PHP : Les chaînes de caractères

L2 MPCIE - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2019

Les chaînes de caractères

Importance des chaînes en Développement Web

- Constituent l'essentiel du contenu des pages Web.
- Sont manipulées pour créér 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('d"));`

Affichage des chaînes de caractères

Syntaxe Heredoc

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

heredoc.php

```php
$p = 2;
$str=<<<IDF
hello variable \$p $p
IDF;
// !!! AUCUN caractère avant et après "IDF;"
echo $str;
```

Manipulation des chaînes de caractères

Syntaxe Nowdoc

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

nowdoc.php

```php
$p = 2;
$str=<<<'IDF'
hello variable \$p $p
IDF;
// !!! AUCUN caractère avant et après "IDF;"
echo $str;
```

Affichage formatté

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 formattage `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 à droite (à gauche par défaut).
- Un nombre indiquant le nombre de caractères pour la chaîne formattée.
- 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

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

Ordre de passage modifiable avec `%n\$c`

```php
$ch1 = "Monsieur" ;
$ch2 = "Madame" ;
echo sprintf ("%2\$s, %1\$s, ...",$ch1,$ch2);
```

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 chaîne 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

```php
$nom = "ENgelS" ;
$prenom = "jEan" ;
$adresse = "21, rue compoinT";
$ville = "75018 pAris" ;
$mail = "ENGELS@funPHP.Com" ;
$prenom = ucfirst (strtolower($prenom));
$nom = strtoupper($nom) ;
$adresse = ucwords(strtolower($adresse));
$ville = strtoupper($ville);
$mail = strtolower($mail);
echo "Mes coordonnées <br />";
echo $prenom, $nom, "<br />";
echo $adresse, "<br />" ;
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-catactères utilisés dans les regex : `.`, `\`, `+`, `*`, `?`, `[`, `]`, `(`, `)`, `$` et `^`.

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 `<br/>`.

`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

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

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

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

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 ($+$, $-$, $\star$, $/$, $\%$) combinant chaîne et nombre.

Comparaison de chaînes

<, >, <= et >=

Comparent 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

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

```php
//Passage chaîne –> tableau
$ch1="L'avenir est à PHP7 et MySQL";
$tab1=explode(" ",$ch1);
echo $ch1,"<br />";
print_r($tab1);
echo "<hr />";
$ch2="C:\wampserver\www\php7\chaines\string2.php";
$tab2=explode("\\",$ch2);
echo $ch2,"<br />";
print_r($tab2);
echo "<hr />";
//Passage tableau –> chaîne
$tab3[0]="Bonjour";
$tab3[1]="monsieur";
$tab3[2]="Rasmus";
$tab3[3]="Merci!";
$ch3=implode(" ",$tab3);
echo $ch3,"<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)

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

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

Restriction de caractères

Avec ^ entre crochets

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

Début et fin de chaîne :

Avec ^ hors crochets

- `/^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

```php
$ch1="30-19-1970";
$ch2="4-01-18";
$motif="/([0-9]{1,2})-([0-9]{2})-([0-9]{2,4})/i";
$result=array();
preg_match($motif,$ch1,$result);
print_r($result); //Array([0] => 30-19-1970 [1] => 30 [2] => 19 [3] => 1970)
preg_match($motif,$ch2,$result);
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

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

Autres fonctions pour les expression régulières

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