

PHP : Les chaînes de caractères

L3 Informatique - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr



Janvier 2020

Les chaînes de caractères

Importance des chaînes en Développement Web

- Constituent l'essentiel du contenu des pages Web.
- Sont manipulées pour créer des pages à partir de fichiers ou de BDD.
- Sont le type de données envoyées par formulaire.

Affichage

Avec `echo` en séparant les expressions par :

- Le point `.` (expressions concaténées).
- La virgule `,`

Avec la fonction `print()` pour inclure des expressions fonctionnelles :

- `print("Nous sommes le ".date("m.d.y' "));`

Affichage des chaînes de caractères

Syntaxe Heredoc

Permet de définir des chaînes assez longues et fonctionne comme les guillemets doubles.

heredoc.php

```
1 $p = 2;  
2 $str=<<<IDF  
3 hello 'ma' "variable" \$p $p  
4 IDF;  
5 // !!! AUCUN caractère avant et après "IDF;"  
6 echo $str;
```

Affichage des chaînes de caractères

Syntaxe Nowdoc

Permet de définir des chaînes assez longues et fonctionne comme les guillemets simples.

nowdoc.php

```
1 $p = 2;  
2 $str=<<<'IDF'  
3 hello variable \ $p $p  
4 IDF;  
5 // !!! AUCUN caractère avant et après "IDF;"  
6 echo $str;
```

Affichage formaté

A la C

```
void printf(string format, string $ch1, ...string $chN)
```

- Affiche le contenu de \$ch1,...\$chN
- Selon le format spécifié dans la chaîne `format`.

Autres fonctions (`$tab` est un tableau de chaînes) :

- `string sprintf(string format, string $ch1, ...string $chN).`
- `string vprintf(string format, array $tab).`
- `void vsprintf(string format, array $tab).`

Chaîne de formatage `format` composée :

- D'un texte personnalisé.
- Et de directives d'affichage constituées de caractères spéciaux indiquant comment les variables `string` en paramètre sont incorporées et affichées.

Directives d'affichage

Composées, dans l'ordre,

Du caractère % suivi par :

- Un caractère de remplissage (une espace par défaut ou utiliser 'r' pour caractère de remplissage r).
- Le caractère - pour un alignement du 'mot' à gauche (à droite par défaut).
- Un nombre indiquant le nombre de caractères pour la chaîne formatée.
- Un point suivi d'un entier pour fixer le nombre de décimales à afficher.
- Une lettre spécifiant le type de valeur à afficher :
 - %b pour affichage en binaire, %d en base 10, %f en flottant, %o en octal, %x ou %X en hexadécimal.
 - %c pour affichage par code ASCII.
 - %s pour affichage littéral.

Exemples

exemple4-1.php

```
1 echo "<h3>Votre facture </h3>";
2 echo sprintf("<b>%'_28s %'_22s %'_22s <br /></b>", "Prix
H.T.", "T.V.A.", "Prix T.T.C.");
3 $ht[1] = 27.55 ;
4 $ht[2] = 3450.40 ;
5 $ht[3] = 320.75 ;
6 $total=0;
7 for ($i=1; $i<4; $i++) {
8     echo "Article $i", sprintf ("%'_20.2f %'_22.2f %'_20.2f
<br />", $ht[$i], $ht[$i]*0.196, $ht[$i]*1.196);
9     $total+=$ht[$i];
10 }
11 echo str_repeat(" ", 71), "<br />";
12 echo "TOTAL", sprintf ("%'_20.2f %'_22.2f %'_20.2f <br
/>", $total, $total*0.196, $total*1.196);
```

Ordre de passage modifiable avec %n\ \$c

```
1 $sch1 = "Monsieur" ;
2 $sch2 = "Madame" ;
3 echo sprintf ("%2\ $s, %1\ $s, ...", $sch1, $sch2);
```

Longueur, Accès, Codage ASCII

```
int strlen(string)
```

Donne le nombre de caractères de la chaîne.

```
boolean empty(mixed)
```

Teste si une chaîne est vide.

Accès aux caractères avec []

Les chaînes de caractères sont assimilables à des tableaux indicés (le premier caractère est d'indice 0).

```
int ord(string l) et string chr(int c)
```

- `ord` retourne le code ASCII du caractère `l`.
- `chr` retourne le caractère de code ASCII `c`.

exemple4-2.php

```
1 for ($i=1; $i<6; $i++) {  
2   $nb=rand(65, 90);  
3   $code.=chr($nb);  
4 }  
5 echo "Votre mot de passe est : ", $code;
```


Modification de la casse

Minuscules / Majuscules / Capitalisation

- `string strtolower(string)` : tout en minuscules.
- `string strtoupper(string)` : tout en majuscules.
- `string ucfirst(string)` : 1ère lettre en majuscule.
- `string ucwords(string)` : 1ère lettre de chaque mot en majuscule.

exemple4-3.php

```
1 $nom = "ENgels" ;
2 $prenom = "jEan" ;
3 $adresse = "21, rue compointT";
4 $ville = "75018 pAris" ;
5 $mail = "ENGELS@funPHP.Com" ;
6 $prenom = ucfirst (strtolower($prenom));
7 $nom = strtoupper($nom) ;
8 $adresse = ucwords(strtolower($adresse));
9 $ville = strtoupper($ville);
10 $mail = strtolower($mail);
11 echo "Mes coordonnées <br />";
12 echo $prenom, $nom, "<br />";
13 echo $adresse, "<br />" ;
14 echo $ville, "<br />" ;
```

Suppression des caractères inutiles

```
string trim(string $ch [, charlist])
```

Supprime les caractères “inutiles” au début et en fin de `$ch` :

- Les espaces (par défaut).
- Avec `charlist`, tabulations (ASCII 9), tabulations verticales (ASCII 11), line feed (ASCII 10), retours chariot (ASCII 13).

`ltrim(...)` supprime en début de chaîne.

`rtrim(...)` supprime en fin de chaîne.

```
string wordwrap(string $ch [, int N [, string  
car [, boolean coupe]]])
```

Insère la chaîne `car` dans la chaîne `$ch` tous les `N` caractères en effectuant une césure des mots de taille $> N$ si `coupe` vaut `TRUE`.

Entités HTML et caractères spéciaux

```
string addslashes(string $ch)
```

Ajoute automatiquement le caractère d'échappement `\` devant toute occurrence de `'`, `"`, `\` et le caractère `NULL` dans `$ch`.

- Utile pour enregistrer des chaînes en BDD.
- Inutile pour les données envoyées par formulaire si la directive `magic_quotes_runtime` est activée dans `php.ini`

```
string stripslashes(string $ch)
```

Fonction inverse.

```
string quotemeta(string $ch)
```

Echappe les méta-caractères utilisés dans les regex :

`.` `\` `+` `*` `?` `[` `]` `(` `)` `$` `^`

Entités HTML et caractères spéciaux

```
string htmlspecialchars(string $ch [, int CTE  
[, string charset]])
```

Pour créer du code HTML ou XML, transforme les caractères spéciaux &, ", ', <, > en entités HTML.

- la constante `CTE` détermine la conversion des guillemets : doubles uniquement `ENT_COMPAT` (2), simples et doubles `ENT_QUOTES` (1), aucun `ENT_NOQUOTES` (0).
- `charset` est le jeu de caractères utilisés (par défaut, UTF-8).

```
string htmlspecialchars_decode(string $ch [int  
$flags])
```

Fonction inverse.

Entités HTML et caractères spéciaux

```
string htmlentities(string $ch [, ...])
```

Transforme tous les caractères éligibles (Unicode>128) en entités HTML.

```
string html_entity_decode(string $ch [, ...])
```

Fonction inverse.

```
string nl2br()
```

Transforme les sauts de ligne \n en
.

```
string strip_tags(string $ch [, string balise_ok])
```

Supprime les balises d'ouverture et fermeture des éléments HTML sauf ceux spécifiés dans `balise_ok`.

strip-tags.php

```
1 $ch = "<script> alert('Hello'); history.back();</script>";  
2 // dangereux :  
3 //echo $ch;  
4 echo strip_tags($ch);
```

Recherche de sous-chaînes

```
int substr_count(string $ch1, string $ch2)
```

Retourne le nombre d'occurrences de `$ch2` dans `$ch1`.

```
string strstr(string $ch1, string $ch2)
```

Retourne `FALSE` si `$ch2` n'apparaît pas dans `$ch1` ou sinon tous les caractères allant de la 1ère occurrence de `$ch2` dans `$ch1` jusqu'à la fin de `$ch1`.

```
string substr(string $ch, int i [, int N])
```

Retourne la sous-chaîne de `$ch` commençant à la position `i` et de longueur maximale `N`.

Recherche de sous-chaînes

```
string str_replace(string $ch1, string $ch2,  
string $ch [, string $n])
```

Remplace les occurrences de \$ch1 par \$ch2 dans \$ch. \$n passée par référence est le nombre de remplacements effectués.

Exemple4-5.php

```
1 $ch = "Perette et le pot au lait. C'est pas de pot!" ;  
2 $ssch = substr($ch, 8, 9) ;  
3 echo $ssch, "<br />" ;  
4 $ssch = substr($ch, 8) ;  
5 echo $ssch , "<br />";  
6 $ch2="pot";  
7 $nb=substr_count($ch,$ch2) ;  
8 echo "Le mot $ch2 est présent $nb fois dans $ch <br />";  
9 $ch3=str_replace('pot','bol',$ch);  
10 echo $ch3, "<br />" ;
```

Recherche de position ou d'existence d'un mot

```
int strpos(string $ch1, string $ch2 [, int p])
```

Donne la position de la chaîne `$ch2` dans `$ch1` en commençant au début de la chaîne ou à la position `p`. Retourne `FALSE` si le motif n'est pas trouvé.

strpos.php

```
1 $chn="go do you go to the togo ?";
2 $pattern="go";
3 $pos=0;
4 while (true) {
5     $pos=strpos ($chn, $pattern, $pos) ;
6     /* ATTENTION : l'usage du == au lieu du === ne donnerait pas le
7      * fonctionnement "attendu" car la position 0 serait automatiquement
8      * convertie à FALSE et le test réussirait */
9     if ($pos === false) break;
10    echo "found pattern at $pos\n";
11    ++$pos;
12 }
```


Capture de sous-chaînes dans des variables

```
array|string sscanf(string $ch, string  
"format" [, $var1,$var2,...])
```

Extrait de `$ch` et stocke dans `$var1`, `$var2` ... les éléments correspondant au format spécifié dans `format` à l'aide de spécificateurs (comme pour `printf`). Retourne le nombre d'éléments.

exemple4-7.php

```
1 $personne = "1685-1750 Jean-Sébastien Bach";  
2 $format="%d-%d %s %s";  
3 $nb = sscanf($personne,$format,$ne,$mort,$prenom,$nom);  
4 echo "$prenom $nom né en $ne, mort en $mort <br />";  
5 echo "Nous lisons $nb informations";
```

Comparaison de chaînes

==

Appliqué à une chaîne et un nombre, convertit la chaîne en nombre (ses premiers caractères numériques sont retenus) pour une comparaison numérique.

Conversion également appliquée en cas d'opérations arithmétiques (+, -, *, /, %) combinant chaîne et nombre.

Comparaison de chaînes

<, >, <= et >=

Compurent les chaînes lexicographiquement selon le code ASCII.

Fonctions alternatives :

`int strcmp(string $ch1, string $ch2) :` sensible à la casse.

`int strcasecmp(string $ch1, string $ch2) :` insensible à la casse.

strcmp.php

```
1 $ch1 = "Blanc"; $ch2 = "Bleu"; $ch3 = "blanc";  
2 echo strcmp($ch1,$ch3) . "<br/>"; // affiche 0  
3 echo strcmp($ch1,$ch2) . "<br/>"; // affiche -4  
4 echo strcmp($ch1,$ch3) . "<br/>"; // affiche -32
```

Transformations de chaînes en tableaux

```
array explode(string sep, string $ch [, int  
N])
```

Décompose la chaîne `$ch` en tableau de mots (sous-chaînes) selon le séparateur `sep` fourni. `N` est le nombre maximum de mots recherchés.

```
string implode(string sep, array $tab)
```

Fonction inverse.

Transformations de chaînes en tableaux

Exemple4-8.php

```
1 //Passage chaîne -> tableau
2 $sch1="L'avenir est à PHP7 et MySQL";
3 $stab1=explode(" ", $sch1);
4 echo $sch1, "<br />";
5 print_r($stab1);
6 echo "<hr />";
7 $sch2="C:\\wampserver\\www\\php7\\chaines\\string2.php";
8 $stab2=explode("\\", $sch2);
9 echo $sch2, "<br />";
10 print_r($stab2);
11 echo "<hr />";
12 //Passage tableau -> chaîne
13 $stab3[0]="Bonjour";
14 $stab3[1]="monsieur";
15 $stab3[2]="Rasmus";
16 $stab3[3]="Merci!";
17 $sch3=implode(" ", $stab3);
18 echo $sch3, "<br />";
```

Les expressions régulières

Expression régulière (alias motif, masque, regex, modèle)

Chaîne de caractères qui décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles.

- Motif d'adresse e-mail.
- Motif de numéro de téléphone.
- Motif de code IBAN ...

Encadrement des motifs avec / et /

/motif/ pour la regex motif.

Caractères spéciaux (alias méta-caractères)

A l'extérieur des [] :

- \ + * ? [] () \$ ^

A l'intérieur des [] :

- \ - ^

Motifs élémentaires

Recherche de chaînes précises

- `/angers/` : `angers` apparaît dans la chaîne analysée.
- `/\.fr/` : `.fr` apparaît.
- `/angers|\.fr/` : `angers` ou `.fr` apparaît.

Recherche d'un ou plusieurs caractères

- `/[axz]/` : l'un des caractères `a`, `x`, `z`.
- `/[b-p]/` : l'une des minuscules entre `b` et `p`.
- `/[A-Z]/` : une majuscule.
- `/[0-9]/` : un chiffre.

Recherche de méta-caractères

A échapper avec l'antislash.

Classes de caractères prédéfinies

Classe	Recherche
<code>[[:alnum:]]</code>	Tous les caractères alphanumériques : <code>[a-zA-Z0-9]</code> .
<code>[[:alpha:]]</code>	Tous les caractères alphabétiques : <code>[a-zA-Z]</code> .
<code>[[:blank:]]</code>	Tous les caractères blancs : espaces, tabulations ...
<code>[[:ctrl:]]</code>	Tous les caractères de contrôle.
<code>[[:digit:]]</code>	Tous les chiffres : <code>[0-9]</code>
<code>[[:print:]]</code>	Tous les caractères imprimables sauf caractères de contrôle.
<code>[[:punct:]]</code>	Tous les caractères de ponctuation.
<code>[[:space:]]</code>	Tous les caractères d'espace : espaces, tabulations, sauts de ligne, ...
<code>[[:upper:]]</code>	Tous les caractères en majuscules : <code>[A-Z]</code> .
<code>[[:xdigit:]]</code>	Tous les caractères en hexadécimal.

Restriction de caractères

Avec \wedge entre crochets

- $/[\wedge axz]/$: un caractère différent de a , x , z .
- $/[\wedge A-Z]/$: un caractère qui n'est pas une majuscule.

Début et fin de chaîne :

Avec \wedge hors crochets

- $/\wedge axz/$: toute chaîne démarrant par axz .

Avec $\$$

- $/axz\$/$: toute chaîne finissant par axz .

Motifs généraux

N'importe quel caractère

Avec `.` (non échappé)

- `/a.z/` : toute chaîne contenant `a` suivi d'un caractère suivi de `z`.

Répétition de caractères

Avec `?` (0 ou 1 fois), `+` (au moins une fois), `*` (0 ou plus)

- `/ab?/` : présence d'un `a` non suivi d'un `b` ou suivi d'un seul `b`.
- `/ab+ /` : présence d'un `a` suivi d'une suite d'au moins un `b`.
- `/ab*/` : présence d'un `a` suivi ou non d'une suite de `b`.

Motifs généraux

Sous-motifs

Regroupement avec $()$

- $/(ab)^+ /$: toute chaîne contenant une série de ab .

Contraintes de cardinalité

Avec $\{n\}$ (n répétitions)

- $/(ab)^{\{5\}} /$: série de 5 ab .

Avec m, n (entre m et n répétitions) :

- $/(ab)^{\{3, 5\}} /$: série de 3 à 5 ab .

Avec m , (au moins m répétitions) :

- $/(ab)^{\{3, \}} /$: série d'au moins 3 ab .

Les fonctions de recherche

```
bool preg_match(string $motif, string $ch [,  
array $tab])
```

Recherche une sous-chaîne de `$ch` satisfaisant la regex `$motif`. Retourne `TRUE` le cas échéant, `FALSE` sinon. `$tab` est un tableau indicé contenant `$ch` comme premier élément puis toutes les sous-chaînes satisfaisant à `$motif` comme éléments suivants.

preg-match.php

```
1 $sch1="30-19-1970";  
2 $sch2="4-01-18";  
3 $motif="/([0-9]{1,2})-([0-9]{2})-([0-9]{2,4})/i";  
4 $result=array();  
5 preg_match($motif,$sch1,$result);  
6 print_r($result); //Array([0] => 30-19-1970 [1] => 30 [2] => 19 [3] =>  
1970)  
7 preg_match($motif,$sch2,$result);  
8 print_r($result); //Array([0] => 4-01-18 [1] => 4 [2] => 01 [3] => 18)
```

Les fonctions de recherche

```
string preg_replace(string $motif, string  
$rep, string $ch)
```

Remplace toute occurrence du motif `$motif` dans `$ch` par `rep` et retourne la chaîne résultat.

preg-replace.php

```
1 $chn="30-09-1970";  
2 $motif="/(\\d+)-(\\d+)-(\\d{2,4})/i";  
3 $remplace="$3/$2/$1";  
4 $str=preg_replace($motif,$remplace,$chn);  
5 echo "chaîne de départ : $chn\\n";  
6 echo "chaîne résultat : $str\\n";
```

Autres fonctions pour les expression régulières

- `preg_split`
- `preg_grep`
- `preg_filter`
- `preg_match`
- `preg_match_all`