# Funzioni per lavorare con le date in PHP

In php | date | datetime | strftime | strtotime | setlocale | funzioni

In php c'è un'ampia varietà di funzioni e anche alcune classi che permettono la gestione delle date in modo facile anche in situazioni che richiedono procedure complesse.
Chiunque abbia una minima conoscenza di php probabilmente avrà usato almeno una volta la funzione:

`date()`

Nonostante ciò la migliore funzione php per il "parsing" delle date, specialmente se vogliamo lavorare con il `setlocale` o comunque per visualizzare date in italiano è:

`strftime()` php doc

## strftime()

Per utilizzare strftime() è necessario fornire il parametro "formato" e eventualmente l'unixtime della data da visualizzare.

```
echo strftime('%d/%m/%Y', time()); // output: 02/12/2013
```

## strtotime()

Una funzione che vi ritroverete più spesso ad utilizzare combinata a strftime è:

`strtotime()` php doc

Questa funzione vi aiuterà a compiere delle operazioni in modo dinamico sulle date ed ottenere l'unixtime corretto.
Strtotime supporta la possibilità di dichiarare *testualmente* che cosa fare con una determinata data.
Dichiariamo una data da convertire in unixtime in questo modo:

```
echo strtotime('2013-12-02'); // ottengo l'unixtime
```

Una delle cose più divertenti che possiamo fare con questa funzione è ad esempio aggiungere o togliere, mesi, anni, giorni, settimane etc...
Vediamo alcuni esempi:

```
strtotime('2013-12-02 + 1 year'); // aggiunge un anno
strtotime('2013-12-02 - 2 years'); // tolgo 2 anni
strtotime('2013-12-02 + 3 months'); // aggiungo 3 mesi
strtotime('2013-12-02 - 10 days'); // tolgo 10 giorni
strtotime('2013-12-02 - 1 day'); // tolgo 1 giorno
strtotime('2013-12-02 + 1 week'); // aggiungo 1 settimana
```

Automaticamente per ognuna di queste date otterremo la data in unixtime in modo da poter agevolmente visualizzarla con strftime().

```
$plusoneyear = strtotime('2013-12-02 + 1 year');
echo strftime('%d/%m/%Y', $plusoneyear); // stampa 02/12/2014
```

A volte però è necessario conoscere date non facili da ottenere con una addizione o una sottrazione, con strtotime è possibile scrivere ad esempio:

```
echo strtotime("next Monday"); // ottengo la data di Lunedì prossimo
echo strtotime("+1 week 2 days 4 hours 2 seconds"); // alla data di oggi
aggiungo 1 settimana 2 giorni 4 ore e 2 secondi
```

Un esempio più complesso potrebbe essere:

```
<?php

  $start = strtotime('next Monday');
  $meseintero = strtotime('next Month');

  for ($start; $start < $meseintero; next) {
          echo strftime('%d %B %Y', $start)."<br />";
          $start = strtotime(strftime('%Y-%m-%d', $start).'next Monday');
  }

?>
```

*Con questo script voglio sapere tutte le date dei prossimi Lunedì fino al prossimo mese.*

La variabile $start ottiene la data del prossimo Lunedì, mentre $meseintero variabilizza la data di un mese esatto da oggi.
Il codice all'interno del ciclo stampa con un "echo" la data del Lunedì prossimo, successivamente aumenta la variabile $start di un ulteriore Lunedì ripetendo la stessa procedura fintanto che la condizione specificata nel ciclo for sia soddisfatta; ovvero che $start è minore di $meseintero.
Quindi partendo da un ipotetico 03/12/2013 l'output risulterà questo:

```
09 December 2013
16 December 2013
23 December 2013
30 December 2013
```

Che risultano essere tutti i prossimi lunedì.

## setlocale()

Nell'ultimo esempio abbiamo visto che strftime quando ha come parametro %B mostra il nome del mese in inglese, ecco che con setlocale possiamo facilmente convertire tutte le parole riguardati "date" nella lingua che preferiamo, nel nostro caso in italiano.

Possiamo scrivere infatti a titolo di esempio:

```
setlocale(LC_ALL, 'it_IT');
echo strftime('%B'); // output Dicembre
```

Attenzione! Nel caso il setlocale non dovesse funzionare, non preoccupatevi, in alcune installazioni il pacchetto locale potrebbe non chiamarsi it_IT, per conoscere il nome del pacchetto locale della lingua italiana, se avete accesso alla console, scrivete:

```
$ locale -a
```

(chiaramente il simbolo del dollaro va omesso, l'ho scritto per identificare che è un comando da terminale/console). Dovrebbe apparirvi una lista di nomi di pacchetti, copiate il nome della lingua e utilizzatela al posto di it_IT nella funzione setlocale, ad esempio se nella lista l'italiano si chiama itIT.utf-8, avremo:

```
setlocale(LC_ALL, 'itIT.utf-8');
```

Se non avete accesso al terminale, create una pagina php con il seguente codice:

```
<?php system('locale -a'); ?>
```

Eseguite la pagina tramite browser, verranno stampati a video i pacchetti.

## La classe DateTime

Questa estensione è presente dalla versione 5.2 di php.
Istanziamo una variabile $data come ogni altra classe, dopodiché potremo andare ad agire su di essa in molti modi.

```
$data = new DateTime();
```

### getTimestamp()

Otteniamo l'unixtime di $data:

```
echo $data->getTimestamp(); // output: 1386174071
```

### format();

Possiamo specificare il formato con cui ottenere la data:

```
echo $data->format('d/m/Y'); // output: 04/12/2013
```

### Modifica

Ci sono diversi modi per modificare $data, uno di questi può essere:

```
echo $data->format('d/m/Y').'<br />'; // output: 04/12/2013
$data->modify('tomorrow');
echo $data->format('d/m/Y'); //output: 05/12/2013
```

La modifica può avvenire anche testualmente:

```
$data->modify('+ 3 days');
$data->modify('+ 1 year');
$data->modify('yesterday');
etc...
```

La stessa cosa può essere fatta al momento dell'instanziamento, ad esempio:

```
$data = new DateTime('yesterday');
```

Altri metodi di modifica sono:

```
$date->setTimestamp(1386028800);
$date->setDate(2013, 12, 05);
$date->setTime(10, 2, 15);
```

### Confronti

Con la classe DateTime è anche possibile creare confronti:

```
$pippo = new DateTime('1932-05-02');
$pluto = new DateTime('1931-04-30');

if($pippo > $pluto)
    echo 'Pluto è apparso prima di Pippo';
```

Inoltre con diff() siamo in grado di conoscere alcune informazioni comparando due date:

```
$differenza = $pluto->diff($pippo);
echo $differenza->format('Pluto è uscito %y anno e %d giorni prima di Pippo');
```

Per conoscere effettivamente che cosa abbiamo a disposizione in $differenza:

```
var_dump($differenza);
```

## TimeZone

In passato ho avuto qualche problema a lavorare con il Timezone, con DateTime tutto diventa molto più semplice.

```
date_default_timezone_set('Europe/Rome');
$data = new DateTime('now');

echo $data->format('d/m/Y H:m'); //output: timezone +1
```

Se dobbiamo però modificare on-the-fly il timezone, utilizziamo setTimezone:

```
date_default_timezone_set('Europe/Rome');
$data = new DateTime('now');

echo $data->format('d/m/Y H:m').'<br />'; // output: 04/12/2013 18:12

$timezone = new DateTimeZone('America/Los_Angeles');
$data->setTimezone($timezone);

echo $data->format('d/m/Y H:m'); output: 04/12/2013 09:12
```

## Conclusione

In sostanza ci sono diverse alternative in php per giocare con le date, a mio parere il metodo migliore se non dobbiamo effettuare molte operazioni è l'utilizzo di strftime(), nel caso si dovesse lavorare con procedure più complesse, mediante l'utilizzo e la comparazione di più date, credo che la Classe DateTime faccia al caso vostro.
Potete trovare più informazioni ed esempi nella documentazione php:
http://www.php.net/manual/en/book.datetime.php