U.T. 3: Manejo de Fecha y Hora en PHP 5 (parte 4)

Contenidos

- Clase DateTime
- Clase DateTimeZone
- Clase DateInterval
- Clase DatePeriod

Clase DateTime

- Clase muy útil para trabajar con fechas y horas en PHP
- Desde la versión de **PHP 5.2** ya hay soporte estable para la clase **DateTime.**
- Definir, en primer lugar, nuestra Zona Horaria:
  - en el archivo **php.ini** dentro de la sección **Date,** o bien,
  - durante la ejecución con la función **date_default_timezone_set.**
  - La función date_default_timezone_set () genera un error si contradice la configuración del php.ini.
  - Listado de zonas horarias soportadas

    http://www.php.net/manual/es/timezones.php

Clase DateTime

- Definir nuestra Zona Horaria en el archivo **php.ini** en la sección **Date**

```
;[Date]
;Defines the default timezone used by the date functions
date.timezone = Europa/Madrid
```

Clase DateTime : Ejemplo

- Crear un objeto de la clase e imprimirlo:

```
<?php
  date_default_timezone_set('Europe/Madrid');
  $nuevaFecha = new DateTime();          //Se genera objeto con la fecha actual
  echo $nuevaFecha->format("Y-m-d H:i:s");       // formato aceptado por date()
?>
```

- Otro ejemplo que utiliza la configuración horaria de php.ini

```
<?php
   date_default_timezone_set('Europe/London'); //puede fallar si contradice
php.ini
   if (date_default_timezone_get())
        echo 'date_default_timezone_set: ' . date_default_timezone_get() . '<br />';
   if (ini_get('date.timezone'))
       echo 'date.timezone: ' . ini_get('date.timezone');
 ?>
```

Clase DateTime : Ejemplo

- Podemos generar el objeto con una fecha específica si así lo deseamos o podemos utilizar algunas cadenas

```
$hoy = new DateTime('now');
echo "\n". $hoy->format("Y-m-d H:i:s");
$ayer = new DateTime('yesterday');
echo "\n". $ayer->format("Y-m-d H:i:s");
$maniana = new DateTime('tomorrow');
echo "\n". $maniana->format("Y-m-d H:i:s");
```

- Formatos relativos de fecha/hora

http://www.php.net/manual/es/datetime.formats.relative.php

  - Conviene examinar la lista de observaciones sobre el comportamiento de las fechas expresadas con formatos relativos.

Clase DateTime : Métodos

- El método constructor de esta clase
  - public __construct ([string $time ="now" [, DateTimeZone $timezone = **NULL**]])

Clase DateTime : Métodos

- Para hacer operaciones con las fechas podemos utilizar los métodos:
  - DateTime::add () (soportada desde la versión 5.3)

    Añade una cantidad de días, meses, años, horas, minutos y segundos al objeto DateTime.
  - DateTime::sub () (desde la versión 5.3)

    Sustrae una cantidad de días, meses, años, horas, minutos y segundos de un objeto DateTime.
  - DateTime::modify () (desde la versión 5.2)

    Modifica la marca de tiempo.
  - DateTime::diff() (desde la versión 5.3)

    Devuelve la diferencia entres objetos DateTime en forma de ***objeto DateInterval.***

Clase DateTime : Métodos

- DateTime::format(string $format) (soportada desde la versión 5.2)

  Devuelve la fecha formateada según el formato dado el formato es el aceptado por date(). Este método no utiliza regiones. Todas las salidas son en inglés.

- Para obtener el formato de fechas y horas en español habría que redefinir el método format(), por ejemplo.

- DateTime::createFromFormat (string $format , string $time [, DateTimeZone $timezone])

  Devuelve un nuevo objeto DateTime formateado según el formato especificado.

Clase DateTime : Ejemplo

- Se detectan los años bisiestos así como el cambio de mes:

```
<?php
        $nuevaFecha = new DateTime('2011-01-25');
        $nuevaFecha->add(new DateInterval('P10D'));    //PERIOD 10 DAYS
        echo "\n".$nuevaFecha->format('Y-m-d');        //Esto imprime 2011-02-04
?>
```

Clase DateTime : Ejemplo

- Si contamos sólo con la versión 5.2 podemos utilizar el método **modify** para hacer operaciones:

```
<?php
        //Probando año bisiesto
        $nuevaFecha = new DateTime("2012-02-28");
        $nuevaFecha->modify("+1 day");
        echo "\n". $nuevaFecha->format("Y-m-d");        //imprime 2012-02-29
        //Probando cambio de año
        $nuevaFecha = new DateTime("2011-01-05");
        $nuevaFecha->modify("-10 day");
        echo "\n". $nuevaFecha->format("Y-m-d");        //imprime 2010-12-26
         $nuevaFecha->modify("-1 hour");
         $nuevaFecha->modify("+10 minutes");
?>
```

Clase DateTime : Ejemplo

- Obtener la diferencia entre 2 fechas:

```
<?php
        $fecha1 = new DateTime('2011-01-01');
        $fecha2 = new DateTime('2011-05-03');
        $intervalo = $fecha1->diff($fecha2);
        echo "\n“ . $intervalo->format('%R%a días');    //Imprime 122 días
?>
```

Clase DateTimeZone

- Representación de la zona horaria.
- Puede utilizarse para cambiar o resetear la zona horaria de un objeto DateTime.

```
<?php
        // obtener y mostrar la hora actual en  Pacific/Auckland
        $zona = new DateTimeZone('Pacific/Auckland');
        $fecha = new DateTime(NULL, $zona);
        echo “<br>” . $fecha->format('H:i:s');    // 01:58:02
?>
```

Clase DateInterval

- Representa un intervalo de fechas.
- Un intervalo de fechas almacena una cantidad fija de momentos (en años, meses, días, horas, etc.), o bien un string de un momento relativo en el formato que admite el constructor de DateTime.
- **Métodos :**
  - public __construct (string ***$interval_spec***)
  - public static DateInterval createFromDateString (string $time)
  - public string format (string $format)

Clase DateInterval

- Una ***especificación de intervalo*** se construye:
  - El formato empieza con la letra ***P***, de "período."
  - Cada período de duración está representado por un valor entero seguido de un indicador de período.
  - Si la duración contiene elementos de hora, esta parte de la especificación está precedida por una letra ***T***.

Indicador de Período	Descripción
Y	años
M	meses
D	días
W	semanas. Éstas se convierten a días, por lo que no se puede combinar con ***D***.
H	horas
M	minutos
S	segundos

Clase DateInterval: Ejemplos

Indicador de Periodo	Descripción
P2D	Dos días
PT2S	Dos segundos
P6YT5M	Seis años y cinco minutos
P1Y4D	Un año y cuatro días

- Los tipos de unidades deben ser escritos desde la unidad de escala más grande a la izquierda a la unidad de escala más pequeña a la derecha.
- Así los años van antes que los meses, meses antes que días, días antes que minutos, etc.

Clase DateInterval: método format()

Carácter *format*	Descripción
%	Literal ***%***
Y, y	Años, numérico, al menos 2 dígitos empezando con 0
M,m	Meses, numérico, al menos 2 dígitos empezando con 0
D,d	Días, numérico, al menos 2 dígitos empezando con 0
a	Número total de días como resultado de una operación con **DateTime:diff()**, o de lo contrario ***(unknown)***
H,h	Horas, numérico, al menos 2 dígitos empezando con 0
I, i	Minutos, numérico, al menos 2 dígitos empezando con 0
S,s	Segundos, numérico, al menos 2 dígitos empezando con 0
R	Signo "**-**" cuando es negativo, "***+***" cuando es positivo
r	Signo "**-**" cuando es negativo, vacío cuando es positivo

- Cada carácter de formato debe estar precedido con un signo (%).

Clase DateInterval: Ejemplo

```
<?php
        $timespan=10;
        $d1 = new DateTime();
        $d2 = new DateTime();
        $d2->add(new DateInterval('PT'.$timespan.'S'));
        $resul = $d2->diff($d1);
        echo '<br>'.$resul->format('%s');   // Visualiza los 10 segundos
?>
```

Clase DatePeriod

- Representa un período de fechas.
- Un período de la fecha permite la iteración sobre un conjunto de fechas y horas, se repiten a intervalos regulares, durante un período determinado.
- **Métodos**
  - public __construct (DateTime $start , DateInterval $interval , int $recurrences [, int $options])
  - public __construct (DateTime $start , DateInterval $interval , DateTime $end [, int $options])
  - public __construct (string $isostr [, int $options])