Les dates

El maneig de dates i hores en PHP ha evolucionat molt, passant de les funcions bàsiques a una orientació a objectes molt més robusta i fiable. Entendre aquest sistema és crucial per a qualsevol aplicació, especialment per a la correcta gestió de fusos horaris.

El funcionament intern de les dates en PHP

PHP gestiona el temps i les dates basant-se en el concepte de **Timestamp Unix** (Marca de temps Unix), tot i que la majoria de les funcions modernes treballen internament amb objectes.

El timestamp Unix

Un *timestamp* Unix és un valor sencer que representa el **nombre de segons** transcorreguts des de l'**Època Unix** (la mitjanit UTC de l'1 de gener de 1970).

- **Avantatge:** És universal, independent del fus horari i ideal per a emmagatzemar dates a bases de dades o fer càlculs de temps (sumar o restar segons).
- **Funció Clau:** La funció `time()` retorna el *timestamp* Unix actual.

El fus Horari (Timezone)

El fus horari és la configuració més crítica. PHP necessita saber en quin fus horari ha d'interpretar i mostrar les dates, ja que un mateix *timestamp* pot representar una hora diferent segons la ubicació.

Configuració (PHP.INI)

Dos paràmetres al fitxer `php.ini` (o configurats dinàmicament al codi) afecten directament el maneig de dates:

`date.timezone`

Aquest és el paràmetre **obligatori** i més important. Estableix el fus horari per defecte que PHP utilitzarà per a totes les operacions de data i hora.

- **Configuració Recomanada:** S'ha d'establir a una cadena de fus horari vàlida (e.g., `Europe/Madrid`, `America/New_York`).
- **Problema Comú:** Si no es defineix, PHP genera un avís (`Warning`) i utilitza un fus horari per defecte insegur, cosa que pot causar errors de càlcul d'una hora.
- **Configuració Dinàmica:** Es pot sobrescriure al codi amb `date_default_timezone_set('Europe/Madrid')`.

`date.default_latitude` / `date.default_longitude`

Aquests paràmetres, juntament amb `date.sunrise_color` i `date.sunset_color`, s'utilitzen per a funcions de càlcul d'efemèrides com ara `date_sunrise()` i `date_sunset()`. No afecten el format de data ni l'hora.

Funcions disponibles (Procedural)

PHP ofereix un conjunt de funcions antigues però encara molt utilitzades, que converteixen entre *timestamps* i cadenes de text (strings):

Lectura de temps

- **`time()`:** Retorna el *timestamp* Unix actual.
- **`microtime(true)`:** Retorna el *timestamp* Unix amb precisió de microsegons (com a `float`).

Formatació de Dates

- **`date(string $format, int $timestamp = time()): string`:**
  - Converteix un *timestamp* Unix (o el temps actual per defecte) en una cadena de data/hora formatada.
  - El paràmetre `$format` utilitza caràcters especials (e.g., `Y` per a l'any complet, `m` per al mes numèric, `H` per a les hores).
  - *Exemple:* `date('d/m/Y H:i:s')` retorna `"26/09/2025 11:24:54"`.
- **`gmdate()`:** Igual que `date()`, però sempre formatat en el fus horari **GMT/UTC**.

Conversió de cadena a Timestamp

- **`strtotime(string $time, int $now = time()): int`:**
  - La funció **`strtotime()`** de PHP és una de les funcions més potents i flexibles, tot i ser de l'enfocament procedural antic. La seva funció principal és analitzar (o interpretar) una cadena de text en anglès (*human-readable*) que descriu una data o hora i convertir-la al **Timestamp Unix** corresponent (el nombre de segons des de l'1 de gener de 1970 00:00:00 UTC).
    És popular per la seva capacitat d'interpretar formats absoluts (com "2025-12-31") i formats relatius (com "next Monday" o "+2 weeks").
  - *Exemples:* `strtotime('tomorrow')`, `strtotime('+1 week')`, `strtotime('2025-01-01')`.

> https://www.php.net/manual/es/function.strtotime.php

- **`mktime()`:** Retorna el *timestamp* Unix d'una data específica, permetent passar any, mes, dia, hora, minut, i segon com a arguments separats.

Exemples. Formats Date

```
 <?php
$diaAvui=date("d");
echo "formato: date(\"d\")->   ".$diaAvui."<br/>";
$diaAvui=date("D");
echo "formato: date(\"D\")->   ".$diaAvui."<br/>";
$diaAvui=date("j");
echo "formato: date(\"j\")->   ".$diaAvui."<br/>";
$diaAvui=date("l");
echo "formato: date(\"l\")->   ".$diaAvui."<br/>";
$diaAvui=date("N");
echo "formato: date(\"N\")->   ".$diaAvui."<br/>";
$diaAvui=date("S");
echo "formato: date(\"S\")->   ".$diaAvui."<br/>";
$diaAvui=date("d").date("S");
echo "formato: date(\"d\").date(\"S\")->   ".$diaAvui."<br/>";
$dia=date("d-m-Y");
echo "formato: ".$dia."<br/>";
$dia=date("d-m-Y",strtotime("15-12-2025"));
```

```
echo "formato: ".$dia."<br/>";
$dia=date("N",strtotime("8-11-2025"));
echo "formato: ".$dia."<br/>";
```

```
<?php
echo "Avui es: " . date ("j M Y G:i:s") . "<br/>";
echo date ("YmdGis") . "<br/>"; //202509281534
echo "Dia setmana: " . date ("w") . "<br/>";
echo "Dia setmana: " . date ("w", strtotime("09/11/2025") ). "<br/>";

echo "Mes actual: " . date("n") . "<br/>";
$mes=date("n")+1;
echo "Mes actual: " . $mes . "<br/>";

$darrerDiaMes = date("t")."-".date("n")."-".date("Y");
echo "Dia setmana darrer dia mes: " . date("w",strtotime($darrerDiaMes));
```

Exemples. Date avui

```
 <?php
$diaAvui=date("N");
switch ($diaAvui) {
    case 1:
        $diaSemana="Lunes";
        break;
    case 2:
        $diaSemana="Martes";
        break;
    case 3:
        $diaSemana="Miercoles";
        break;
    case 4:
        $diaSemana="Jueves";
        break;
```

```
    case 5:
        $diaSemana="Viernes";
        break;
    case 6:
        $diaSemana="Sabado";
        break;
    case 7:
        $diaSemana="Domingo";
        break;
}
$mesAvui=date("n");
switch ($mesAvui) {
    case 1:
        $Mes="Enero";
        break;
    case 2:
        $Mes="Febrero";
        break;
    case 3:
        $Mes="Marzo";
        break;
    case 4:
        $Mes="Abril";
        break;
    case 5:
        $Mes="Mayo";
        break;
    case 6:
        $Mes="Junio";
        break;
    case 7:
        $Mes="Julio";
        break;
    case 8:
        $Mes="Agosto";
        break;
    case 9:
        $Mes="Septiembre";
        break;
```

```
    case 10:
        $Mes="Octubre";
        break;
    case 11:
        $Mes="Noviembre";
        break;
    case 12:
        $Mes="Diciembre";
        break;
}

echo $diaSemana.", ".date("j")." de ".$Mes." de ".date("Y");
```

Exemples. Operacions Date

```
 <?php
$fecha_actual = date("d-m-Y");
//sumo 1 día
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Sumo 31 dia:       '.date("d-m-Y",strtotime($fecha_actual."+ 31
days"))."<br/><br/>";
//resto 1 día
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Resto 31 dia:       '.date("d-m-Y",strtotime($fecha_actual."- 31
days"))."<br/><br/>";

//sumo 1 mes
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Sumo 1 mes:       '.date("d-m-Y",strtotime($fecha_actual."+ 1 month"))."<br/><br/>";

//resto 1 mes
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Resto 1 mes:       '.date("d-m-Y",strtotime($fecha_actual."- 1
month"))."<br/><br/>";

//sumo 1 año
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
```

```
echo 'Sumo 1 año:       '.date("d-m-Y",strtotime($fecha_actual."+ 1 year"))."<br/><br/>";
//resto 1 año
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Resto 1 año:       '.date("d-m-Y",strtotime($fecha_actual."- 1 year"))."<br/><br/>";


//sumo 1 semana
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Semana Siguiente: '. date("d-m-Y",strtotime($fecha_actual."+ 1 week"))."<br/><br/>";

//resto 1 semana
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Semana Anterior: '. date("d-m-Y",strtotime($fecha_actual."- 1 week"))."<br/><br/>";

$semanaSiguiente = time() + (7 * 24 * 60 * 60); // 7 días; 24 horas; 60 minutos; 60
segundos
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Semana Siguiente: '. date('d-m-Y', $semanaSiguiente) ."<br/><br/>";

$semanaSiguiente = time() - (7 * 24 * 60 * 60); // 7 días; 24 horas; 60 minutos; 60
segundos
echo 'Ahora:            '. date('d-m-Y') ."<br/>";
echo 'Semana Anterior: '. date('d-m-Y', $semanaSiguiente) ."<br/><br/>";
```

> **strtotime(STRING).**
> Retorna un object timestamp a partir d'una data expresada en la cadena de caracters
> - Si el separador es / llavors la data esta expressada MES/DIA/ANY
> - Si el separador es - llavors la data està expressada DIA/MES/ANY

La funció `strtotime()` de PHP té un comportament diferent i potencialment confús quan s'utilitza la barra inclinada (`/`) com a separador entre els components numèrics de la data (mes, dia, any), en comparació amb el guió (`-`).

Aquesta diferència es basa en la manera com PHP, i en última instància, la biblioteca C subjacent, intenta inferir quin format de data s'està utilitzant.

El Comportament de `strtotime()` segons el Separador

La clau de la interpretació de `strtotime()` rau en la distinció entre el format americà (**M/D/Y**) i el format europeu (**D/M/Y**).

El Guió (`-`): Format ISO 8601 (Any-Mes-Dia)

Quan utilitzes el guió, `strtotime()` assumeix el format més estricte i universal: **Any-Mes-Dia (Y−M−D)**.

Entrada (Gió)	Format Interpretar	Explicació
`"2025-03-01"`	Y−M−D	S'interpreta com l'1 de març de 2025.
`"03-01-2025"`	Y−M−D	S'interpreta com el 3 de gener de 2025.

Conclusió amb `-`: El guió indueix el format internacional (ISO 8601), que prioritza l'any al principi (o l'any al final si es presenta només amb 3 components, assumint M−D−Y).

La Barra Inclinada (`/`): Format Americà (Mes/Dia/Any) US

Quan utilitzes la barra inclinada (`/`), `strtotime()` prioritza el format de data curt utilitzat comunament als Estats Units: **Mes/Dia/Any (M/D/Y)**.

Entrada (Barra)	Format Interpretar	Explicació
`"03/01/2025"`	**M/D/Y**	S'interpreta com l'1 de març de 2025.
`"01/03/2025"`	**M/D/Y**	S'interpreta com el 3 de gener de 2025.
`"31/01/2025"`	Error	Si el primer número supera 12 (més d'un mes), l'anàlisi falla, ja que s'espera un mes.

Conclusió amb `/`: La barra inclinada obliga a utilitzar la lògica **Mes/Dia/Any (M/D/Y)**, independentment de les preferències regionals del servidor.

L'Ambivalència i el Risc d'Errors

El problema sorgeix quan un desenvolupador en una regió on s'utilitza el format Dia/Mes/Any (D/M/Y) assumeix que la barra inclinada mantindrà aquest ordre.

Desenvolupador	Espera	PHP Llegeix (M/D/Y)	Resultat (Error)
Europeu	12 de maig de 2025 (`12/05/2025`)	5 de desembre de 2025	**Data Incorrecta**
Europeu	5 de desembre de 2025 (`05/12/2025`)	12 de maig de 2025	**Data Incorrecta**

Això es coneix com a **"ambigüitat de format de data"** i pot provocar errors subtils i difícils de detectar en la lògica de l'aplicació.

La solució recomanada: `DateTime` amb Format Explícit

Per evitar qualsevol ambigüitat, la pràctica recomanada és **evitar totalment l'ús de `strtotime()`** per analitzar dates d'entrada de l'usuari amb format D/M/Y o M/D/Y. En el seu lloc, s'ha d'utilitzar el mètode **`DateTime::createFromFormat()`**, ja que obliga a PHP a utilitzar el format de data que tu especifiques amb exactitud.

PHP

```
<?php
// Entrada de l'usuari (format català/europeu esperat: DD/MM/YYYY)
$data_entrada = "01/03/2025";

// 1. Ús ambigü (i erroni en context europeu):
// $timestamp_error = strtotime($data_entrada);
// S'interpreta com Març 1 (M/D) en lloc de l'1 de Març (D/M).

// 2. Ús Correcte (Orientat a Objectes):
// Indica explícitament a PHP que la cadena és Dia/Mes/Any
$data_objecte = DateTime::createFromFormat('d/m/Y', $data_entrada);

if ($data_objecte) {
    // La data és l'1 de març de 2025
    echo "Data analitzada correctament: " . $data_objecte->format('Y-m-d');
} else {
    echo "Error: El format de data no és vàlid.";
}
```

Amb `createFromFormat()`, elimines qualsevol conjectura de PHP sobre si estàs utilitzant un guió, una barra inclinada o un punt, ja que tu defineixes la màscara d'anàlisi exacta.

Objectes per dates (Enfocament OO)

L'enfocament Orientat a Objectes (OOP) és l'estàndard modern en PHP, ja que resol problemes complexos de càlculs de dates i gestió de fusos horaris amb molta més fiabilitat.

`DateTime` (o `DateTimeImmutable`)

Representa una data i hora concretes, sempre associada a un fus horari:

Mètode Clau	Descripció	Exemple
Constructor	Crea un nou objecte data.	`new DateTime()` (ara) o `new DateTime('2025-10-20')`.
`format(string $format)`	Formata l'objecte data en una cadena (similar a la funció `date()`).	`$data->format('Y-m-d')`
`setTimezone(DateTimeZone $timezone)`	Canvia el fus horari de l'objecte sense alterar el punt en el temps.	`$data->setTimezone(new DateTimeZone('America/New_York'))`
`createFromFormat()`	Analitza una data d'una cadena amb un format conegut de forma segura.	`DateTime::createFromFormat('d/m/Y', '26/09/2025')`

`DateTimeZone`

Un objecte que encapsula un fus horari concret.

- **Ús:** Es fa servir com a argument en el constructor de `DateTime` o al mètode `setTimezone()` .
- *Exemple:* `new DateTimeZone('Asia/Tokyo')`.

`DateInterval`

Representa un període de temps (e.g., 2 anys, 3 mesos i 5 dies).

- **Ús:** S'utilitza en operacions de data/hora.
- *Exemple:* `new DateInterval('P2Y3M5D')` (P de Període, 2 Anys, 3 Mesos, 5 Dies).

`DatePeriod`

Representa una col·lecció de dates i hores dins d'un període, iterant segons un `DateInterval`.

- **Ús:** Generar una llista de totes les hores laborables entre dues dates.

`DateTime` per càlculs (Immutabilitat)

La major fortalesa de l'OOP és el maneig de càlculs amb els mètodes **`add()`** i **`sub()`** (i **`diff()`** per calcular la diferència entre dues dates).

Per evitar modificar l'objecte original (un risc en programació), és molt recomanable utilitzar `DateTimeImmutable`, que sempre retorna una nova instància amb el canvi aplicat:

PHP

```php
<?php
$ara = new DateTimeImmutable('now'); // Data original inalterable

// Sumem un període de temps (un mes i 5 dies)
$interval = new DateInterval('P1M5D');
$futur = $ara->add($interval); // Crea un nou objecte

echo "Ara: " . $ara->format('Y-m-d') . "\n";
echo "Futur: " . $futur->format('Y-m-d') . "\n";
```

Revisió núm. 1
Admin l'ha creat 2025-09-26 11:24:13 UTC
Admin l'ha actualitzat 2025-09-26 11:48:04 UTC