18 DE NOVIEMBRE DE 2023

ESTUDIO CLASE DATETIME

ÁLVARO CORDERO MIÑAMBRES
IES LOS SAUCES

CLASE DATETIME

INTRODUCCION

Esta clase nos sirve para representar la fecha y la hora.

- **Documentación oficial →** https://www.php.net/manual/es/class.datetime.php
- **Versión PHP →** PHP 5 >= 5.2.0, PHP 7, PHP 8

SINOPSIS DE LA CLASE

Esta clase implementa la clase DateTimeInterfaces (https://www.php.net/manual/es/class.datetimeinterface.php)

Constantes más usadas

- **DateTime::ATOM →** Representa un formato de fecha y hora específico según el estándar ISO 860.

  Corresponde al formato de fecha y hora en el siguiente formato:

  - YYYY-MM-DDTHH:MM:SS±hh:mm.

    - 'YYYY' representa el año con cuatro dígitos.
    - MM representa el mes con dos dígitos.
    - DD representa el día con dos dígitos.
    - T es el separador entre la fecha y la hora.
    - HH representa la hora con dos dígitos (formato de 24 horas).
    - MM representa los minutos con dos dígitos.
    - SS representa los segundos con dos dígitos.
    - ±hh:mm representa la diferencia con la hora UTC en horas y minutos.

  Por ejemplo, la fecha y hora actual en el formato DateTime::ATOM se vería algo así: 2023-11-18T15:30:00+00:00. Esta cadena representa el 18 de noviembre de 2023 a las 15:30:00 en el tiempo coordinado universal (UTC).

  Puedes usar esta constante al trabajar con la clase DateTime en PHP para formatear fechas y horas según el estándar ISO 8601. Por ejemplo:

```php
$now = new DateTime();
$formattedDate = $now->format(DateTime::ATOM);
echo $formattedDate;
```

- **DateTime::COOKIE →** Representa un formato de fecha y hora, pero en este caso sigue el formato de fecha y hora definido por el estándar de cookies de HTTP.

  Tiene la siguiente estructura:

  - l, d-M-Y H: i:s T

    - 'l' representa el día de la semana en el idioma de la configuración regional.
    - 'd' representa el día del mes con dos dígitos.
    - 'M' representa el nombre corto del mes en el idioma de la configuración regional.
    - 'Y' representa el año con cuatro dígitos.
    - 'H' representa la hora en formato de 24 horas.
    - 'i' representa los minutos con dos dígitos.
    - 's' representa los segundos con dos dígitos.
    - 'T' representa la zona horaria abreviada.

  Por ejemplo, la fecha y hora actual en el formato DateTime::COOKIE se vería algo así: "Saturday, 18-Nov-2023 15:30:00 UTC".

  Puedes usar esta constante de la siguiente manera:

```php
$now = new DateTime();
$formattedDate = $now->format(DateTime::COOKIE);
echo $formattedDate;
```

  Esto imprimirá la fecha y hora actual en el formato DateTime::COOKIE. Este formato es útil cuando necesitas mostrar fechas y horas en un formato fácilmente legible por humanos y compatible con las convenciones de las cookies HTTP.

Métodos mas usados

- **DateTime::Constructor →** El constructor de la clase DateTime tiene varias formas de ser utilizado, pero la forma más común es la siguiente:

  - public DateTime::__construct([string $datetime = "now" [, DateTimeZone $timezone = NULL]])

    Aquí hay una breve explicación de los parámetros:

    - $datetime: Un string que representa la fecha y hora. Puede ser en varios formatos válidos de fecha y hora, como "now" para la fecha y hora actual, o en el formato "YYYY-MM-DD HH:MM:SS". Si este parámetro se omite, se asumirá "now" de manera predeterminada.

    - $timezone: Un objeto DateTimeZone que representa la zona horaria en la que se debe interpretar la fecha y hora. Si se omite, se utilizará la zona horaria predeterminada del sistema.

  - Aquí hay algunos ejemplos de cómo puedes utilizar el constructor de la clase DateTime:

    - Crear un objeto DateTime con la fecha y hora actuales en la zona horaria predeterminada del sistema:

      ```php
      $now = new DateTime();
      ```

    - Crear un objeto DateTime para una fecha y hora específicas:

      ```php
      $customDate = new DateTime("2023-11-18 15:30:00");
      ```

    - Especificar una zona horaria diferente:

      ```php
      $customDateWithTimeZone = new DateTime("2023-11-18 15:30:00", new DateTimeZone("America/New_York"));
      ```

- **DateTime::add()** → Este método agrega un objeto DateInterval al objeto DateTime. Un DateInterval representa un intervalo de tiempo, como una cantidad específica de días, horas, minutos, etc.

```php
public DateTime::add(DateInterval $interval)
```

  - Aquí hay algunos ejemplos de cómo puedes utilizar este método

```php
$now = new DateTime("2023-11-18 15:30:00");
$interval = new DateInterval("P1D"); // Agrega un día
$now->add($interval);
echo $now->format("Y-m-d H:i:s"); // Mostrará la fecha y hora d
```

  - En este ejemplo, se crea un objeto DateTime para el 18 de noviembre de 2023 a las 15:30:00. Luego, se crea un objeto DateInterval que representa un intervalo de tiempo de un día ("P1D"). Finalmente, el método add() se utiliza para agregar ese intervalo al objeto DateTime. El resultado se imprime con format() y mostrará la nueva fecha y hora después de agregar un día.

  - Puedes ajustar el DateInterval según tus necesidades para agregar diferentes cantidades de días, horas, minutos, etc.

- **DateTime::setTimeZone()** → Establece la zona horaria para el objeto DateTime

```
public DateTime::setTimezone(DateTimeZone $timezone): DateTime
```

  - Aquí hay una breve explicación de los parámetros:

    - Object → Solamente para el estilo por procedimientos: Un objeto DateTime devuelto por date_create(). La función modifica este objeto.

    - Timezone → Un objeto DateTimeZone que representa la zona horaria deseada.

- Devuelve el objeto DateTime para la cadena de métodos o false en caso de error.

- Aquí hay algunos ejemplos de uso

```php
<?php
$fecha = new DateTime('2000-01-01', new DateTimeZone('Pacific/Nauru'));
echo $fecha->format('Y-m-d H:i:sP') . "\n";

$fecha->setTimezone(new DateTimeZone('Pacific/Chatham'));
echo $fecha->format('Y-m-d H:i:sP') . "\n";
?>
```

- **DateTime::setTimeStamp()** → Establece la fecha y la hora basándose en una marca temporal de Unix

```php
public DateTime::setTimestamp(int $unixtimestamp): DateTime
```

- Aquí hay una breve explicación de los parámetros:

  - Object → Solamente para el estilo por procedimientos: Un objeto DateTime devuelto por date_create(). La función modifica este objeto.

  - Unixtimestamp→ La marca temporal de Unix que representa la fecha.

- Devuelve el objeto DateTime para la cadena de métodos o false en caso de error.

- Ejemplos de uso.

```php
<?php
$fecha = new DateTime();
echo $fecha->format('U = Y-m-d H:i:s') . "\n";

$fecha->setTimestamp(1171502725);
echo $fecha->format('U = Y-m-d H:i:s') . "\n";
?>
```

- **DateTime::format()** → Devuelve la fecha formateada según el formato dado

```
public DateTime::format(string $format): string
```

  - Aquí hay una breve explicación de los parámetros:
    - Object → Solamente para el estilo por procedimientos: Un objeto DateTime devuelto por date_create()
    - format→ Formato aceptado por date().
  - Devuelve la fecha formateada en caso de éxito o false en caso de error.
  - Ejemplo de uso

```php
<?php
$date = new DateTime('2000-01-01');
echo $date->format('Y-m-d H:i:s');
?>
```

- **DateTime::getLastErrors()** → Devuelve un array con las advertencias y los errores encontrados mientras se analizaba una cadena de fecha/hora.

```
public static DateTime::getLastErrors(): array|false
```

  - Esta función no tiene parámetros.
  - Devuelve un array que contiene información acerca de las advertencias y los errores, o false si no hay advertencias ni errores.
  - Ejemplo de uso

```php
<?php
try {
    $fecha = new DateTime('asdfasdf');
} catch (Exception $e) {
    // Sólo con propósitos de demostración...
    print_r(DateTime::getLastErrors());

    // La forma real orientada a objetos de hacer esto es
    // echo $e->getMessage();
}
?>
```