Manejo de fechas con PHP

Heike Bonilla Redondo

Índice

1. MANEJO DE FECHAS CON PHP

Las fechas se pueden expresar con diferentes formatos, esto hace que no se fácil tratarlas. En PHP no hay un tipo de dato fecha, se tratan como cadenas e internamente como un entero, lo que se llama fecha UNIX.

En PHP podemos hacerlo de dos maneras:

- **Utilizando el grupo de funciones time, date, getdate, checkdate, strtotime, mktime, …**
- **Utilizando la clase para manejo de fechas y horas DATETIME**

Nos vamos a verlo de las dos manera aunque centraremos el posterior uso a la clase DateTime.

Esta librería trae consigo unos cuantos métodos sumamente interesantes. Entre ellas:

- **DateTime (Constructor)**
- **createFromFormat**
- **modify**
- **format**
- **date_diff**
- **date_add y date_sub**

2. Algunas funciones de manejo de fechas usadas en PHP

time()

Lo más adecuado en todos los casos suele ser almacenar la fecha en formato unix en una variable:

```
$fecha = time();
echo $fecha;
```

De este modo guardamos en la variable $fecha todos los segundos que han transcurrido desde las doce de la noche del 1 de enero de 1970. $fecha será, por tanto, un número entero.

date()

Para sacar partido a los datos recogidos con la función time() tenemos la función "date()" para que muestre la fecha u hora en el formato que nos interese:

```
$fecha_con_formato = date("formato",$fecha);
echo $fecha_con_formato;
```

Los parámetros más importantes que podemos introducir en "formato" son los siguientes:

Parámetro	Descripción	Parámetro	Descripción
d	Día del mes con ceros iniciales (De 01 a 31).	h	Hora en formato 12-horas con ceros iniciales.
j	Día del mes sin ceros iniciales (De 1 a 31).	H	Hora en formato 24-horas con ceros iniciales.
m	Número del mes con ceros iniciales (De 01 a 12).	i	Minutos con ceros iniciales.
n	Número del mes sin ceros iniciales (De 1 a 12).	s	Segundos con ceros iniciales.
Y	Año con cuatro dígitos (Ejemplo: 2007).	A	AM o PM.
y	Año con dos dígitos (Ejemplo: 07).	T	Configuración de zona horaria del servidor.

Puedes ver aquí una lista de todos los disponibles).

Ejemplo: Si es el 19 de diciembre de 2012 y son las 12:25, el siguiente código:

```
$fecha_con_formato = date("d/m/Y -- H:i:s",$fecha);
```

nos devolverá:

19/12/2012-- 12:25:33

mktime(hora, min, seg, mes, día, año)

Devuelve la marca de tiempo Unix correspondiente a los argumentos dados. Esta marca de tiempo es un entero que contiene el número de segundos entre la Época Unix (1 de Enero del 1970 00:00:00 GMT) y el instante especificado.

Los argumentos pueden omitirse de derecha a izquierda; cualquier argumento que se omita será establecido al valor actual según la fecha y hora locales.

```php
<?php
echo date("M-d-Y", mktime(0, 0, 0, 12, 32, 1997));
echo date("M-d-Y", mktime(0, 0, 0, 13, 1, 1997));
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 1998));
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 98));
?>
```

Devolverá:

Jan-01-1998

Jan-01-1998

Jan-01-1998

Jan-01-1998

strtotime

strtotime (string $time [, int $now = time()]) : int

time

Una cadena de fecha/hora. Los formatos válidos se explican en Formatos de fecha y hora.

now

La marca de tiempo que se usa como base para el cálculo de las fechas relativas.

Convierte una descripción de fecha/hora textual a una fecha Unix

Algunas consideraciones a tener en cuenta:

Si el número del año se especifica en un formato de dos dígitos, los valores entre 00-69 hacen referencia a 2000-2069 y 70-99 a 1970-1999. Vea las notas de abajo para las posibles diferencias en sistemas de 32 bit (las fechas posibles podrían terminar en 2038-01-19 03:14:07).

Las fechas en los formatos *m/d/y* o *d-m-y* no son ambiguas al observar el separador entre los distintos componentes:

- si el separador es una barra (*/*), se asume el formato norteamericano *m/d/y*
- si el separador es un guion (-) o un punto (.), se asume el formato europeo *d-m-y*.
- si, el año se proporciona en un formato de dos dígitos y el separador es un guion (-), la cadena de la fecha se analiza como *y-m-d*. crearía un conflicto por lo que recomienda el uso del formato YYYY-MM-DD

```php
echo "Now ".strtotime("now")  ."<br>";
//Si la fecha es anterior a 1970 el entero devuelto será negativo
echo "Año anterior a 1970 ".strtotime("10 September 1950")."<br>";
/*Si el número del año se especifica en un formato de dos dígitos,
 *los valores entre 00-69 hacen referencia a 2000-2069 y 70-99 a 1970-1999.
 */
echo date(("d/m/Y"), strtotime("10 September 1967"))  ."<br>";
echo date(("d/m/Y"), strtotime("10 September 1923"))  ."<br>";
echo date(("d/m/y"), strtotime("+1 day"))."<br>";
echo date(("d/m/y"), strtotime("+1 week, now")) ."<br>";
echo strtotime("+1 week 2 days 4 hours 2 seconds") ."<br>";
echo strtotime("next Thursday")."<br>";
echo strtotime("last Monday")."<br>";
```

checkdate

bool checkdate (int `$month` , int `$day` , int `$year`)

Comprueba la validez de una fecha formada por los argumentos. Una fecha se considera válida si cada parámetro está propiamente definido.

```
<?php
    $fecha_valida = checkdate(2, 29, 2012);
    if ($fecha_valida=='true')
            echo 'ok';
    else
       echo 'no';
?>
```

Devolverá ok porque 2012 fue bisiesto

getdate

array **getdate** ([int `$timestamp` = time()])

Devuelve un array asociativo que contiene la información de la fecha de *`timestamp`*, o el momento local actual si no se da *`timestamp`*.

"seconds"	*Representacion numérica de los segundos*	*0 a 59*
"minutes"	*Representacion numérica de los minutos*	*0 a 59*
"hours"	*Representacion numérica de las horas*	*0 a 23*
"mday"	*Representacion numérica del día del mes*	*1 a 31*
"wday"	*Representacion numérica del día de la semana*	*0 (para Domingo) hasta 6 (para Sábado)*
"mon"	*Representacion numérica de un mes*	*1 hasta 12*
"year"	*Una representacion numérica completa de una año, 4 dígitos*	*Ejemplos: 1999 o 2003*
"yday"	*Representacion numérica del día del año*	*0 hasta 365*
"weekday"	*Una representación textual completa del día de la semana*	*Sunday hasta Saturday*
"month"	*Una representación textual completa de un mes, como January o March*	*January hasta December*
0	*Los segundos desde la Época Unix, similar a los valores devueltos por time() y usados por date().*	*Dependiente del Sistema, típicamente -2147483648 hasta 2147483647.*

```
<?php
$hoy = getdate();
print_r($hoy);
?>
```

Devolverá

Array ([seconds] => 25 [minutes] => 12 [hours] => 23 [mday] => 19 [wday] => 6 [mon] => 1 [year] => 2013 [yday] => 18 [weekday] => Saturday [month] => January [0] => 1358633545)

EJEMPLOS COMUNES DE MANEJO DE FECHAS CON FUNCIONES PHP

Calcular una fecha después de un plazo

Vamos a calcular cuál será la fecha en un plazo de 2 meses de hoy. Los pasos son los siguientes:

```
<?php
    $hoy = getdate();
    $dos_meses_mas_tarde = mktime(12, 0, 0, $hoy['mon']+2, $hoy['mday'], $hoy['year']);
    echo (date ("d-m-y", $dos_meses_mas_tarde));
?>

Devolverá una fecha dos meses después de la actual.
```

Vamos a calcular la fecha actual 90 días después de la fecha actual:

```
<?php

    $today = getdate();

    $month = $today['mon'];
    $day = $today['mday'];
    $year = $today['year'];

    $garantia_expira = mktime(12, 0, 0, $month, $day+90, $year);

    echo (date ("d-m-Y", $garantia_expira));
?>
```

Vamos a tomar hoy, por ejemplo, es 20 enero de 2012. La operación aritmética que se llevará a cabo es de 20 + 90, que será igual a 112. La función tiene en cuenta esto y devolverá: 20-04-2013

Calcular una edad

Un ejemplo de cómo calcular las fechas entre 2 fechas son las siguientes:

```
<?php

    $month = 28;
    $day = 8;
    $year = 1985;

    $creation_utime = mktime(0,0,0, $month, $day, $year);
    $now_utime = time();

    $age_utime = $now_utime - $creation_utime;
    $product_age = floor($age_utime / (365 * 24 * 60 * 60));

    echo 'El producto fue hecho hace '. $product_age .' año(s) atrás';

?>
```

3. TRATAMIENTO DE FECHAS EN PHP CON DateTime

Esta librería trae consigo unos cuantos métodos sumamente interesantes. Entre ellas:

- **DateTime (Constructor)**

- createFromFormat
- modify
- format
- date_diff
- date_add y date_sub

DateTime (Constructor)

Representa la fecha y hora en formato OO.

Existen funciones alias para poder utilizarla en programación procedural (date_create)

Estilo orientado a objetos

public **DateTime::__construct** ([string `$time` = "now" [, DateTimeZone `$timezone` = **NULL**]])
Emite una Exception en caso de error

```
try {
    $fecha = new DateTime("3/10/2019");
} catch (Exception $e) {
    echo $e->getMessage();
    exit;
}
var_dump ($fecha);
echo $fecha->format('d-m-Y HH:mm');

/* Algunas fechas válidas
 * $date = new DateTime('2015-06-19');
 * $date = new DateTime('+10 days'); // 10 días posteriores a la hora actual
 * $date = new DateTime('Yesterday'); Ayer
 */
```

Estilo por procedimientos

date_create ([string `$time` = "now" [, DateTimeZone `$timezone` = **NULL**]]) : DateTime
Devuelve false en caso de error

```
$fecha = date_create('Tomorrow');

var_dump($fecha);

if ($fecha)
    date_format($fecha, 'Y-m-d');
```

En ambos casos devuelve objeto DateTime

Parámetro $time contiene una cadena en formato fecha/hora válido

Parámetro $timezone, por defecto toma la zona que tiene configurada el servidor.

createFromFormat

Analiza una cadena con un instante según un formato especificado, devuelve un objeto DateTime a partir del instante introducido o false en el caso de fallo.

Si queremos crear la fecha 32/12/2019 creará el día del calendario válido según este “fecha” introducida, en este caso sería 1/1/2020.

Estilo orientado a objetos

public static **DateTime::createFromFormat** (string `$format` , string `$time` [, DateTimeZone `$timezone`]) : DateTime

```
$fecha = DateTime::createFromFormat('j-M-Y', '15-Feb-2009');
echo $fecha->format('Y-m-d');
```

Estilo por procedimientos

date_create_from_format (string `$format` , string `$time` [, DateTimeZone `$timezone`]) : DateTime

```
$fecha = date_create_from_format('j-M-Y', '15-Feb-2009');
echo date_format($fecha, 'Y-m-d');
```

modify

Altera la marca temporal de un objeto DateTime aumentando o disminuyendo en un formato aceptado por los formatos relativos.

Devuelve el objeto DateTime modificado o false en caso de error.

Estilo orientado a objetos

public **DateTime::modify** (string `$modify`) : DateTime

```
$fecha = new DateTime('2006-12-12');
$fecha->modify('+1 day');
echo $fecha->format('Y-m-d');
```

Estilo por procedimientos

date_modify (DateTime `$object` , string `$modify`) : DateTime

```
$fecha= date_create('2006-12-12');
date_modify($fecha, '+1 day');
echo date_format($fecha, 'Y-m-d');
```

format

Devuelve una cadena que contiene la fecha según el formato dado o false en caso de error. El formato aceptado es el mismo que el aceptado por date.

Estilo orientado a objetos

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

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

Estilo por procedimientos

date_format (DateTimeInterface `$object` , string `$format`) : string

```
$date = date_create('2000-01-01');
echo date_format($date, 'Y-m-d H:i:s');
```

date_diff

Devuelve un intervalo resultado de la diferencia entre dos objetos fecha o false en caso de error. El resultado será un objeto de la clase DateInterval. El resultado podemos formatearlo para mostrarlo con DateInterval::format

Estilo orientado a objetos

public **DateTime::diff** (DateTimeInterface `$datetime2` [, bool `$absolute` = **FALSE**]) : DateInterval

```
$datetime1 = new DateTime('2009-10-11');
$datetime2 = new DateTime('2009-10-13');
$interval = $datetime1->diff($datetime2);
echo $interval->format('%R%a días');
```

Estilo por procedimientos

date_diff (DateTimeInterface `$datetime1` , DateTimeInterface `$datetime2` [, bool `$absolute` = **FALSE**]) : DateInterval

```
$datetime1 = date_create('2009-10-11');
$datetime2 = date_create('2009-10-13');
$interval = date_diff($datetime1, $datetime2);
echo $interval->format('%R%a días');
```

date_add

Añade un periodo de tiempo a una fecha. El periodo de tiempo será un objeto DateInterval. Devuelve un objeto DateTime o false en caso de error.

Estilo orientado a objetos

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

```
$fecha = new DateTime('2000-01-01');
$fecha->add(new DateInterval('P10D'));
echo $fecha->format('Y-m-d') . "\n";
```

Estilo por procedimientos

date_add (DateTime `$object` , DateInterval `$interval`) : DateTime

```
$fecha = date_create('2000-01-01');
date_add($fecha, date_interval_create_from_date_string('10 days'));
echo date_format($fecha, 'Y-m-d');
```

Ejercicios con funciones

1. Realiza una función que acepte una fecha como cadena con el formato dd-mm-aaaa compruebe si la fecha es correcta y nos devuelva la fecha en formato UNIX.
2. Realiza una función que acepte una fecha como cadena con el formato aaaa-mm-dd compruebe si la fecha es correcta y nos devuelva la fecha en formato UNIX.
3. Realiza una función que reciba la fecha en formato UNIX y devuelva la fecha en formato dd-mm-aaaa y aaaa-mm-dd según un segundo argumento que le pasemos a la función.
4. Realiza una función que nos devuelva el nº de días que han pasado entre dos fechas. Hay que tener en cuenta que tendremos que validar las fechas antes o durante la función.
5. Realiza una página que tenga dos cajas de texto para introducir dos fechas. Debemos mostrar la diferencia entre las dos fechas en días totales.

   Se deben realizar todas las comprobaciones necesarias.

Ejercicios con la clase DateTime

Detalla como comentario en el código los tipos de datos de los parámetros y de los datos devueltos.

6. Realiza una función que acepte una fecha como cadena con el formato dd-mm-aaaa compruebe si la fecha es correcta y nos devuelva la fecha en formato aaaa-mm-dd.

7. Realiza una función que acepte una fecha como cadena con el formato aaaa-mm-dd compruebe si la fecha es correcta y nos devuelva la fecha en formato dd-mm-aaaa.
8. Realiza una función que calcule la edad de una persona. Dentro de la función no se realiza la validación de la fecha, se supone que la habremos hecho antes.
9. Realiza una página que tenga dos cajas de texto para introducir dos fechas. Debemos mostrar la diferencia entre las dos fechas en días totales y desglosada en días, meses, años, …

   Se deben realizar todas las comprobaciones necesarias.
10. Realiza una función que muestre una imagen diferente según la estación del año. Para facilitarlo podemos tener en cuenta como primavera (marzo, abril, mayo) y así sucesivamente.