Tableaux

TP 02 du Module 02 – Les bases du langage PHP

Proposition de solution

- Exercice 1

```php
<?php
$gerard[] = "gérard";
$gerard[] = "paris";
$gerard[] = 67;
$aurelie[] = "aurélie";
$aurelie[] = "nantes";
$aurelie[] = 31;
$personnes["dupont"] = $gerard;
$personnes["badin"] = $aurelie;

var_dump($personnes);
```

L'instruction `$gerard[] = "gérard";` crée un tableau et positionne dans la case d'indice 0 la valeur « gérard ». La seconde instruction est semblable mais le tableau existe à présent. Cette instruction ajoute une nouvelle valeur au tableau dans la case d'indice 1. Les six premières instructions créent deux tableaux à une seule dimension. Les deux instructions suivantes permettent de mettre ces deux tableaux dans un tableau. C'est une manière de créer un tableau multidimensionnel en PHP.

Exécution

```
C:\wamp64\www\CoursPHP\1-bases\tp2\tp2q1.php:11:
array (size=2)
  'dupont' =>
    array (size=3)
      0 => string 'gérard' (length=7)
      1 => string 'paris' (length=5)
      2 => int 67
  'badin' =>
    array (size=3)
      0 => string 'aurélie' (length=8)
      1 => string 'nantes' (length=6)
      2 => int 31
```

- Exercice 2

```php
<?php
$personnes2 = [
    "dupont" =>
    ["prénom" => "gérard", "ville de résidence" => "paris", "âge" => 67],
    "badin" =>
    ["prénom" => "aurélie", "ville de résidence" => "nantes", "âge" => 31]];

var_dump($personnes2);
```

Il aurait tout à fait été possible de faire quelque chose de similaire à la solution proposée pour l'exercice précédent. Il aurait suffi d'ajouter entre les crochets des six premières instructions les clés adéquates.

La solution proposée ici se base sur la création et l'initialisation d'un tableau en une seule instruction avec `array()` ou sa version plus courte `[]`.

Exécution

```
C:\wamp64\www\CoursPHP\1-bases\tp2\tp2q2.php:5:
array (size=2)
  'dupont' =>
    array (size=3)
      'prénom' => string 'gérard' (length=7)
      'ville de résidence' => string 'paris' (length=5)
      'âge' => int 67
  'badin' =>
    array (size=3)
      'prénom' => string 'aurélie' (length=8)
      'ville de résidence' => string 'nantes' (length=6)
      'âge' => int 31
```

- Exercice 3

Tableau de l'exercice 1

```php
<?php
require 'tp2q1.php';
echo '<ul>';
foreach ($personnes as $nom => $valeurs) {
    echo '<li>Element ' . $nom . '<ul>';
    for ($i = 0; $i < count($valeurs); $i++) {
        echo '<li>élément ' . $i . ' : ' . $valeurs[$i] . ' </li>';
    }
    echo '</ul></li>';
}
echo '</ul>';
```

L'instruction `require 'tp2q1.php';` fait appel au code réalisé pour l'exercice 1 afin que le tableau soit créé et initialisé.

L'instruction `echo '<ul>';` débute une liste à puces.

Le `foreach ($personnes as $nom => $valeurs)` va parcourir toutes les lignes du tableau `$personnes`. Successivement les variables `$nom` et `$valeurs` vont valoir respectivement la clé et la valeur de la première ligne puis de la seconde jusqu'à la dernière ligne.

L'instruction `echo '<li>Element ' . $nom . '<ul>';` positionne un élément dans la liste et commence une nouvelle liste. Ceci permet de faire un second niveau dans la liste.

La boucle `for ($i = 0; $i < count($valeurs); $i++)` parcours chaque élément du tableau. La fonction `count()` permet de connaître la taille de ce tableau.

Exécution

```
C:\wamp64\www\CoursPHP\1-bases\tp2\tp2q1.php:11:
array (size=2)
  'dupont' =>
    array (size=3)
      0 => string 'gérard' (length=7)
      1 => string 'paris' (length=5)
      2 => int 67
  'badin' =>
    array (size=3)
      0 => string 'aurélie' (length=8)
      1 => string 'nantes' (length=6)
      2 => int 31
```

- Element dupont
  - élément 0 : gérard
  - élément 1 : paris
  - élément 2 : 67
- Element badin
  - élément 0 : aurélie
  - élément 1 : nantes
  - élément 2 : 31

Tableau de l'exercice 2

```php
<?php
require 'tp2q2.php';
echo '<ul>';
foreach ($personnes2 as $nom => $valeurs) {
    echo "<li>Element $nom<ul>";
    foreach ($valeurs as $clé => $valeur) {
        echo "<li>$clé : $valeur</li>";
    }
    echo "</ul></li>";
}
echo '</ul>';
```

Exécution

```
C:\wamp64\www\CoursPHP\1-bases\tp2\tp2q2.php:8:
array (size=2)
  'dupont' =>
    array (size=3)
      'prénom' => string 'gérard' (length=7)
      'ville de résidence' => string 'paris' (length=5)
      'âge' => int 67
  'badin' =>
    array (size=3)
      'prénom' => string 'aurélie' (length=8)
      'ville de résidence' => string 'nantes' (length=6)
      'âge' => int 31
```

- Element dupont
  - prénom : gérard
  - ville de résidence : paris
  - âge : 67
- Element badin
  - prénom : aurélie
  - ville de résidence : nantes
  - âge : 31

- Exercice 4

```php
<?php
$mails = ['jean@eni.fr', 'fred@linux.net', 'lea@renault.fr', 'caroline@eni.fr
',
    'contact@eni-ecole.fr', 'valentina@ferrari.it', 'melanie@eni-ecole.fr',
    'philippe@eni.fr', 'typhaine@belfort.fr', 'louis@leparisien.fr'];
foreach ($mails as $m) {
    $d = explode('@', $m)[1];
    if (isset($domaines[$d])) {
        $domaines[$d] ++;
    } else {
        $domaines[$d] = 1;
    }
}
var_dump($domaines);
```

L'instruction `$d = explode('@', $m)[1]` récupère le nom de domaine du mail :

- La fonction `explode()` découpe une chaîne de caractères en fonction d'un délimiteur choisi (@ dans ce cas). Chaque partie est positionnée dans une case du tableau retourné.
- Seule la seconde partie est conservée car c'est le nom de domaine qui est recherché. Pour cela, le `[1]` permet d'accéder à la seconde case du tableau obtenu.

Il est à noter l'absence de vérification que l'ensemble des mails soient bien formés (présence du @). Cette vérification pourra être ajoutée après avoir pris connaissance de la séquence vidéo suivante.

Exécution

```
C:\wamp64\www\CoursPHP\1-bases\tp2\tp2q4.php:13:
array (size=7)
  'eni.fr' => int 3
  'linux.net' => int 1
  'renault.fr' => int 1
  'eni-ecole.fr' => int 2
  'ferrari.it' => int 1
  'belfort.fr' => int 1
  'leparisien.fr' => int 1
```

- Exercice 5

```php
<?php
for ($i = 1; $i <= 63; $i++) {
    $t1[] = $i;
}
$t2[] = 0;
foreach ($t1 as $val) {
    $t2[] = $val / 10;
}

foreach ($t2 as $reel) {
    $t3[(string) $reel] = sin($reel);
}

echo '<table style="border-
collapse:collapse"><tr><th style="border:1px solid black">x</th><th style="bo
rder:1px solid black">sin(x)</th></tr>';
foreach ($t3 as $x => $sinx) {
    echo '<tr><td style="border:1px solid black">' . $x . '</td><td style="bo
rder:1px solid black">' . $sinx . '</td></tr>';
}
echo '</table>';
```

La première boucle crée un tableau dont les valeurs sont tous les entiers compris entre 1 et 63.

Pour le second tableau, la première case est créée individuellement avec l'instruction `$t2[] = 0;`. Les autres cases de ce tableau sont créées en prenant les valeurs du premier tableau et en les divisant par dix.

Chaque valeur du second tableau est utilisée pour remplir le troisième tableau. Le tableau est un tableau associatif :

- La clé est la valeur transformée en chaîne de caractères. Une clé ne peut être qu'un entier ou une chaîne de caractères. Si le cast en chaîne `(string)` n'est pas utilisé, la valeur sera automatiquement transformée en entier. Dans ce cas-là, la valeur 0.1 serait convertie en 0 et viendrait écraser la case `$t3[0]`.
- La valeur associée à cette clé est la valeur du second tableau à laquelle la fonction sinus (`sin()`) a été appliquée.

La fin du script permet l'affichage d'un tableau au format HTML en parcourant le troisième tableau.

Exécution

x	sin(x)
0	0
0.1	0.099833416646828
0.2	0.19866933079506
0.3	0.29552020666134
0.4	0.38941834230865
0.5	0.4794255386042
0.6	0.56464247339504
0.7	0.64421768723769
0.8	0.71735609089952
0.9	0.78332690962748
1	0.8414709848079

1.1	0.89120736006144
1.2	0.93203908596723
1.3	0.96355818541719
1.4	0.98544972998846
1.5	0.99749498660405
1.6	0.99957360304151
1.7	0.99166481045247
1.8	0.9738476308782
1.9	0.94630008768741
2	0.90929742682568
2.1	0.86320936664887
2.2	0.80849640381959
2.3	0.74570521217672
2.4	0.67546318055115
2.5	0.59847214410396
2.6	0.51550137182146
2.7	0.42737988023383
2.8	0.33498815015591
2.9	0.23924932921398
3	0.14112000805987
3.1	0.04158066243329
3.2	-0.05837414342758
3.3	-0.15774569414325
3.4	-0.25554110202683
3.5	-0.35078322768962
3.6	-0.44252044329485
3.7	-0.52983614090849
3.8	-0.61185789094272
3.9	-0.68776615918397
4	-0.75680249530793
4.1	-0.81827711106441
4.2	-0.87157577241359
4.3	-0.91616593674945
4.4	-0.95160207388952
4.5	-0.9775301176651
4.6	-0.99369100363346
4.7	-0.9999232575641
4.8	-0.99616460883584
4.9	-0.98245261262433
5	-0.95892427466314
5.1	-0.92581468232773
5.2	-0.88345465572015
5.3	-0.8322674422239
5.4	-0.77276448755599
5.5	-0.70554032557039
5.6	-0.63126663787232
5.7	-0.55068554259764
5.8	-0.46460217941376
5.9	-0.37387666483024
6	-0.27941549819893
6.1	-0.1821625042721
6.2	-0.083089402817496
6.3	0.01681390048435