La syntaxe du langage PHP

Lydia Rodriguez--de la Nava
CNAM
Juin 2024

Le PHP dans le HTML

La page HTML statique

Vous vous souvenez sûrement (enfin j'espère !!) de la syntaxe d'une page HTML :

```
<html>
        <head>
                <meta charset="utf-8" />
                <title>Une page web</title>
        </head>
        <body>
                <h1> Un grand titre ! </h1>
                <p> Aujourd'hui on est le mercredi 12 juin 2024. </p>
        </body>
</html>
```

- la page est statique
- c'est-à-dire qu'elle affiche tout le temps la même chose : il est tous les jours le 12/06/2024 !

Une page dynamique

On peut écrire du code PHP directement dans le HTML, dans une balise `<?php ?>`. Par exemple:

```
<html>
        <head>
                <meta charset="utf-8" />
                <title>Une page web</title>
        </head>
        <body>
                <?php echo "<h1> Un grand titre </h1>" ?>
                <p> Aujourd'hui on est le mercredi 12 juin 2024. </p>
        </body>
</html>
```

- Avec `echo`, on écrit une chaîne de caractère directement dans le HTML
- PHP est un langage de programmation, donc on peut utiliser : des variables, des `if`, des boucles, des fonctions... pour rendre notre page dynamique

Exemple : Afficher la date et l'heure actuelle

```
<html>
        <head>
                <meta charset="utf-8" />
                <title>Une page web</title>
        </head>
        <body>
                <h1> Un grand titre </h1>
                <p> Date : <?php echo date("d-m-y");  ?> </p>
                <p> Heure : <?php echo date("h-i-s"); ?> </p>
        </body>
</html>
```

- `date` est une fonction de base dans PHP qui retourne une date sous forme de chaîne selon le format qu'on passe en argument
- pour voie sa documentation, cf https://www.php.net/manual/fr/function.date.php

Les bases

Les variables

Les variables PHP commencent toujours par le symbole `$` :

```php
<?php
        $ma_variable = 2;

        $une_chaine = "bonjour";

        $pi = 3.14;
?>
```

On affecte une valeur à une variable avec un `=`, de la même façon qu'on le faisait en C.

Type des variables

Comme vous pouvez le voir, les variables ont un **typage implicite**, c'est-à-dire qu'on n'a pas besoin de préciser que `$ma_variable` est un entier :

```php
<?php
        // pour l'instant $ma_variable n'a pas de type
        // (ie. elle a été déclarée, mais pas initialisée)
        $ma_variable;

        // maintenant $ma_variable est un entier
        // (ie. la variable a été initialisée)
        $ma_variable = 2;

        // maintenant $ma_variable est une chaîne !
        $ma_variable = "du texte";
?>
```

Une variable a donc le droit de changer de type au cours de son existences

Afficher une variable

Imaginons que j'ai mis dans une variable l'adresse du CNAM parce que je sais que je vais l'utiliser à plusieurs occasions dans ma page. Alors je peux l'afficher, en incluant ce code à l'endroit de mon choix dans code HTML :

```
<?php
        $adresse = "2 rue Conté";

        echo "L'adresse du CNAM est $adresse";
?>
```

J'affiche l'adresse, et je pourrais la réutiliser plus tard, sans avoir à la réécrire (parce que j'ai la flemme !).

Tester le type d'une variable

Puisqu'une variable peut changer de type en cours de route, il est parfois utile de vérifier qu'une variable est bien du type que l'on avait prévu :

```
$a = 2;

is_integer($a); // renvoie `true`
is_string($a); // renvoie false
```

- is_bool() - renvoie `true` si une variable est un booléen
- is_float() - renvoie `true` si une variable est de type nombre décimal
- is_numeric() - renvoie `true` si une variable est un nombre ou une chaîne numérique
- is_string() - renvoie `true` si une variable est de type chaîne de caractères
- is_array() - renvoie `true` si une variable est un tableau
- is_object() - renvoie `true` si une variable est de type objet

Les opérateurs

Pas de grande nouveauté au niveau des opérateurs, ils sont globalement similaires à ceux utilisés en C :

- **Arithmétique** : `+` `-` `*` `/` `%` `++` `--`
- **Affectation** : `=` `+=` `-=` `*=` `/=` `%=`
- **Comparaison** : `==` `<` `<=` `>` `>=` `!=`
- **Logique** : `&&` `||` `!`
  - on peut aussi utiliser `and` à la place de `&&` et `or` à la place de `||`
- **Opérateur ternaire** : `condition ? instruction1 : instruction2`

Des nouveaux opérateurs

- `===` et `!==` : fait un comparaison *stricte*, c'est-à-dire qu'on vérifie que les variable sont du même type ET ont la même valeur
  - par exemple `5 == "5"` renvoie `true`, alors que `5 === "5"` renvoie `false`
- La concaténation : pour concaténer deux chaînes de caractère, on a les opérateur `.` et `.=`
  - par example `"Bon" . "jour"` renvoie `"Bonjour"`
  - si j'ai `$a = "Bon"`, alors `$a .= "jour"` renvoie `"Bonjour"`

Les conditions

Le typique `if ... else`

Le fonctionnement est le même que ce qu'on a déjà vu : si une condition est vraie, on exécute le code à l'intérieur du if.

```
<?php
        if ( cond1 ) {
                // du code qui s'exécute si cond1 est `true`
        } elseif ( cond2 ){
                // du code qui s'exécutz si cond2 est `true`
        } else {
                // du code qui s'exécutera si cond1 et cond2 sont `false`
        }
?>
```

On voit une petite différence avec le C puisqu'ici on a un nouveau mot clé `elseif` (pour rappel en C on doit faire `else if`).

Le `switch case`

```php
<?php
        switch ( $cas ) {
                case CAS1:
                        echo "c'est le premier cas";
                        break;
                case CAS2:
                        echo "c'est le deuxième cas";
                        break;
                case CAS3:
                        echo "c'est le troisième cas";
                        break;
                default:
                        echo "ce n'est aucun cas";
        }
?>
```

Le `switch case` (suite)

équivaut à faire :

```php
<?php
        if ($cas == CAS1) echo "c'est le premier cas";
        elseif ($cas == CAS2) echo "c'est le deuxième cas";
        elseif ($cas == CAS3) echo "c'est le troisième cas";
        else echo "ce n'est aucun cas";
?>
```

Les boucles

La boucle `for`

Même utilisation qu'en C :

```
<?php
        for ($i=0; $i < 10; i++) {
                echo $i;
        }
?>
```

Le code doit afficher tous les entiers de `0` à `9`.

La boucle `while`

Toujours la même utilisation qu'en C :

```php
<?php
        $i = 0;

        while ($i < 10) {
                echo $i;
                $i++;
        }
?>
```

La boucle `do while`

Similaire à la boucle `while` sauf que la condition est testée à la fin d'une itération (ie. à la fin d'un tour de boucle).

```
`<?php
        $i = 0;
        do {
                echo $i;
                $i++;
        } while ($i < 10);
?>`
```

On entre donc obligatoirement au moins une fois dans la boucle, alors qu'avec le `while`, si la condition est fausse dès le début, on n'entre jamais à l'intérieur.

La boucle `foreach`

Une nouveauté qui permet d'itérer plus simplement sur les tableaux :

```php
<?php
        $arr = array(1, 2, 3, 4);

        foreach ($arr as $value) {
                $value = $value * 2;
        }
?>
```

On reviendra dessus plus tard lorsqu'on parlera des tableaux.

Les tableaux

Un tableau, vraiment ?

Ce qu'on appelle un tableau en PHP (ou un `array`) est en réalité différent d'un tableau en C.

On a l'habitude des tableaux où on accède à ses valeurs grâce à son indice : la première case a pour indice `0`, la deuxième a pour indice `1`, etc...


tab-en-c.excalidraw

Les `map` : associer une valeur à une clé

Un tableau en PHP est en réalité ce qu'on appelle un `map`, ou une carte en français.

Cela permet de l'utiliser comme un *tableau*, une *liste*, *une table de hachage*, un *dictionnaire*, une *pile*, une *file*....

Le principe est simple en réalité : à chaque case du tableau, on associe une **clé** qui permet d'y accéder, et cette clé peut être de n'importe quel type.

Création d'un tableau

```php
<?php
        $arr = array(
                "l" => "lundi",
                "ma" => "mardi",
                "me" => "mercredi",
                "j" => "jeudi",
                "v" => "vendredi",
                "s" => "samedi",
                "d" => "dimanche"
        );

        echo $arr["ma"]; // affiche mardi
?>
```

À gauche on a les **clés** et à droite les **valeurs** qui leur correspondent.

Un tableau comme en C

Par défaut, si on ne précise pas explicitement une clé à la création d'un tableau, on obtient un tableau comme en C. C'est-à-dire que les clés sont des entiers, en commençant par `0` et en ajoutant `1` à chaque fois.

```php
$arr = array(5, 10, 3, 9, 1);
```

équivaut à faire

```php
$arr = array(
        0 => 5,
        1 => 10,
        2 => 3,
        3 => 9,
        4 => 5
);
```

Exemple d'utilisation : Combien ai-je de fruits ?

Imaginons que je veuille faire un tableau qui me permet de compter combien de fruit j'ai dans mon panier à fruit, je vais créer un tableau ainsi :

```php
$fruits = array(
        "pomme" => 3,
        "kiwi" => 2,
        "banane" => 10,
        "fraise" => 300
);
```

Si mange une banane, je peux garder mon compte ainsi :

```php
$fruits["banane"]--;
```

Itérer sur un array

C'est ici qu'on se rend compte qu'on a besoin de la boucle `foreach` qui a été mentionnée précédemment.

Puisqu'on ne peut pas forcément simplement itérer sur une variable `$i`, on préfère utiliser `foreach` :

```
        foreach ($fruits as $f) {
                echo $f;
        }
```

Ce code affiche les *valeurs* de mon tableau fruit, c'est-à-dire qu'à chaque tour de boucle, la variable `$f` change de valeur pour prendre celle de la case actuelle.

Dans la première itération `$f` vaut `3`, dans la deuxième `$f` vaut 2, dans la troisième `$f` vaut `9` et dans la quatrième `$f` vaut `300`.

Itérer sur un array (suite)

Si je veux afficher le nom des fruits et leur quantité, je peux aussi récupérer la clé de chaque case :

```
        foreach ($fruits as $nom => $quantite) {
                echo $nom . " : " . $quantite;
        }
```

va afficher

```
pomme : 3,
kiwi : 2,
banane : 9,
fraise : 300
```

Vérifier l'existence d'une variable

Si je veux vérifier qu'il y a une clé `"poire"` dans mon tableau `$fruits`, on peut vérifier l'existence d'une variable grâce à la fonction `isset()` :

```php
<?php
        if (isset($fruits["poire"])) {
                echo "Les poires existent dans mon tableau";
        }
        else {
                echo "Les poires n'existent pas dans mon tableau";
        }
?>
```

Ajouter une valeur à mon tableau

S'il n'y a pas de poire dans mon tableau, je peux le faire très simplement :

```php
$fruits["poire"] = 0;
```