L'objet DateTime

La classe `DateTime`

Nous allons voir une classe dans cette leçon, mais nous n'avons pas encore étudié la programmation orientée objet.

Ne vous concentrez pas sur les notions relatives à cette dernière, nous allons la voir en détail dans plusieurs chapitres dédiés.

Créer un objet date

Pour créer un objet date, il suffit de passer en premier argument une chaîne date / heure (par défaut ce sera `now`) et éventuellement un fuseau horaire en deuxième argument.

Par exemple :

```
<pre>
<?php
print_r(new DateTime());
/* DateTime Object
(
    [date] => 2021-04-30 17:28:18.780698
    [timezone_type] => 3
    [timezone] => UTC
) */
```

La chaîne date / heure peut être dans tous les formats que nous avons vu dans le chapitre, voici quelques exemples :

```
<?php
new DateTime('yesterday');
new DateTime('-5 days');
new DateTime('-1 week');
new DateTime('2021-02-30');
new DateTime('yesterday');
new DateTime('2021-05-01 12:30:12');
```

Formater un objet `DateTime`

Pour formater un objet `DateTime` il suffit d'utiliser la méthode `format()`.

Nous verrons en détail les méthodes dans un chapitre ultérieur.

Quelques exemples :

```
<?php
$date1 = new DateTime('yesterday');
echo $date1->format('d-m-Y'), '<br>'; // 29-04-2021

$date2 = new DateTime('-5 days');
echo $date1->format('Y-m-d H:i:s'), '<br>'; // 2021-04-29 00:00:00

$date3 = new DateTime('-1 week');
echo $date1->format('g:ia o
 l jS F Y'), '<br>'; // 12:00am on Thursday 29th April 2021
```

A noter qu'il n'est pas possible d'afficher une date localement.

La liste des formats supportés est disponible ici.

Comparer des objets `DateTime`

Pour comparer des objets `DateTime`, vous pouvez utiliser les opérateurs de comparaison :

```
<pre>
<?php
$audj = new DateTime('today');
$hier = new DateTime('yesterday');

var_dump($hier > $audj); // bool(false)
var_dump($hier < $audj); // bool(true)
var_dump($hier == $audj); // bool(false)
```

Pour obtenir la différence exacte entre deux dates, vous pouvez **utiliser la méthode `diff()`** :

```
<pre>
<?php
```

```
$audj = new DateTime('today');
$hier = new DateTime('yesterday');

print_r($audj->diff($hier));
/* DateInterval Object
(
    [y] => 0
    [m] => 0
    [d] => 1
    [h] => 0
    [i] => 0
    [s] => 0
    [f] => 0
    [weekday] => 0
    [weekday_behavior] => 0
    [first_last_day_of] => 0
    [invert] => 1
    [days] => 1
    [special_type] => 0
    [special_amount] => 0
    [have_weekday_relative] => 0
    [have_special_relative] => 0
) */
```

Vous aurez un objet `DateInterval` contenant la différence précise entre les deux dates.

Il suffit ensuite d'afficher la différence en utilisant la méthode `format()` que nous avons vu, par exemple :

```
<pre>
<?php
$date1 = new DateTime('today noon');
$date2 = new DateTime('-3 days -5 hours');

$diff = $date1->diff($date2);
echo $diff->format('%d jours %H heures'); // 2 jours 22 heures
```

Modifier des objets `DateTime`

Vous pouvez ajouter ou soustraire une durée à un objet `DateTime` avec la méthode `modify()`.

Par exemple :

```
<pre>
<?php
$date1 = new DateTime('today noon');
  $dans3Semaines = $date1->modify('+3 weeks');
echo $dans3Semaines->format('d/m/Y'); // 21/05/2021
```

Définir le fuseau horaire

Vous pouvez passer en deuxième argument un objet `DateTimeZone` pour définir le fuseau horaire de la date :

```
<pre>
<?php
$date1 = new DateTime('today noon', new DateTimeZone('Europe/Paris'));
echo $date1->format('d/m/Y G:i:s e'), '<br>'; // 30/04/2021 12:00:00
Europe/Paris
echo $date1->format('d/m/Y G:i:s P'); // 30/04/2021 12:00:00 +02:00
```

Le `+2` signifie ici +2 heures par rapport à `GMT`.