/products/main.html">Adobe Inc.</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">1 835,86</prix>
</logiciel>
<logiciel code="245305205">
  <nom>GoLive 5.0 MAJ</nom>
  <commentaire>Mise à jour depuis 4.0 La dernière version de Golive.</commentaire>
  <editeur lien="http://www.adobe.com/products/main.html">Adobe Inc.</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">901,78</prix>
</logiciel>
<logiciel code="">
  <nom>Hot Dog Pro 6.0</nom>
  <commentaire>Un logiciel d'édition de pages Web</commentaire>
  <editeur lien="http://www.sausagetools.com/products/index.html">Sausage Tools</editeur>
  <langue>US</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="$US">99,95</prix>
</logiciel>
<logiciel code="13406843">
  <nom>Namo WebEditor 4.0</nom>
  <commentaire>Un éditeur Web à la portée de tous.</commentaire>
  <editeur lien="http://www.wska.com/">Wska</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">716,40</prix>
</logiciel>
<logiciel code="">
  <nom>Web Construction Kit 4.0</nom>
  <commentaire>Un logiciel d'édition de pages Web</commentaire>
  <editeur lien="http://pierresoft.com/">PierreSoft</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
```

```
    <prix monnaie="FRF">330,00</prix>
   </logiciel>
   <logiciel code="13413363">
    <nom>Dreamweaver 4</nom>
    <commentaire>Créez vos pages HTML et gérez votre site Web.</commentaire>
    <editeur lien="http://www.macromedia.com/software/">Macromedia</editeur>
    <langue>US</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">2 508,44</prix>
   </logiciel>
   <logiciel code="13414265">
    <nom>Frontpage 2002</nom>
    <commentaire>L'outil XP de création et de gestion de site.</commentaire>
    <editeur lien="http://www.microsoft.com/france/internet/produits/developpement.asp">Microsoft</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">1 271,12</prix>
   </logiciel>
   <logiciel code="13404148">
    <nom>Web Expert 2000</nom>
    <commentaire>Editeur HTML professionnel.</commentaire>
    <editeur lien="http://www.visic.com/webexpert/">Visicom</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">429,00</prix>
   </logiciel>
  </categorie>
  <categorie nom="Feuille de Style">
   <logiciel code="">
    <nom>CoffeeCup StyleSheet Maker Version 4.0</nom>
    <commentaire>Créer vos feuilles de style avec ce logiciel puissant.</commentaire>
    <editeur lien="http://www.coffeecup.com/style/">Coffee Cup</editeur>
    <langue>EN</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="$US">30,00</prix>
   </logiciel>
   <logiciel code="">
    <nom>TopStyle Pro 2.1</nom>
    <commentaire>La version 2.5 disponible en Beta test.</commentaire>
    <editeur lien="http://www.bradsoft.com/topstyle/">Bradbury Software</editeur>
    <langue>EN</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="$US">49,95</prix>
   </logiciel>
   <logiciel code="">
    <nom>Amaya 5.0</nom>
    <commentaire>La référence des éditeurs reconnus par le W3C.</commentaire>
    <editeur lien="http://www.w3.org/Amaya/">Amaya</editeur>
    <langue>EN</langue>
    <plateforme>Win/Linux/Unix</plateforme>
    <prix monnaie="$US">00,00</prix>
   </logiciel>
   <logiciel code="">
    <nom>Style Master 1.9</nom>
    <commentaire>Supporte les dernières recommandations du W3C en matière de feuilles de style.</commentaire>
    <editeur lien="http://www.westciv.com/style_master">Westciv Webware</editeur>
    <langue>EN</langue>
    <plateforme>Win/Mac</plateforme>
    <prix monnaie="$US">29,00</prix>
   </logiciel>
   <logiciel code="">
    <nom>Prime Style 2.0.2</nom>
    <commentaire/>
    <editeur lien="http://www.pconsulting.com.au/style/">Prime Consulting</editeur>
    <langue>EN</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="$US">25,00</prix>
   </logiciel>
   <logiciel code="">
    <nom>StyleOne 2.0.2</nom>
    <commentaire>Un éditeur permettant de créer facilement des feuilles de style.</commentaire>
    <editeur lien="http://www.3-t.com/3-T/products/styleone/">Triple-T</editeur>
    <langue>EN</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="$US">15,00</prix>
```

```
  </logiciel>
  <logiciel code="">
    <nom>BlueFish 0.6</nom>
    <commentaire>Un éditeur de feuille de style pour Linux</commentaire>
    <editeur lien="http://bluefish.openoffice.nl/">Olivier Sessink</editeur>
    <langue>EN</langue>
    <plateforme>Linux</plateforme>
    <prix monnaie="$US">00,00</prix>
  </logiciel>
  <logiciel code="">
    <nom>Interaction 3.51</nom>
    <commentaire>La verion 3.6 bientôt disponible.</commentaire>
    <editeur lien="http://interaction.in-progress.com/">Media Design</editeur>
    <langue>EN</langue>
    <plateforme>Mac</plateforme>
    <prix monnaie="$US">279,00</prix>
  </logiciel>
 </categorie>
 <categorie nom="XML et XSL">
  <logiciel code="">
    <nom>Cooktop  2.200</nom>
    <commentaire>Un éditeur XML, XSLT, XPath et DTD puissant et totalement gratuit.</commentaire>
    <editeur lien="http://xmleverywhere.com/cooktop/">XML Everywhere</editeur>
    <langue>EN</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="$US">00.00</prix>
  </logiciel>
  <logiciel code="">
    <nom>XML Spy 3.5</nom>
    <commentaire>La version 4 bientôt disponible.</commentaire>
    <editeur lien="http://www.xmlspy.com/default.html">Altova Inc.</editeur>
    <langue>EN</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="$US">199,00</prix>
  </logiciel>
  <logiciel code="">
    <nom>XMLwriter v1.21</nom>
    <commentaire>Permet de créer des documents XML.</commentaire>
    <editeur lien="http://xmlwriter.net/">Wattle Software</editeur>
    <langue>EN</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="$US">75,00</prix>
  </logiciel>
  <logiciel code="">
    <nom>XMetaL 2.1</nom>
    <commentaire>Logiciel puissant et flexible pour la création de documents XML.</commentaire>
    <editeur lien="http://www.softquad.com/top_frame.sq">SoftQuad, Inc.</editeur>
    <langue>EN</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="$US">495,00</prix>
  </logiciel>
  <logiciel code="">
    <nom>Turbo XML v2.2</nom>
    <commentaire>Solution professionnelle de développement de documents XML.</commentaire>
    <editeur lien="http://216.122.205.184/solutions/turbo_xml/">Extensibility, Inc</editeur>
    <langue>EN</langue>
    <plateforme>Win/Linux/ Unix/Mac</plateforme>
    <prix monnaie="$US">269,95</prix>
  </logiciel>
  <logiciel code="">
    <nom>Xalan 1.1</nom>
    <commentaire>Un processeur XSL exécutant les recommandations du W3C en ce qui concerne XSLT et XPath.</comı
    <editeur lien="http://xml.apache.org/xalan-c/index.html">Apache Soft. Found.</editeur>
    <langue>EN</langue>
    <plateforme>Win/Linux/unix</plateforme>
    <prix monnaie="$US">00,00</prix>
  </logiciel>
  <logiciel code="">
    <nom>Xerces 1.5.0</nom>
    <commentaire>Un analyseur syntaxique XML suivant les recommandations et les standars du W3C (DOM 1.0, DOM 2.
    <editeur lien="http://xml.apache.org/xerces-c/index.html">Apache Soft. Found.</editeur>
    <langue>EN</langue>
    <plateforme>Win/Linux/Unix</plateforme>
    <prix monnaie="$US">00,00</prix>
  </logiciel>
```

```
  <logiciel code="">
    <nom>Sabblotron 0.60</nom>
    <commentaire>Un processeur XSL rapide est compact supportant les recommandations du W3C.</commentaire>
    <editeur lien="http://www.gingerall.com/charlie-bin/get/webGA/act/sablotron.act">Ginger Alliance</editeur>
    <langue>EN</langue>
    <plateforme>Win/Linux/Unix</plateforme>
    <prix monnaie="$US">00,00</prix>
  </logiciel>
  <logiciel code="">
    <nom>iXSLT Developer's Edition</nom>
    <commentaire>Un processeur XSLT conforme aux recommandations du W3C permettant de transformer des données
    <editeur lien="http://www.infoteria.com/products/product_page.jsp?id=/product/product_1.xml">Infoteria</editeur>
    <langue>EN</langue>
    <plateforme>Win/Unix</plateforme>
    <prix monnaie="$US">150,00</prix>
  </logiciel>
  <logiciel code="">
    <nom>XT Version 19991105</nom>
    <commentaire>Un processeur XSL écrit entièrement en Java.</commentaire>
    <editeur lien="http://www.jclark.com/xml/xt.html">James Clark</editeur>
    <langue>EN</langue>
    <plateforme>Win/Linux/ Unix/Mac</plateforme>
    <prix monnaie="$US">00,00</prix>
  </logiciel>
</categorie>
<categorie nom="Base de données">
  <logiciel code="13414439">
    <nom>Borland Kylix Developpement Serveur</nom>
    <commentaire>Accélérez vos developpement Web sous Linux !</commentaire>
    <editeur lien="http://www.borland.com/">Borland</editeur>
    <langue>US</langue>
    <plateforme>Win/Linux</plateforme>
    <prix monnaie="FRF">14 269,00</prix>
  </logiciel>
  <logiciel code="11404284">
    <nom>Cold Fusion Studio 4.5</nom>
    <commentaire>Pour le développement en COLD FUSION.</commentaire>
    <editeur lien="http://www.allaire.com/products/">Allaire</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">4 650,00</prix>
  </logiciel>
  <logiciel code="09600300">
    <nom>FileMaker Pro 5.0</nom>
    <commentaire>Pour une gestion de base de données facile.</commentaire>
    <editeur lien="http://www.filemaker.fr/products/index.html">File Maker Int.</editeur>
    <langue>FR</langue>
    <plateforme>Win/Mac</plateforme>
    <prix monnaie="FRF">2 478,59</prix>
  </logiciel>
  <logiciel code="13400948">
    <nom>Site Server 3.0</nom>
    <commentaire>Pour le développement d'applications Web.</commentaire>
    <editeur lien="http://www.microsoft.com/france/msdn/technologies/SiteServer/default.asp">Microsoft</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">11 730,00</prix>
  </logiciel>
  <logiciel code="233403189">
    <nom>Site Server 3.0 (Open)</nom>
    <commentaire>Pour le développement d'applications Web.</commentaire>
    <editeur lien="http://www.microsoft.com/france/msdn/technologies/SiteServer/default.asp">Microsoft</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">5 778,70</prix>
  </logiciel>
  <logiciel code="13414150">
    <nom>Visual InterDev 6.0 - Media Pack</nom>
    <commentaire>Media Pack, ne peut être vendu séparémenent.</commentaire>
    <editeur lien="http://www.microsoft.com/france/vinterdev/default.asp">Microsoft</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">184,62</prix>
  </logiciel>
</categorie>
```

```
<categorie nom="Java">
  <logiciel code="11406923">
    <nom>JBuilder 4.0 Entreprise</nom>
    <commentaire>Développement d'applications Java.</commentaire>
    <editeur lien="http://www.borland.com/">Borland</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">22 105,00</prix>
  </logiciel>
  <logiciel code="11406924">
    <nom>JBuilder 4.0 Professionnel</nom>
    <commentaire>Développement d'applications Java.</commentaire>
    <editeur lien="http://www.borland.com/">Borland</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">7 695,00</prix>
  </logiciel>
  <logiciel code="11401040">
    <nom>Visual J++ 6.0 Professionnel</nom>
    <commentaire>Tirez pleinement parti du langage Java.</commentaire>
    <editeur lien="http://www.microsoft.com/france/msdn/famille.asp">Microsoft</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">4 008,60</prix>
  </logiciel>
  <logiciel code="11413960">
    <nom>VisualAge Java Professional Edition - ver. 3.5</nom>
    <commentaire>Un outil révolutionnaire de développement Java.</commentaire>
    <editeur lien="http://www-4.ibm.com/software/ad/">IBM</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">1 165,00</prix>
  </logiciel>
</categorie>
<categorie nom="Client FTP">
  <logiciel code="13413427">
    <nom>Cute FTP 4.0 (Code de débridage)</nom>
    <commentaire>Le transfert FTP facile avec Cute FTP.</commentaire>
    <editeur lien="http://www.cuteftp.com/">Globalspace</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">321,72</prix>
  </logiciel>
  <logiciel code="13404286">
    <nom>WS_FTP Pro</nom>
    <commentaire>Monoposte Pour des transferts FTP transparents et rapides.</commentaire>
    <editeur lien="http://www.ipswitch.com/products/index.html">Ipswitch</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">417,40</prix>
  </logiciel>
  <logiciel code="">
    <nom>Fetch v 4.0</nom>
    <commentaire>Client FTP avec un interface de type exploreur, il permet le pointer-cliquer et le glisser-déposer de fichie
    <editeur lien="http://fetchsoftworks.com/">Jim Matthews</editeur>
    <langue>US</langue>
    <plateforme>Mac</plateforme>
    <prix monnaie="$US">5.00</prix>
  </logiciel>
  <logiciel code="13406578">
    <nom>Ftp Expert 2</nom>
    <commentaire>Un client FTP de nouvelle génération entièrement en français.</commentaire>
    <editeur lien="http://www.visic.com/FTPExpert/index.html">Visicom</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">249,00</prix>
  </logiciel>
  <logiciel code="">
    <nom>Ftp Expert 2 MAJ</nom>
    <commentaire>Un client FTP de nouvelle génération.</commentaire>
    <editeur lien="http://www.visic.com/FTPExpert/index.html">Visicom</editeur>
    <langue>FR</langue>
    <plateforme>Win</plateforme>
    <prix monnaie="FRF">149,00</prix>
  </logiciel>
```

```
</categorie>
<categorie nom="Serveur Web">
 <logiciel code="13413458">
  <nom>Citrix Metaframe 1.8 Entreprise 15 clt</nom>
  <commentaire>Pour déployer des applications en entreprise.</commentaire>
  <editeur lien="http://citrix.telmat-net.fr/demo/default.htm">Citrix</editeur>
  <langue>US</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">52 132,00</prix>
 </logiciel>
 <logiciel code="13406284">
  <nom>Jrun Server 3.0 Professiona 1 CPUl</nom>
  <commentaire>1 CPU Jrun Server 3.0 Professional.</commentaire>
  <editeur lien="http://www.allaire.com/products/JRun/">Allaire</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">6 460,00</prix>
 </logiciel>
 <logiciel code="13407150">
  <nom>Lotus Notes 5.0 Retail</nom>
  <commentaire>1 licence d'accès client pour Lotus Notes.</commentaire>
  <editeur lien="http://www.lotus.fr/world/france.nsf/abui/999B7080EEA107D0C12568EE0031AFA7?opendocument">Lot
  <langue>US</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">576,64</prix>
 </logiciel>
 <logiciel code="">
  <nom>Apache v 1.3.19</nom>
  <commentaire>Apache est un solide serveur Web</commentaire>
  <editeur lien="http://www.apache.org/">Apache Soft. Found.</editeur>
  <langue>US</langue>
  <plateforme>Linux</plateforme>
  <prix monnaie="FRF">00,00</prix>
 </logiciel>
 <logiciel code="">
  <nom>BIND v 9.1.2</nom>
  <commentaire>BIND est une application du service de noms de domaines (DNS)</commentaire>
  <editeur lien="http://www.isc.org/products/BIND/">Internet Soft. Cons.</editeur>
  <langue>US</langue>
  <plateforme>Linux</plateforme>
  <prix monnaie="FRF">00,00</prix>
 </logiciel>
 <logiciel code="">
  <nom>CustomDNS v 0.4</nom>
  <commentaire>CustomDNS est un serveur DNS modulaire.</commentaire>
  <editeur lien="http://customdns.sourceforge.net/">Eric Kidd</editeur>
  <langue>US</langue>
  <plateforme>Linux</plateforme>
  <prix monnaie="FRF">00,00</prix>
 </logiciel>
</categorie>
<categorie nom="Graphisme Web">
 <logiciel code="13413477">
  <nom>Fireworks 4</nom>
  <commentaire>Optimisez vos images pour le web.</commentaire>
  <editeur lien="http://www.macromedia.com/software/">Macromedia</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">2 508,69</prix>
 </logiciel>
 <logiciel code="13406152">
  <nom>Flash 5.0</nom>
  <commentaire>Créez des sites web avec animations vectorielles.</commentaire>
  <editeur lien="http://www.macromedia.com/software/">Macromedia</editeur>
  <langue>FR</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">3 087,11</prix>
 </logiciel>
 <logiciel code="13414322">
  <nom>Freehand 10</nom>
  <commentaire>L'outil idéal pour la conception graphique Web.</commentaire>
  <editeur lien="http://www.macromedia.com/software/">Macromedia</editeur>
  <langue>US</langue>
  <plateforme>Win</plateforme>
  <prix monnaie="FRF">4 170,83</prix>
```

```
    </logiciel>
    <logiciel code="">
      <nom>Gif Movie Gear 3</nom>
      <commentaire>Un logiciel pour créer des Gifs animés.</commentaire>
      <editeur lien="http://www.visic.com/GifMovieGear/index.html">Visicom</editeur>
      <langue>FR</langue>
      <plateforme>PC</plateforme>
      <prix monnaie="FRF">285,00</prix>
    </logiciel>
  </categorie>
  <categorie nom="Produits Adobe">
    <logiciel code="15205305">
      <nom>Photoshop 6.0</nom>
      <commentaire>La nouvelle version de Photoshop  en anglais.</commentaire>
      <editeur lien="http://www.adobe.com/products/main.html">Adobe Inc.</editeur>
      <langue>US</langue>
      <plateforme>Win</plateforme>
      <prix monnaie="FRF">4 979,00</prix>
    </logiciel>
    <logiciel code="15200815">
      <nom>Adobe Streamline 4.0</nom>
      <commentaire>Risque de rupture de stock , nous contacter...</commentaire>
      <editeur lien="http://www.adobe.com/products/main.html">Adobe Inc.</editeur>
      <langue>FR</langue>
      <plateforme>Win</plateforme>
      <prix monnaie="FRF">1 633,00</prix>
    </logiciel>
    <logiciel code="15214130">
      <nom>After Effects 5 Standard</nom>
      <commentaire>L'outil de pointe pour les animations.</commentaire>
      <editeur lien="http://www.adobe.com/products/main.html">Adobe Inc.</editeur>
      <langue>FR</langue>
      <plateforme>Win</plateforme>
      <prix monnaie="FRF">6 553,00</prix>
    </logiciel>
    <logiciel code="152014481">
      <nom>Illustrator 9 (Doc)</nom>
      <commentaire>La documentation d'Illustrator 9.</commentaire>
      <editeur lien="http://www.adobe.com/products/main.html">Adobe Inc.</editeur>
      <langue>FR</langue>
      <plateforme>Win</plateforme>
      <prix monnaie="FRF">499,00</prix>
    </logiciel>
  </categorie>
</logitheque>
```

27.4.3 / Intégration de fonctions PHP

Les fonctions de PHP peuvent être utilisées directement dans les feuilles de style XSLT.

En effet, **les fonctions natives de PHP ou celles du programmeur peuvent être exploitées** dans une feuille de style, en utilisant l'expression *php:function* dans un élément XSLT *xsl:value-of*. L'attribut *xmlns* doit également indiqué l'espace de noms *http://php.net/xsl* et un préfixe *php*.

```
<xsl:stylesheet
        version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:php="http://php.net/xsl">
 <xsl:template match="/">
  <xsl:value-of
      select="php:function('nomFonction',
                [liste_arguments])"/>
 </xsl:template>
</xsl:stylesheet>
```

Il faut **fournir à la fonction *php:function()* un nom de fonction et éventuellement un ou plusieurs arguments**. Ces derniers doivent être séparés par des virgules et être encadrés par l'instruction *string(...)*.

```
php:function('une fonction',
        string('arg1'), ..., string('arg2'))
```

En conséquence, il y a autant d'arguments à la fonction *php:function()* que la fonction cible en possède.

```
php:function('str_replace',
          string('cible'),
          string('remplacement'),
          string('source...'))
//équivaut à
str_replace('cible', 'remplacement', 'source...');
```

Les éléments XSLT en recèlent un particulièrement intéressant, en l'occurrence *xsl:variable*. Effectivement **l'élément *xsl:variable* est capable de contenir la valeur d'un noeud XML, qu'il sera possible de passer à une fonction PHP**. La modification de la valeur d'un noeud devient alors tout à fait réalisable par ce moyen.

```
<xsl:variable name="var" select="/chemin"/>
```

L'élément *xsl:variable* comporte deux attributs :

- **name** correspond au nom de la variable,
- **select** a pour but de sélectionner un noeud précis dans l'arborescence XML.

Une variable XSLT s'appelle dans des régles de style en faisant précéder son nom par un caractère dollar (*$var*). Dans un attribut, l'appel de la variable doit être entouré par des accolades : *{$var}*.

La méthode *registerPhpFunctions()* de l'objet *XsltProcessor* indique au processeur XSLT de prendre en compte les fonctions PHP référencées dans la feuille de style et de les exécuter lors du processus de tranformation.

```
$proc->registerPhpFunctions();
```

```
<?php
$xml = <<<XML
<?xml version="1.0" encoding="iso-8859-1" ?>
<couleurs>
  <couleur id="white">couleur blanche</couleur>
  <couleur id="red">couleur rouge</couleur>
  <couleur id="green">couleur verte</couleur>
  <couleur id="blue">couleur bleue</couleur>
  <couleur id="black">couleur noire</couleur>
</couleurs>
```

```
XML;

$xsl = <<<XSL
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsl:stylesheet
            version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            xmlns:php="http://php.net/xsl">
  <xsl:template match="/">
    <h3>Les couleurs</h3>
    <ul>
      <xsl:for-each select="//couleur">
        <xsl:variable name="var" select="."/>
        <li>
          <xsl:value-of
                select="php:function('str_replace',
                                     string('couleur '),
                                     string(''),
                                     string($var))"/>
        </li>
      </xsl:for-each>
    </ul>
    <p style="color:blue">
      <xsl:value-of
            select="php:function('horodater')"/>
    </p>
  </xsl:template>
</xsl:stylesheet>
XSL;
?>
<html>
<body>
<?php
  function horodater(){
    $ch = 'Date d\'édition : '
          . strftime("%d/%m/%Y %H:%M:%S");
    return $ch;
  }
  $doc_xsl = new DomDocument();
  $doc_xsl->loadXML($xsl);
  $doc_xml = new DomDocument();
  $doc_xml->loadXML($xml);

  $proc = new XsltProcessor();

  $proc->registerPhpFunctions();

  $xsl = $proc->importStylesheet($doc_xsl);
  $doc = $proc->transformToDoc($doc_xml);

  echo $doc->saveXML();
?>
</body>
</html>
```

27.5 / DOM XML de PHP 4.X

Le modèle d'objet de doxument (DOM) XML est désormais supporté par le langage PHP à l'aide d'un extension spécifique.

Les fonctions sont disponibles à partir du moment où PHP est **configuré avec l'option *--with-dom=[répertoire]*** et si **la librairie GNOME *xml library*** est utilisée.

La création d'un objet DOMDocument s'effectue par l'intermédiaire de la fonction *xmldoc*.

```
$chaine_XML = '<?xml version="1.0"?>'
                . '<element_racine>'
                . '  Texte dans la racine'
                . '  <noeud_enfant>Contenu textuel</noeud_enfant>'
                . '  <autre_noeud_enfant attribut="Valeur"/>
                . '</element_racine>';
$doc_xml = xmldoc($chaine_XML);
```

Il est également possible de **créer un objet DOMDocument à partir d'un fichier externe** en utilisant la fonction *xmldocfile*.

```
$fichier_xml = "fichier.xml";
$doc_xml = xmldocfile($fichier_xml);
```

Ensuite, **l'extraction d'informations relatives à l'objet DOMDocument** peuvent être réalisée par le biais de ses méthodes ou de ses propriétés.

```
// Obtenir le noeud racine
$noeud_racine = domxml_root($doc_xml);

// Obtenir les noeuds attributs
$attributs = domxml_attributes($doc_xml);

// Obtenir les noeuds enfants
$noeuds_enfants = domxml_children($doc_xml);

// retourne le nom de l'objet DOMDocument
echo $doc_xml->name;

// retourne le type de l'objet DOMDocument
echo $doc_xml->type;

// retourne le contenu textuel de l'objet DOMDocument
$tab_enfants = $doc_xml->children;
echo $tab_enfants[0]->content;
```

De la même façon, **chacune de ces méthodes et propriétés peut être appliquée à chaque noeud** extrait du document XML.

```
for ($i = 0; $i < sizeof($noeuds_enfants); $i++)
{
 $type_noeud = $noeuds_enfants[$i]->type;

 if ($type_noeud == XML_TEXT_NODE)
  echo '<h2>Contenu : ' . $noeuds_enfants[$i]->content . '<br>';
 else
 {
  $noeud = $noeuds_enfants[$j]->children;
  echo '<h2>Nom :' . $noeuds_enfants[$i]->name . '<br>';

  if ($noeud[0]->type == XML_TEXT_NODE)
   echo 'Contenu : ' . $noeud[0]->content . '<br>';
 }
 echo 'Type : ' . $type_noeud . '</h2>';

 if ($type_noeud == XML_ELEMENT_NODE)
 {
  $attributs = domxml_attributes($noeuds_enfants[$i]);
  echo '<h3>Les attributs du noeud '
       . $noeuds_enfants[$i]->name . '</h3>';
  for($j = 0; $j < sizeof($attributs); $j++)
```

```
    {
     $noeud_texte = $attributs[$j]->children;
     echo 'Nom :' . $attributs[$j]->name . '<br>'
                  . 'Type : ' . $attributs[$j]->type . '<br>'
                  . 'Contenu : ' . $noeud_texte[0]->content . '<br>';
    }
   }
}
```

La création des documents XML, de ses noeuds et de ses attributs, peut s'accomplir par des fonctions spécialisées.

```
// Création d'un document XML vide
$doc_xml_vide = domxml_new_xmldoc("1.0");

// Création du noeud racine pour le document XML
$ref_racine = domxml_add_root($doc_xml_vide, "element_racine");

// Création d'un enfant du noeud racine
$ref_enfant =
      domxml_new_child("element_enfant", "valeur du noeud");

// Création d'un attribut pour le noeud enfant
$ref_attribut = domxml_set_attribute($ref_enfant,
                                     "nom_attribut",
                                     "valeur de l'attribut");
```

La fonction *dumpmem* permet finalement de créer le document XML dans une chaîne de caractères.

```
$chaine_XML = domxml_dumpmem($doc_xml_vide);
```

Cette chaîne de caractères XML pourra être sauvegardée dans un fichier à l'aide des fonctions de système de fichiers ou directement affichée dans le navigateur du client.

```
// Création d'un fichier en lecture et écriture
$id_fichier = fopen("document.xml", "a+");
rewind($id_fichier);

// Ecriture de la chaîne XML dans le fichier
fwrite($id_fichier, $chaine_XML);

// Affichage du contenu du fichier
echo fread($id_fichier, filesize($id_fichier));

fclose($id_fichier);
```

Par ailleurs, **la fonction *xmltree* permet de transformer l'arborescence d'un document XML en un tableau PHP**.

```
$tab_xml = xmltree($doc_xml);
```

Enfin, **les expressions XPath sont supportées par le biais de deux fonctions**, l'une créant un contexte à partir du document XML et l'autre étant chargée d'évaluer l'expression puis de retourner un tableau de valeurs.

```
$obj_contexte = xpath_new_context($dom_xml);

$tab_resultat = xpath_eval($obj_contexte);
```

27.5.1 / Les types de noeuds XML

Les types de noeuds XML sont représentés par des désignations valides indiquant chaque composant du modèle d'objet du document (DOM). Le type de noeud détermine des valeurs valides et si le noeud peut avoir des noeuds enfants.

Les Types de données

Constante	Valeur
Description	
XML_ELEMENT_NODE	1
représente un noeud élément <ELEMENT>...</ELEMENT>.	
XML_ATTRIBUTE_NODE	2
représente un noeud attribut <ELEMENT ATTRIBUT="valeur"/>.	
XML_TEXT_NODE	3
représente un noeud textuel <ELEMENT>Texte</ELEMENT>.	
XML_CDATA_SECTION_NODE	4
représente une section CDATA <!CDATA[Texte non analysé]>.	
XML_ENTITY_REF_NODE	5
représente une référence d'entité.	
XML_ENTITY_NODE	6
représente une entité <, &.	
XML_PI_NODE	7
représente une instruction de traitement <?xml-stylesheet...?>.	
XML_COMMENT_NODE	8
représente un commentaire <!-- Texte -->.	
XML_DOCUMENT_NODE	9
représente un noeud document.	
XML_DOCUMENT_TYPE_NODE	10
représente une déclaration de type de document <!DOCTYPE element_racine [...]>.	
XML_DOCUMENT_FRAG_NODE	11
représente un fragment de l'arborescence d'un document.	
XML_NOTATION_NODE	12
représente une notation.	
XML_GLOBAL_NAMESPACE	1
représente un espace de nom global.	
XML_LOCAL_NAMESPACE	2
représente un espace de nom local.	

27.5.2 / Les classes *DOMDocument* et *DOMNode*

Deux classes principales sont définies dans la librairie DOM XML. Il s'agît des classes *DOMDocument* et *DOMNode*.

Ces deux classes possèdent chacunes **des méthodes et des propriétés** spécifiques permettant de récupérer ou d'ajouter des informations.

La classe *DOMDocument*

Les méthodes

Nom	Fonction
Description	
root	**domxml_root()**
retourne le noeud racine du document XML.	
children	**domxml_children()**
retourne un tableau contenant les enfants du document XML.	
add_root	**domxml_add_root()**
ajoute un nouveau noeud racine à un document XML.	
dtd	**domxml_intdtd()**
retourne la déclaration de type de document.	
dumpmem	**domxml_dumpmem()**
retourne une chaîne de caractères rerésentant les données du document XML.	
xpath_init	**xpath_init()**
initialise une expression XPath.	
xpath_new_context	**xpath_new_context()**
crée un nouveau contexte pour une expression XPath.	
xptr_new_context	**xptr_new_context()**
crée un nouveau contexte pour un pointeur.	

Les propriétés

Nom	Type
Description	
doc	class DOMDocument
retourne l'objet DOMDocument lui-même.	
name	chaîne
retourne le nom du document XML.	
url	chaîne
retourne l'adresse URL du document XML.	
version	chaîne
retourne le numéro de version de XML.	
encoding	chaîne
retourne l'encodage du document XML.	
standalone	entier long
indique si le document XML est autonome.	
type	entier long
indique le type du document XML.	
compression	entier long
indique si le fichier est compressé.	
charset	entier long
retourne le jeu de caractères du document XML.	

La classe *DOMNode*

Les méthodes

Nom	Fonction
Description	
lastchild	**domxml_last_child()**
retourne le dernier enfant du noeud courant.	
children	**domxml_children()**
retourne une liste des noeuds enfants dans un tableau.	
parent	**domxml_parent()**
retourne le noeud parent du noeud courant.	
new_child	**domxml_new_child()**
ajoute un nouvel enfant au noeud courant.	
get_attribute	**domxml_get_attribute()**
retourne la valeur d'un attribut du noeud courant.	
set_attribute	**domxml_set_attribute()**
fixe la valeur d'un attribut du noeud courant.	
attributes	**domxml_attributes()**
retourne une liste d'attributs du noeud courant.	

node	**domxml_node()**
retourne un noeud.	
set_content	**domxml_set_content()**
fixe le contenu du noeud courant.	

Les propriétés

Nom	Type
Description	
node	class DomNode
retourne l'objet DOMNode lui-même.	
type	entier long
retourne le type du noeud courant.	
name	chaîne
retourne le nom du noeud courant.	
content	chaîne
retourne le contenu du noeud courant.	

Exemple [voir]

```
<?php
 $chaine_xml = '<? xml version="1.0" ?>'
             . '<element_racine>'
             . '<element_enfant_1>Première valeur</element_enfant_1>'
             . '<element_enfant_2 attribut="valeur attribut">'
             . 'Seconde valeur'
             . '</element_enfant_2>'
             . '</element_racine>';
 $document = xmldoc($chaine_xml);

 $element_racine = $document->root();

 $noeuds = $element_racine->children();

 while ($noeud = array_shift($noeuds))
 {
  if ($noeud->name == "element_enfant_2")
  {
    $enfant = $noeud;
    break;
  }
 }

 $attribut = $enfant->getattr("attribut");

 $texte_noeud = array_shift($enfant->children());

 $valeur = $texte_noeud->content;

 echo '<u>Contenu du second élément enfant :</u> '
       . '<b>' . $texte . '</b>'
       . '<br><u>Contenu de son attribut :</u> '
       . '<b>' . $attribut . '</b>';
?>
```

27.5.3 / Les fonctions du DOM XML

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur le modèle d'objet de document XML (eXtended Markup Language).

Fonction
Description
$objet = **xmldoc**($chaine);
crée un objet DOM pour un document XML.
$objet = **xmldocfile**($fichier);
crée un objet DOM à partir d'un fichier XML.
$objet = **xmltree**($chaine);
crée un arbre d'objet PHP à partir d'un document XML.
$objet_racine = **domxml_root**($objet_dom);
retourne l'élément racine du document XML.
$objet = **domxml_add_root**($chaine);
ajoute une racine *$chaine* au document XML.
$chaine = **domxml_dumpmem**($objet_XML);
retourne l'objet XML dans une chaîne une chaîne de caractères.
$tableau = **domxml_attributes**($objet_noeud);
retourne les attributs d'un noeud XML dans un tableau.
$objet = **domxml_get_attribute**($objet_noeud, $nom_attribut);
retourne l'attribut d'un noeud.
domxml_set_attribute($objet_noeud, $nom_attribut, $valeur_attribut);
fixe le nom et la valeur d'un attribut dans un noeud XML.
$tableau = **domxml_children**($objet_noeud);
retourne les enfants d'un noeud dans un tableau.
$objet = **domxml_new_child**($nom_noeud, $contenu);
ajoute un nouvel enfant.
$objet_dom = **domxml_new_xmldoc**($version);
crée un document XML vide.
$objet_contexte = **xpath_new_context**($objet_dom);
crée un nouveau contexte xpath.
$tableau = **xpath_eval**($objet_contexte);
évalue une expression xpath.

27.5.4 / Les fonctions du DOM XML 4.3

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur le modèle d'objet de document XML (eXtended Markup Language).

L'activation du module DOM XML/XSLT sur les machines équipées de MS Windows, nécessite de copier les librairies *php_domxml.dll* et *php_xslt.dll* présentes dans le dossier *DLL* du répertoire d'installation de PHP, dans le dossier système *SYSTEM32* du répertoire d'installation du système d'exploitation Windows.

Fonction
Description
$chaine = **DomAttribute->name()**;
retourne le nom d'un attribut.
true | false = **DomAttribute->specified()**;
vérifie si l'attribut est spécifié.
$valeur = **DomAttribute->value()**;
retourne la valeur d'un attribut.
$id = **DomDocument->add_root($nom)**;
ajoute un noeud racine (déprécié).
$objet \| false = **DomDocument->create_attribute($nom, $valeur)**;
crée un noeud attribut.
$chaine \| false = **DomDocument->create_cdata_section($contenu)**;
crée un noeud CDATA.
$objet \| false = **DomDocument->create_comment($comment)**;
crée un noeud commentaire.
objet | false = **DomDocument->create_element_ns($URI, $nom[, $prefixe])**;
crée un nouveau noeud élement avec un espace de noms associé.
objet | false = **DomDocument->create_element($nom)**;
crée un noeud élément.
objet | false = **DomDocument->create_entity_reference($contenu)**;
crée une référence d'entité.
objet | false = **DomDocument->create_processing_instruction($contenu)**;
crée un noeud instruction de traîtement.
objet | false = **DomDocument->create_text_node($contenu)**;
crée un noeud textuel.
objet | false = **DomDocument->doctype()**;
retourne le type de document.
objet | false = **DomDocument->document_element()**
retourne le noeud racine.
$chaine = **DomDocument->dump_file($fichier[, $compression[, $format]])**;
décharge l'arbre XML interne dans un fichier.
$chaine = **DomDocument->dump_mem([, $format[, $encodage]])**;
décharge l'arbre XML interne dans une chaîne de caractères.

objet | false = **DomDocument->get_element_by_id($id)**;

recherche un élémentpar l'intermédiaire de son identificateur.

objet | false = **DomDocument->get_elements_by_tagname($nom)**;

retourne les éléments d'un document par l'intermédiaire de son nom.

$chaine = **DomDocument->html_dump_mem()**;

décharge l'arbre XML interne dans une chaîne de caractères comme du HTML.

$entier = **DomDocument->xinclude()**;

substitue XIncludes dans un objet DomDocument.

$tableau = **DomDocumentType->entities()**;

retourne la liste d'entités.

false | true = **DomDocumentType->internal_subset()**;

retourne le sous-ensemble interne.

$chaine = **DomDocumentType->name()**;

retourne le nom du type de document.

$tableau = **DomDocumentType->notations()**;

retourne la liste de notations.

$chaine = **DomDocumentType->public_id()**;

retourne le *public id* du type de document.

$chaine = **DomDocumentType->system_id()**;

retourne le *system id* du type d'un document.

objet | false = **DomElement->get_attribute_node($attribut)**;

retourne la valeur d'un attribut.

objet | false = **DomElement->get_attribute($nom)**;

retourne la valeur d'un attribut.

$objet | false = **DomElement->get_elements_by_tagname($nom)**;

retourne un élément par l'intermédiaire de son nom.

true | false = **DomElement->has_attribute($nom)**;

vérifie si l'attribut existe pour le noeud élément.

true | false = **DomElement->remove_attribute($nom)**;

supprime un attribut.

true | false = **DomElement->set_attribute($nom, $valeur)**;

ajoute un nouvel attribut.

$chaine = **DomElement->tagname()**;

retourne le nom d'un élément.

true | false = **DomNode->add_namespace($URI, $prefixe)**;

ajoute une déclaration d'espace de noms à un noeud.

$objet | false = **DomNode->append_child($noeud)**;

ajoute un nouvel enfant à la fin des enfants du noeud.

$objet | false = **DomNode->append_sibling($noeud)**;

ajoute le nouveau frère à un noeud.

$tableau = **DomNode->attributes()**;

retourne la liste d'attributs.

$tableau = **DomNode->child_nodes()**;

retourne les enfants du noeud.

objet | false = **DomNode->clone_node()**;

clone un noeud.

$chaine = **DomNode->dump_node()**;

décharge un noeud unique.

true | false = **DomNode->first_child()**;

retourne le premier enfant du noeud.

$chaine = **DomNode->get_content()**;

retourne le contenu du noeud.

true | false = **DomNode->has_attributess()**;

vérifie si le noeud a des attributs.

true | false = **DomNode->has_child_nodes()**;

vérifie si le noeud à des enfants.

objet | false = **DomNode->insert_before($noeud, $ref_noeud)**;

insère le nouveau noeud comme un enfant.

true | false = **DomNode->is_blank_node()**;

vérifie si le noeud est vierge.

true | false = **DomNode->last_child()**;

retourne le dernier enfant du noeud.

$objet | false = **DomNode->next_sibling()**;

retourne le prochain noeud de même parent.

$chaine = **DomNode->node_name()**;

retourne le nom du noeud.

$entier = **DomNode->node_type()**;

retourne le type de noeud.

$chaine = **DomNode->node_value()**;

retourne la valeur du noeud.

$objet | false = **DomNode->owner_document()**;

retourne le document à qui appartient le noeud.

$objet | false = **DomNode->parent_node()**;

retourne le parent d'un noeud.

$chaine = **DomNode->prefix()**;

retourne le prafixe d'espace de noms d'un noeud.

$objet | false = **DomNode->previous_sibling()**;

retourne le prochain noeud de même parent.

$objet | false = **DomNode->remove_child($ancien_enfant)**;

supprime un enfant à partir d'une liste d'enfant.

$objet | false = **DomNode->replace_child($ancien_noeud, $nouveau_noeud)**;

remplace un enfant.

$objet \| false = **DomNode->replace_node($nouveau_noeud)**;
remplace le noeud.
true | false = **DomNode->set_content($contenu)**;
fixe le contenu d'un noeud.
true | false = **DomNode->set_name($nom)**;
Sfixe le nom d'un noeud.
DomNode->set_namespace($URI[, $prefixe]);
fixe l'espace de noms d'un noeud.
$objet | false = **DomNode->unlink_node()**;
efface un noeud
$chaine = **DomProcessingInstruction->data()**;
retourne les données d'un noeud d'instruction de traîtement
$chaine = **DomProcessingInstruction->target()**;
retourne la cible d'un noeud d'instruction de traîtement.
DomXsltStylesheet->process($DOMDocument [, $parametres[, $bool_param_xpath]]);
applique une transformation XSLT sur un objet DomDocument.
$chaine = **DomXsltStylesheet->result_dump_file($DOMDocument, $fichier)**;
décharge le résultat de la transformation XSLT dans un fichier.
$chaine = **DomXsltStylesheet->result_dump_mem($DOMDocument)**;
décharge le résultat de la transformation XSLT dans une chaîne.
$objet \| false = **Domxml_new_doc($chaine)**;
crée un document XML vide.
$objet \| false = **Domxml_open_file($chaine)**;
crée un objet DOM à partir d'un fichier XML.
$objet \| false = **Domxml_open_mem($chaine)**;
crée un objet DOM à partir d'un document XML.
$chaine = **Domxml_version()**;
fournit la version de la librairie XML.
$objet \| false = **Domxml_xmltree($chaine)**;
crée un arbre d'objets PHP à partir d'un document XML.
$objet \| false = **Domxml_xslt_stylesheet_doc($DOMDocument)**;
crée un objet DomXsltStylesheet à partir d'un objet DomDocument.
$objet \| false = **Domxml_xslt_stylesheet_file($fichier_XSL)**;
crée un objet DomXsltStylesheet à partir d'un document XML dans un fichier.
$objet \| false = **Domxml_xslt_stylesheet($document_XSL)**;
crée un objet DomXsltStylesheet à partir d'un document XML dans une chaîne de caractères.
$tableau = **xpath_eval_expression($contexte)**;
évalue le chemin XPointer dans la chaîne donnée.
$tableau = **xpath_eval($contexte, $expression_xpath[, $noeud_contexte])**;
évalue le chemin XPointer dans la chaîne donnée.

$objet \| false = **xpath_new_context($DOMDocument)**;
crée un nouveau contexte XPath.
$entier = **xptr_eval([$contexte, $chaine_evaluation])**;
évalue le chemin XPointer dans la chaîne donnée.
$chaine = **xptr_new_context($gestionnaire_doc)**;
crée un nouveau contexte XPath.

27.5.4.1 / Exemples de traitement avec DOM XML

Exemple [voir]

```
<!-- Fichier : fichier.xml -->
<?xml version="1.0" encoding="iso-8859-1"?>
<liste>
  <logiciel>
    <nom langue="US" systeme_exploitation="Win">Cooktop 2.200</nom>
    <commentaire>
      Un editeur XML, XSLT, XPath et DTD puissant et totalement gratuit.
    </commentaire>
    <editeur adresse="http://xmleverywhere.com/cooktop/">
      XML Everywhere
    </editeur>
    <prix monnaie="$US">00.00</prix>
  </logiciel>
  <logiciel>
    <nom langue="US" systeme_exploitation="Win">XML Spy 4.1</nom>
    <commentaire>Un editeur XML desormais mature.</commentaire>
    <editeur adresse="http://www.xmlspy.com/default.html">
      Altova Inc.
    </editeur>
    <prix monnaie="$US">199,00</prix>
  </logiciel>
  <logiciel>
    <nom langue="US" systeme_exploitation="Win">
      XML Spy 4.1 B2B Server
    </nom>
    <commentaire>
      La version 4 en version Business to business.
    </commentaire>
    <editeur adresse="http://www.xmlspy.com/default.html">
      Altova Inc.
    </editeur>
    <prix monnaie="$US">1 999,00</prix>
  </logiciel>
  <logiciel>
    <nom langue="US" systeme_exploitation="Win">XMLwriter v1.21</nom>
    <commentaire>Permet de creer des documents XML.</commentaire>
    <editeur adresse="http://xmlwriter.net/">Wattle Software</editeur>
    <prix monnaie="$US">75,00</prix>
  </logiciel>
</liste>

<?php
// Fichier : traitement.php
  function contenu_textuel($noeud_parent)
  {
    $noeuds = $noeud_parent->children();
    while($noeud = array_shift($noeuds))
    {
      if ($noeud->type == XML_TEXT_NODE)
      {
        $resultat = $noeud->content;
        return $resultat;
      }
    }
  }

  function traitement_element($noeud_parent, $nom)
  {
    $noeuds = $noeud_parent->children();
    while($noeud = array_shift($noeuds))
    {
      if ($noeud->name == $nom)
      {
        $resultat = contenu_textuel($noeud);
        return $resultat;
      }
    }
  }

  function traitement_attribut($noeud_parent, $nom, $attribut)
  {
    $noeuds = $noeud_parent->children();
    while($noeud = array_shift($noeuds))
```

```
  {
    if ($noeud->name == $nom)
    {
      $resultat = $noeud->getattr($attribut);
      return $resultat;
    }
  }
}

$tab_elements = array("nom","commentaire","editeur","prix");
$tab_attributs = array("langue"=>0,
                       "systeme_exploitation"=>0,
                       "adresse"=>2,
                       "monnaie"=>3);

$xml_doc = xmldocfile("c:\chemin\fichier.xml")
     or die("Impossible d'ouvrir le fichier XML !");

$element_racine = $xml_doc->root();

$noeuds_enfants = $element_racine->children();

foreach($noeuds_enfants as $noeud)
{
  if($noeud->type == XML_TEXT_NODE) continue;
  for($i = 0; $i < sizeof($tab_elements); $i++)
  {
    ${$tab_elements[$i]} = traitement_element($noeud, $tab_elements[$i]);
  }

  foreach($tab_attributs as $cle=>$valeur)
  {
    $$cle = traitement_attribut($noeud, $tab_elements[$valeur], $cle);
  }

  echo "<h4>Logiciel : " . $nom
       . "</h4>Langue : " . $langue
       . "<br>Système d'exploitation : " . $systeme_exploitation
       . "<br>Commentaire : " . $commentaire
       . "<br>Editeur : " . $editeur
       . "<br>Adresse : " . $adresse
       . "<br>Prix : " . $prix . " " . $monnaie;
}
?>
```

Exemple [voir]

```
<!-- Fichier : formulaire.html -->
<html>
  <body>
  <form method="post" action="traitement.php">
     <table border="0">
       <tr>
       <td>Nom</td>
       <td><input type="text" name="e_nom" size="30"></td>
       </tr>
       <tr>
       <td>Langue</td>
       <td><input type="text" name="e_langue" size="3"></td>
       </tr>
       <tr>
       <td>Système d'exploitation</td>
       <td><input type="text" name="e_systeme_exploitation" size="5"></td>
       </tr>
       <tr>
       <td>Commentaire</td>
       <td><textarea name="e_commentaire" cols="20" rows="3"></textarea></td>
       </tr>
       <tr>
       <td>Editeur</td>
       <td><input type="text" name="e_editeur" size="20"></td>
       </tr>
       <tr>
       <td>Adresse</td>
       <td><input type="text" name="e_adresse" size="30"></td>
       </tr>
       <tr>
       <td>Prix - Monnaie</td>
       <td><input type="text" name="e_prix" size="5">
         - <input type="text" name="e_monnaie" size="3"></td>
       </tr>
       <tr>
       <td> </td>
       <td><input type="submit" name="soumettre" value="Envoyer"></td>
       </tr>
     </table>
  </form>
  </body>
</html>

<?php
// Fichier : traitement.php
  function creation_noeud($noeud_parent, $nom, $valeur)
  {
  $noeud_parent->new_child($nom, $valeur);
  return $noeud_parent->lastchild();
  }

  $tab_elements = array("nom","commentaire","editeur","prix");
  $tab_attributs = array("langue"=>0,
                  "systeme_exploitation"=>0,
                  "adresse"=>2,
                  "monnaie"=>3);

  $xml_doc = xmldocfile("c:\chemin\fichier.xml")
  or die("Impossible d'ouvrir le fichier XML !");

  $element_racine = $xml_doc->root();

  $logiciel = creation_noeud($element_racine, "logiciel", "");

  foreach($tab_elements as $element)
  {
  $nom_contenu = "e_" . $element;
  ${$element} = creation_noeud($logiciel, $element, ${$nom_contenu});
  }

  foreach($tab_attributs as $attribut=>$num_element)
  {
  $nom_contenu = "e_" . $attribut;
```

```
  ${$tab_elements[$num_element]}->set_attribute($attribut,
                                                ${$nom_contenu});
  }

  $chaine_XML = $xml_doc->dumpmem();

  $id_fichier = fopen("fichier.xml", "w+");
  rewind($id_fichier);
  fwrite($id_fichier, $chaine_XML);

  echo fread($id_fichier, filesize("fichier.xml"));

  fclose($id_fichier);
?>
```

27.6 / L'extension XSLT de PHP 4.X

Le langage XSLT (Extensible Stylesheet Language Transformation) permet de transformer des documents XML à partir de règles de modèles (*template*) en d'autres documents XML, HTML ou encore WML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" encoding="ISO-8859-1"/>
 <xsl:param name="num"/>
 <xsl:template match="/">
  <html>
   <body>
    <xsl:value-of select="logitheque/logiciel[$num]/nom"/>
    <br/>
    <xsl:value-of select="logitheque/logiciel[$num]/editeur"/>
    <br/>
    <xsl:choose>
     <xsl:when test="logitheque/logiciel[$num]/editeur/@lien != ''">
      <xsl:value-of select="logitheque/logiciel[$num]/editeur/@lien"/>
     </xsl:when>
     <xsl:otherwise>
      <xsl:message terminate="yes">
       <xsl:text>
        L'élément appelé ne contient pas d'URL !
       </xsl:text>
      </xsl:message>
     </xsl:otherwise>
    </xsl:choose>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Le module XSLT utilise **les librairies Expat et Sablotron**. Sous Unix, l'option *--with-sablot* ou *--enable-xslt --with-xslt-sablot* doivent être activées.

La création d'un analyseur XSLT s'effectue par l'entremise de la fonction *xslt_create* retournant un identifiant utilisable par d'autres fonctions.

```
$id_analyseur_XSLT = xslt_create();
```

L'application d'une feuille de style XSLT à un document XML est réalisée par le biais de la fonction *xslt_process*.

```
xslt_set_base('file://chemin_physique/');
$resultat = xslt_process($id_analyseur_XSLT,
                         $fichier_xml,
                         $fichier_xslt,
                         NULL,
                         $tab_args_xslt,
                         $tab_params_xslt);
```

Parfois, il peut être préférable de lire les données par des fonctions de fichier telles que *fread* puis de soumettre le contenu textuel à la fonction *xsl_process*.

La fonction *xsl_process* accepte également **un tableau de paramètres à passer à la feuille de style**, tandis que **le tableau d'arguments *tab_args_xslt* peut être utilisé pour passer des données XML ou/et XSLT** à la fonction *xslt_process* (voir exemple).

```
// Script PHP...
$parametres = array("nom_param"=>"valeur");
$arg = array('/_xml' => $donnee_xml, '/_xsl' => $donnee_xsl);
  ...
$resultat = xslt_process($id_analyseur_XSLT,
                         'arg:/_xml', 'arg:/_xsl',
```

```
                        NULL, $arg, $param);
...
// Feuille de style
...
  <xsl:param name="nom_param"/>
...
```

Le résultat de la transformation est directement disponible dans la variable *$resultat* qui peut être affichée à l'écran par une instruction *echo*. Si la transformation avait échoué, le contenu de cette variable serait *false*.

```
echo $resultat;
```

La fonction *xslt_process* permet de **transformer un fichier XML par une feuille de style en récupérant le résultat dans un autre fichier XML**, soit le quatrième argument de cette fonction.

```
xslt_process($id, 'fic.xml', 'fic.xsl', 'fic_result.xml');
```

Les erreurs retournées par l'analyseur XSLT sont récupérées par deux fonctions *xslt_errno* et *xslt_error* fournissant respectivement son code numérique et son message.

```
echo sprintf('Erreur :'
            . 'Code : %d'
            . 'Message : %s',
            xslt_errno($id_analyseur_XSLT),
            xslt_error($id_analyseur_XSLT));
```

L'analyseur XSLT peut être détruit par la fonction *xslt_free* au terme de son utilisation dans un script.

```
xslt_free($id_analyseur_XSLT);
```

Il est également possible de **réaliser directement des tranformations XSLT** sans créer auparavant un analyseur XSLT.

La fonction *xslt_transform* permet d'exécuter ce genre de transformation avec à l'instar de *xslt_run* la possibilité de passer des paramètres et des arguments. Cette fonction est obsolète depuis la version 4.0.3.

```
$chaine_xsl = join ('', file('fichier.xsl'));
 $chaine_xml = join ('', file('fichier.xml'));
 xslt_transform($chaine_xsl, $chaine_xml,
                "arg:/_result",
                $tab_params, $tab_args,
                $resultat);
 echo $resultat;
```

La fonction *xslt_run* accomplissait de la même façon une tranformation mais est obsolète depuis la version 4.0.3 de PHP.

```
xslt_run($id, $xsl, $xml, $resultat, $arg, $param);
```

Deux fonctions opérent une transformation de toutes les données XML situées entre elles. L'une (*xslt_output_begintransform*) annonce le début de la transformation, l'autre (*xslt_output_endtransform*) la fin. Cette fonction est obsolète depuis la version 4.0.3.

```
<?php
 $fichier_xsl = 'fichier.xsl';

 xslt_output_begintransform($fichier_xsl);
 $doc_xml = new_xmldoc('1.0');
 $poesie = $doc_xml->new_root('poesie');
 $poesie->new_child('titre', 'Locution des pierrots');
 $poesie->new_child('texte', join('',file('texte.txt')));
 $poesie->new_child('auteur', 'Jules Laforgue');
 echo $doc_xml->dumpmem();
 xslt_output_endtransform();
?>
```

27.6.1 / Les fonctions XSLT

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des transformations de documents XML par le langage XSLT.

Fonction
Description
true | false = **xslt_closelog**($reference_XSLT);
efface le fichier d'historique par rapport à une référence valide sur un analyseur XSLT (Fonction dépréciée depuis la version 4.0.3).
$identifiant_analyseur = **xslt_create**();
crée un nouvel analyseur XSLT.
$nombre = **xslt_errno**($reference_XSLT);
retourne le numéro d'erreur courant.
$resultat = **xslt_error**($reference_XSLT);
retourne le message d'erreur courant.
$chaine = **xslt_fetch_result**($reference_XSLT [, $resultat]);
lit un résultat (Fonction dépréciée depuis la version 4.0.3).
xslt_free($reference_XSLT);
libère les ressources d'un analyseur XSLT.
true | false = **xslt_openlog**($reference_XSLT, $fichier_log [, $niveau]);
modifie le fichier log (Fonction dépréciée depuis la version 4.0.3).
xslt_output_begintransform($fichier_XSLT);
commence la transformation XSLT (Fonction dépréciée depuis la version 4.0.3).
xslt_output_endtransform();
termine une transformation XSLT (Fonction dépréciée depuis la version 4.0.3).
false | $chaine_result = **xslt_process**($reference_XSLT, $donnee_XML, $donnee_XSL [, $resultat [, $args_XSLT [, $tab_param_XSLT]]]);
transforme des données XML avec les données XSL et retourne le résultat.
xslt_run($reference_XSLT, $fichier_XSLT, $fichier_XML [, $resultat [, $tab_param_XSLT [, $args_XSLT]]]);
applique une feuille de style à un fichier XML (Fonction dépréciée depuis la version 4.0.3).
xslt_set_base($reference_XSLT, $adresse_URI)
fixe l'adresse URI de base pour toutes les transformations XSLT.
xslt_set_encoding($reference_XSLT, $encodage)
fixe l'encodage pour l'analyse des documents XML.
xslt_set_error_handler($reference_XSLT, gestionnaire)
fixe un gestionnaire d'erreur pour le processeur XSLT.
xslt_set_log($reference_XSLT, $fichier_log)
fixe un fichier log pour écrire d'éventuels messages.
true | false = **xslt_set_sax_handler**($reference_XSLT, $tab_gestionnaires)
modifie les gestionnaires SAX de l'analyseur XSLT.
xslt_set_sax_handlers($reference_XSLT, $tab_gestionnaires)

modifie les gestionnaires SAX pour être appelés lorsque le document XML est traité.
xslt_set_scheme_handler($reference_XSLT, $tab_gestionnaires)
modifie les gestionnaires de Schema pour l'analyseur XSLT.
xslt_set_sax_handlers($reference_XSLT, $tab_gestionnaires)
modifie les gestionnaires de Schema pour l'analyseur XSLT.
xslt_transform($chaine_XSL, $chaine_XML, $resultat, $parametres, $arguments, $resultat_tampon);
exécute une transformation XSLT (Fonction dépréciée depuis la version 4.0.3).

27.6.2 / Exemple de traitement XSLT

Exemple [voir]

```
<!-- Fichier : fichier.xml -->
<?xml version="1.0" encoding="iso-8859-1"?>
<liste>
 <logiciel categorie="HTML">
  <nom langue="US" systeme_exploitation="Win">
   HomeSite 4.5 MAJ
  </nom>
  <commentaire>
   Mise à jour depuis v 4.0 Un éditeur HTML Professionnel.
  </commentaire>
  <editeur lien="http://www.allaire.com/products/">
   Allaire
  </editeur>
  <prix monnaie="FRF">285,00</prix>
 </logiciel>
 <logiciel categorie="HTML">
  <nom langue="FR" systeme_exploitation="Win">
   HTML Transit
  </nom>
  <commentaire>
   Uniquement sur devis : nous appeler.
  </commentaire>
  <editeur lien="http://www.intranetsolutions.com/">HTML Transit</editeur>
  <prix monnaie="FRF">31 900,00</prix>
 </logiciel>
 <logiciel categorie="HTML">
  <nom langue="FR" systeme_exploitation="Win">
   NetObjects Fusion 5.0
  </nom>
  <commentaire>
   Le logiciel le plus complet de création de site.
  </commentaire>
  <editeur lien="http://www.netobjects.com/">
   Microsoft
  </editeur>
  <prix monnaie="FRF">1 239,00</prix>
 </logiciel>
 <logiciel categorie="XML">
  <nom langue="US" systeme_exploitation="Win">
   Cooktop 2.200
  </nom>
  <commentaire>
   Un editeur XML, XSLT, XPath et DTD puissant et totalement gratuit.
  </commentaire>
  <editeur adresse="http://xmleverywhere.com/cooktop/">
   XML Everywhere
  </editeur>
  <prix monnaie="$US">00.00</prix>
 </logiciel>
 <logiciel categorie="XML">
  <nom langue="US" systeme_exploitation="Win">
   XML Spy 4.1
  </nom>
  <commentaire>
   Un editeur XML desormais mature.
  </commentaire>
  <editeur adresse="http://www.xmlspy.com/default.html">
   Altova Inc.
  </editeur>
  <prix monnaie="$US">199,00</prix>
 </logiciel>
 <logiciel categorie="XML">
  <nom langue="US" systeme_exploitation="Win">
   XML Spy 4.1 B2B Server
  </nom>
  <commentaire>
   La version 4 en version Business to business.
  </commentaire>
  <editeur adresse="http://www.xmlspy.com/default.html">
   Altova Inc.
  </editeur>
  <prix monnaie="$US">1 999,00</prix>
 </logiciel>
 <logiciel categorie="XML">
```

```
    <nom langue="US" systeme_exploitation="Win">
     XMLwriter v1.21
    </nom>
    <commentaire>
     Permet de creer des documents XML.
    </commentaire>
    <editeur adresse="http://xmlwriter.net/">
     Wattle Software
    </editeur>
    <prix monnaie="$US">75,00</prix>
   </logiciel>
</liste>
<!-- Fichier : fichier.xsl -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" media-type="text/html; charset=ISO-8859-1"/>
 <xsl:param name="categorie"/>
 <xsl:template match="/">
  <html>
   <head>
    <title>
    La logithèque : la catégorie
     <xsl:value-of select="$categorie"/>
    </title>
   </head>
   <body>
    <xsl:for-each select="/liste/logiciel[@categorie=$categorie]">
     <xsl:variable name="url" select="editeur/@adresse"/>
     <h3>
      <xsl:value-of select="nom"/>
       (<xsl:value-of select="nom/@langue"/>)
     </h3>
     <p><xsl:value-of select="commentaire"/></p>
     <h4>
      <a href="{$url}"><xsl:value-of select="editeur"/></a>
     </h4>
     <u>Prix : </u><br/>
     <p>
      <xsl:value-of select="prix"/>
      <xsl:value-of select="prix/@monnaie"/>
     </p>
    </xsl:for-each>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
<!-- Fichier : transformation.php -->
<?php
 $fxml = "fichier.xml";
 $fxsl = "fichier.xsl";
 $id_xml = fopen($fxml, "rb");
 $id_xsl = fopen($fxsl, "rb");
 $xml = fread($id_xml, filesize ($fxml));
 $xsl = fread($id_xsl, filesize ($fxsl));

 $param = array("valeur"=>"XML");
 $arg = array('/_xml' => $xml, '/_xsl' => $xsl);

 $analyseur_xslt = xslt_create();
 $resultat = xslt_process($analyseur_xslt,
                          'arg:/_xml', 'arg:/_xsl',
                          NULL, $arg, $param);
 if ($resultat != false)
 {
  print($resultat);
 }
 else
 {
  echo "<u>Une erreur est suvenue :</u>"
       . "Code : " . xslt_errno($analyseur_xslt)
       . "Message : " . xslt_error($analyseur_xslt);
 }
 xslt_free($analyseur_xslt);
```

```
?>
```

28 / Le protocole XML-RPC

Le protocole XML-RPC est un standard pour le traitement distribué sur Internet. L'appel de procédure à distance (RPC) est un mécanisme de lancement de procédures pouvant être présentes sur différents serveurs et être programmées dans divers langages de programmation.

Un message XML-RPC est une requête HTTP-POST dont le corps est écrit en XML. Une procédure s'exécute sur le serveur et la valeur retournée est également formatée en XML.

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: leprogrammeurweb.com
Content-Type: text/xml
Content-length: 182

<?xml version="1.0"?>
<methodCall>
  <methodName>methode.action</methodName>
  <params>
    <param>
      <value><i4>1010</i4></value>
    </param>
  </params>
</methodCall>
```

Les paramètres des messages XML-RPC acceptent six types de données différents.

Balise	Type de données	Exemple
<i4><int>	Nombre entier signé sur 4 octets.	780, -23
<boolean>	Valeur booléenne.	0 (false), 1 (true)
<string>	Chaîne de caractères ASCII.	'Bienvenue'
<double>	Nombre à virgule flottante en double précision et signé.	0.129657835, -89.40325
<dateTime.iso8601>	Expression temporelle au format ISO-8601.	20020228T20:51:06
<base64>	Données binaire encodées en base 64.	kf95WNb01Pht6245jHljmp21hz1

Les valeurs repérées par le balisage *<value>* peuvent être non seulement une valeur d'un type précité, mais aussi une structure *<struct>* ou encore un tableau de données *<array>*.

```
<struct>
  <member>
    <name>lowerBound</name>
    <value><i4>18</i4></value>
  </member>
  <member>
    <name>upperBound</name>
    <value><i4>139</i4></value>
  </member>
</struct>

<array>
  <data>
    <value><i4>12</i4></value>
    <value><string>Egypt</string></value>
    <value><boolean>0</boolean></value>
    <value><i4>-31</i4></value>
  </data>
</array>
```

La réponse à la requête est obtenue par le biais d'un message spécifique comportant la balise *<methodResponse>*.

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 163
Content-Type: text/xml
Date: Fri, 18 Apr 2002 16:09:54 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><double>215.50</double></value>
    </param>
  </params>
</methodResponse>
```

Si une erreur est rencontrée, alors une balisage spécial sera renvoyé avec pour valeur le code (*faultCode*) et le message (*faultString*) de l'erreur dans une structure *<struct>*.

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>1</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>Unknown method.</string></value>
        </member>
      </struct>
  </value>
  </fault>
</methodResponse>
```

Exemple [voir]

```php
<?php
/* Fichier 'page_procedure.php' sur le serveur 'leprogrammeurweb.com'. */
 include("xmlrpc.inc");
 include("xmlrpcs.inc");

 function obtenir_reponse($parametres)
 {
  global $xmlrpc_erreur;
  $premiere_valeur = $parametres->params[0];
  $valeur_scalaire = $premiere_valeur->scalarval();

  $retour = $valeur_scalaire * 20;

  return new xmlrpcresp(new xmlrpcval($retour, "int"));
 }

 $serveur = new xmlrpc_server(
     array("calcul" => array("function" => "obtenir_reponse")));
?>

<?php
// fichier disponible chez le client.
 include("xmlrpc.inc");

 if($HTTP_POST_VARS["nombre"]!="")
 {
  $message = new xmlrpcmsg('calcul',
             array(new xmlrpcval($HTTP_POST_VARS["nombre"], "int")));
  $client = new xmlrpc_client("page_procedure.php", "
                leprogrammeurweb.com", 80);
  $client->setDebug(0);
  $reponse = $client->send($message);
  $valeur = $reponse->value();
  if(!$reponse->faultCode())
  {
   echo "<p>Le nombre ". $HTTP_POST_VARS["nombre"]
       . " est " . $valeur->scalarval() . "</p>"
       . "<p>Cette valeur a été obtenue en retour</p><b>"
       . htmlentities($reponse->serialize()). "</b>";
  }
  else
  {
   echo "<u>Faute: </u><p>" . "Code: " . $reponse->faultCode()
     . "<br>Message : " . $reponse->faultString() . "</p>";
  }
 }
 echo '<h4>Saisissez un nombre :</h4>'
     . '<form method="POST">'
     . '<input type="text" size="5" value="' . ${nombre} . '"><br>'
     . '<input type="submit" value="Envoyer" name="submit">'
     . '</form>';
?>
```

28.1 / Les fonctions XML-RPC

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les documents XML avec la librairie XML-RPC (Remote Procedure Calling).

Fonction
Description
$chaine_XML = **xmlrpc_encode_request**($methode, $parametres);
génére une chaîne de caractères XML pour une requête de méthode.
$chaine_XML = **xmlrpc_encode**($valeur);
génére une chaîne de caractères XML pour une valeur PHP.
$tab_variables = **xmlrpc_decode_request**($chaine_XML, methode [, $chaine_encodage]);
décode la chaîne de caractères XML en variables PHP rassemblées dans un tableau.
$tab_types = **xmlrpc_decode**($chaine_XML [, $chaine_encodage]);
décode la chaîne de caractères XML en types PHP rassemblés dans un tableau.
$id_serveur_XMLRPC = **xmlrpc_server_create**();
crée un serveur XMLRPC et retourne un identifiant.
xmlrpc_server_destroy($id_serveur_XMLRPC);
détruit les ressources utilisées par un serveur XMLRPC.
true | false = **xmlrpc_server_register_method**($id_serveur_XMLRPC, $nom_methode, $nom_fonction);
enregistre une fonction PHP vers une méthode de la ressource.
true | false = **xmlrpc_server_register_introspection_callback**($id_serveur_XMLRPC, $nom_fonction);
enregistre une fonction PHP pour générer la documentation.
$valeur = **xmlrpc_server_call_method**($id_serveur_XMLRPC, $chaine_XML, $donnee_utilisateur [, $tab_options_sortie]);
analyse une requête XML et appelle les méthodes associées.
$nombre = **xmlrpc_server_add_introspection_data**($id_serveur_XMLRPC, $tab_donnee);
ajoute des informations d'introspection.
$tab_methodes = **xmlrpc_parse_method_descriptions**($chaine_XML);
décode la chaîne de caractères XML en une liste de descriptions de méthodes.
true | false = **xmlrpc_set_type**($chaine, $type);
modifie le type XMLRPC en base64 ou datetime vers une chaîne de caractères PHP.
$chaine = **xmlrpc_get_type**($valeur);
retourne le type XMLRPC d'une valeur PHP.

29 / Les fonctions WDDX

Le langage PHP dispose de nombreuses fonctions permettant de travailler sur les documents XML avec la librairie WDDX (Web Distributed Data eXchange).

WDDX est **un mécanisme pour l'échange complexe de structures de données** entre des environnements d'application.

```
<?xml version='1.0'?>
<!DOCTYPE wddxPacket SYSTEM 'wddx_0090.dtd'>
<wddxPacket version='0.9'>
 <header comment='Base de données des clients'/>
 <data>
 <struct>
    <var name='nom_base'>
     <string>table_personnes</string>
    </var>
    <var name='nombre_champs'>
     <number>3</number>
    </var>
    <var name='date_creation'>
     <dateTime>2001-10-01T08:30:00</dateTime>
    </var>
    <var name='tri_sur_nom'>
     <boolean value='true'/>
    </var>
    <var name='personnes'>
     <recordset rowCount='3' fieldNames='nom,prenom,sexe'>
     <field name='nom'>
       <string>MARECHEL</string>
       ...
       <string>DOGUE</string>
     </field>
     <field name='prenom'>
       <string>Jean</string>
       ...
       <string>Sylvie</string>
     </field>
     <field name='age'>
       <number>33</number>
       ...
       <number>26</number>
     </field>
     </recordset>
    </var>
  </struct>
 </data>
</wddxPacket>
```

Les paquets WDDX sont des représentations de structures de données instanciées dans des environnements d'application.

L'utilisation de WDDX nécessite l'**installation de la librairie EXPAT et la recompilation de PHP** avec *--with-xml* et *--enable-wddx*.

Fonction
Description
$chaine_WDDX = **wddx_serialize_value**($variable [, $commentaire]);
enregistre une valeur dans un paquet WDDX.
$chaine_WDDX = **wddx_serialize_vars**("variable", ..., "variable_N");
enregistre plusieurs valeurs dans un paquet WDDX.
$id_paquet_WDDX = **wddx_packet_start**($commentaire);
commence un nouveau paquet WDDX avec une structure.
$chaine_XML = **wddx_packet_end**($id_paquet_WDDX);
termine un paquet WDDX.
$chaine_XML = **wddx_add_vars**($id_paquet_WDDX, "variable", ..., "variable_N");
ajoute des variables à un paquet WDDX.
$valeur = **wddx_deserialize**($chaine_WDDX);
lit un paquet WDDX.

Exemple [voir]

```
<?php
 $variable = "Contenu de la variable";

 echo wddx_serialize_value($variable, "document sérialisé");
?>
<!-- Résultat affiché -->
<wddxPacket version='0.9'>
 <header comment='document sérialisé'/>
 <data>
  <string>Contenu de la variable</string>
 </data>
</wddxPacket>

<?php
 $nom = "PARISI";
 $prenom = "Jean-Marc";
 $information = array("date_naissance" => "24 mai 1982",
                      "adresse "=>" 145 Boulevard Raspail",
                      "cp" => 75000,
                      "ville" => "Paris",
                      "pays" => "France",
                      "tph" => "01 42 58 12 63");
 $poste = array ("Programmeur",
                 "Junior",
                 "service Développement A",
                 "jm.paris@site.com");

 echo wddx_serialize_vars("nom", "prenom", "information", "poste");
?>
<!-- Résultat affiché -->
<wddxPacket version='0.9'>
 <header/>
 <data>
  <struct>
   <var name='nom'>
    <string>PARISI</string>
   </var>
   <var name='prenom'>
    <string>24/05/1982</string>
   </var>
   <var name='info'>
    <struct>
    <var name='date_naissance'>
     <string>24 mai 1982</string>
    </var>
    <var name='adresse'>
     <string>145 Boulevard Raspail</string>
    </var>
    <var name='cp'>
     <number>75000</number>
    </var>
    <var name='ville'>
     <string>Paris</string>
    </var>
    <var name='pays'>
     <string>France</string>
    </var>
    <var name='tph'>
     <string>01 42 58 12 63</string>
    </var>
    </struct>
   </var>
   <var name='poste'>
    <array length='4'>
    <string>Programmeur</string>
    <string>Junior</string>
    <string>service Développement A</string>
    <string>jm.paris@site.com</string>
    </array>
   </var>
  </struct>
 </data>
</wddxPacket>
```

```php
<?php
 $nombre = 5;
 $capitales = array("Berlin", "Paris", "Londres", "Rome", "Madrid");

 $id_paquet_WDDX = wddx_packet_start("Capitales européennes");
 wddx_add_vars($id_paquet_WDDX, "nombre", "capitales");

 $representation = wddx_packet_end($id_paquet_WDDX);

 echo $representation;
?>
<!-- Résultat affiché -->
<wddxPacket version='0.9'>
 <header comment='PHP' >
 <data>
  <struct>
   <var name='nombre'>
    <number>5</number>
   </var>
   <var name='capitales'>
    <array length='5'>
    <string>Berlin</string>
    <string>Paris</string>
    <string>Londres</string>
    <string>Rome</string>
    <string>Madrid</string>
    </array>
   </var>
  </struct>
 </data>
</wddxPacket>
```

30 / Intégration de Java

PHP peut interagir avec Java. Un programme PHP a la capacité de profiter des potentialités de la plateforme Java. Ainsi, les paquetages, les classes, les méthodes et les attributs sont accessibles à l'aide de l'extension Java.

L'extension Java est disponible dans la bibliothèque PECL (PHP Extensions Community Library). Cette extension contient trois fichiers:

- *php_java.dll* est la bibliothèque chargée d'interfacer PHP avec la plateforme Java,
- *php_java.jar* contient une classe capable d'exécuter du code Java par introspection,
- *phpsrvlt.jar* fournit des classes chargées de gérer et exécuter des servlets Java.

Toutefois, cette extension tend à devenir obsolète. C'est pourquoi, le module PHP/Java bridge remplacera à terme l'extension expérimentale Java.

- *php_java.dll* est la bibliothèque chargée de d'établir un pont entre PHP et Java,
- *JavaBridge.jar* contient des classes chargées de gérer et d'exécuter du code Java,
- *JavaBridge.war* fournit des classes chargées de gérer et exécuter des servlets Java.

L'installation du pont Java/PHP nécessite en premier lieu son téléchargement sur le site de PHP et l'installation de la machine virtuelle Java (J2SE : Java 2 Standard Edition ou J2EE : Java 2 Enterprise Edition).

Ensuite, il faut compiler PHP avec la directive *--with-java[=Répertoire Installation]*. Le répertoire d'installation est celui du kit de développement Java (JDK : Java Development Kit). La bibliothèque *php_java.dll* doit être placée dans le répertoire */php/ext/* avec les autres extensions PHP.

La procédure d'installation pour Windows passe par la copie du fichier *php_java.dll* dans le répertoire système (*C:\WIN(DOWS|NT)\SYSTEM[32]*). Il peut être nécessaire de modifier des variables d'environnement, en l'occurrence *PATH* et *CLASSPATH*.

```
PATH = C:\J2SDK_1.4.x\bin;...
CLASSPATH = "C:\J2SDK_1.4.x\lib\php_java.jar;
        C:\J2SDK_1.4.x\lib;C:\J2SDK_1.4.x;..."
```

La modification de ces variables peut se faire par l'intermédiaire de l'onglet *Avancé* des *Propriétés* du *Poste de travail* Windows ou en ligne de commande.

Les deux fichiers Jar peuvent être placés dans le répertoire des bibliothèques Java (ex.: *J2SDK_1.4.x\lib*) ou être placé dans le répertoire des extensions PHP.

Enfin, **le fichier de configuration *php.ini* doit être modifié afin d'activer l'extension Java**. Il suffit de supprimer le caractère point-virgule devant la ligne *extension=php_java.dll*.

Dans le fichier *php.ini*, **les options de configuration de l'extension Java doivent être ajoutées** et renseignées soigneusement pour tous les systèmes d'exploitation.

```
[Java]
;Répertoires des classes de l'extension et de Java
java.classpath = "C:\JDK\lib\php_java.jar;C:\JDK\lib;C:\JDK"
;Répertoire des fichiers binaires Java
java.java_home = "C:\JDK\bin"
;Référencement du fichier jvm.dll
java.library = "C:\JDK\bin\client\jvm.dll"
;Répertoire des bibliothèques Java
java.libpath = "C:\JDK\lib"
;Exécutable Java java.exe
java.java = "C:\Dev\J2SDK_1.5.x\bin\java"
;Le niveau des messages d'erreur, par défaut 1
;0 : Journal désactivé, 4 : Journal en mode déboguage
java.log_level = 4
;Le chemin vers le fichier Journal
java.log_file = "C:\Dev\PHP\ext\JavaBridge.log"
;Le nom ou le numéro de socket
java.socketname = 9267
;Le nom de l'hôte pour l'exécution des servlets
```

```
;java.hosts = "127.0.0.1:8080"
;Activation (On) ou désactivation (Off) des servlets
;java.servlet = On
```

Maintenant, il serait opportun de **vérifier si le module Java a bien été installé** et correctement pris en compte par PHP. L'exécution de la fonction *phpinfo()* doit faire apparaître la commande de configuration *--with-java=Répertoire* dans la rubrique *Configure Command* pour les systèmes Unix, et une rubrique Java avec les directives de configuration du fichier *php.ini*.

Normalement l'extension a été parfaitement installée mais le pont Java/PHP n'a pas du être démarré. Si en testant du code, le message ci-dessous est retourné, tel est le cas.

```
Fatal error: php_mod_java(52):
Could not connect to server: No error --
Have you started the java bridge and
set the java.socketname option?
```

Pour résoudre ce problème, il faut **exécuter le fichier *JavaBridge.jar***.

```
java -jar JavaBridge.jar [INET:9267 4 JavaBridge.log]
```

Cette action a pour effet de lancer le pont Java/PHP et de créer le fichier journal spécifié dans les options de configuration du fichier *php.ini*. Les deux arguments finaux ne sont pas nécessaires s'ils sont présents dans le fichier de configuration *php.ini*.

30.1 / Utilisation du pont PHP/Java

Suite à l'instanciation d'une classe Java, les attributs et les méthodes d'instance deviennent disponibles via la syntaxe PHP. Le constructeur *java()* peut prendre une liste des arguments après le nom de la classe à instancier.

```
$chaine = new java('java.lang.StringBuffer', 'Bienvenue ');
$chaine->append('dans ');
$chaine->append('le ');
$chaine->append('monde ');
$chaine->append('Java');
$chaine->insert(0, '<p>');
$chaine->append('</p>');
$taille = $chaine->length();
echo $chaine->toString() . ' (' . $taille . ')';
```

Le constructeur *java_class()* permet d'obtenir une classe à partir de laquelle des méthodes ou des attributs de classe (ou statiques) pourront être invoquées.

```
$systeme = new java_class("java.lang.System");
$proprietes = $systeme->getProperties();
echo '<table>';
foreach ($proprietes as $cle => $valeur)
  echo '<tr><th>' . $cle
    . '</th><td>' . $valeur
    . '</td></tr>';
echo '</table>';
```

La boucle *foreach* permet de parcourir des collections Java, comme des listes (*List*), des ensembles (*Set*) ou des maps (*Map*).

Les tableaux Java profitent également de la boucle *foreach*.

```
$ch = new java('java.lang.String', $chaine->toString());
$tableau = $ch->split(' ');
echo '<ul>';
foreach($tableau as $valeur)
  echo '<i>' . $valeur . '</li/>';
echo '</ul>';
```

L'instruction *instanceof* ne pouvant vérifier que l'instance d'une classe PHP, **la fonction *java_instanceof()*** pallie à cette limititation et **se charge de vérifier si un objet Java est effectivement l'instance d'une classe Java**.

```
try {
 $dbf = new java_class(
          'javax.xml.parsers.DocumentBuilderFactory');
 $oDbf = $dbf->newInstance();
 $oDbf->setValidating(true);

 $db = new java_class('javax.xml.parsers.DocumentBuilder');
 $oDb = $oDbf->newDocumentBuilder();

 $url = new java('java.net.URL', 'file:///C:\fichier.xml');
 $flux = $url->openStream();

 $doc = $oDb->parse($flux);

 $racine = $doc->getDocumentElement();
 $nom = $racine->getTagName();
 $type = $racine->getNodeType();
 $valeur = $racine->getNodeValue();
 echo '<h3>';
 afficherInfos($nom, $type, $valeur);
 echo '</h3>';

 $liste = $racine->getChildNodes();
 parcourir($liste);
}
```

```
catch (java_exception $e) {
  '<p style="color: red">' $e->getMessage() . '</p>';
}

function parcourir($liste){
  echo '<ul>';
  if(!java_instanceof($liste,
                  new java_class('org.w3c.dom.NodeList')))
    return false;
  for($i = 0; $i < $liste->getLength(); $i++){
    $noeud = $liste->item($i);
    if(is_null($noeud) || !($noeud instanceof java))
      continue;
    $type = $noeud->getNodeType();
    $valeur = $noeud->getNodeValue();
    if(java_instanceof($noeud,
                  new java_class('org.w3c.dom.Element'))){
      $nom = $noeud->getTagName();
      afficherInfos($nom, $type, $valeur);
      if($noeud->hasChildNodes()){
        parcourir($noeud->getChildNodes());
      }
    }
    else if(java_instanceof($noeud,
                      new java_class('org.w3c.dom.Text'))){
      $nom = '#text';
      afficherInfos($nom, $type, $valeur);
    }
  }
  echo '</ul>';
}
function afficherInfos($nom, $type, $valeur){
  echo '<li>' . $nom . ' (' . $type . ') = "' . $valeur . '"</li>';
}
```

Le bloc *try...catch* rend obsolète les fonctions *java_last_exception_get()* et *java_last_exception_clear()*. En effet, il n'est plus utile d'invoquer ces méthodes puisque la gestion des exceptions en PHP est devenue suffisamment puissante pour traiter les erreurs d'exécution.

Les blocs *try...catch* peuvent encadrer tout un programme ou des parties bien délimitées pour lesquelles une exception précise doit être gérée séparément ou qu'un message particulier doit être envoyé à l'utilisateur.

La fonction *java_get_session()* permet de faire persister des variables durant l'exécution du script PHP. Normalement, les variables stockées dans la session doivent être des classes ou des instances de classe Java.

Un objet de session possède quatres méthodes.

- **isNew()** vérifie si la session vient d'être créée
- **put()** ajoute une variable de session en la référençant avec un nom,
- **get()** retourne une variable de session par rappor à son nom,
- **destroy()** détruit une session.

```
$session = java_get_session('nom_session');
if($session->isNew()) {
  $sys = new java_class('java.lang.System');
  $date = new java('java.util.Date');
  $format = new Java('java.text.SimpleDateFormat', 'dd/mm/yyyy hh:mm:ss');
  $session->put('system', $sys);
  $session->put('date', $date);
  $session->put('format', $format);
}
else {
  print($session->get('format')->format($session->get('date')));

  $session->destroy();
}
```

30.2 / Les fonctions JavaBridge

La manipulation des objets Java dans un programme PHP, s'effectue à l'aide des fonctions et attributs du module *Java/PHP Bridge*.

Fonction
Description
$objet = **new java('nom_classe')**;
instancie la classe correspondant au nom spécifié.
$reference = **new java_class('nom_classe')**;
référence la classe correspondant au nom spécifié, sans l'instancier.
java_require(fichier.jar;...;fichierN.jar'');
importe les bibliothèques Java spécifiées sous forme de liste.
java_closure($objet_PHP, 'Méthode_PHP', type);
indique à java de solliciter la méthode de l'objet PHP, au sein du programme Java.
$session = **java_get_session('nom_session')**;
crée ou récupère la session correspondant au nompassé en argument.
$booleen = **java_instanceof($objet_java, $classe_java)**;
vérifie si l'objet Java est une instance de la classe spécifiée.
$exception = **java_last_exception_get()**;
retourne la dernière exception lancée.
java_last_exception_clear();
efface la dernière exception lancée.

Attribut	Description
java_exception	représente une exception Java.