Datum und Uhrzeit in PHP

Nicht nur Menschen arbeiten täglich mit Datums- und Zeitangaben, auch Rechner benötigen ständig Zeitangaben für Berechnungen. Welche Möglichkeiten mit PHP zur Verfügung stehen erklärt dieser Artikel.

Die date()-Funktion

Die einfachste Methode um ein Datum auszulesen erfolgt über den Befehl **date()**. Dabei erwartet der Befehl **Parameter** über die auszulesenden Größen, die Reihenfolge der Ausgaben wird im Befehl hinterlegt. Beispiele für solche Parameter findest du in *Tabelle 1*.

Formatzeichen	Beschreibung
%a	Wochentagsname, abgekürzt
%A	Wochentagsname, ausgeschrieben
%b	Monatsname, abgekürzt
%B	Monatsname
%c	Vollständiges Datum
%C	Jahrhundert, als Zahl von 00 bis 99
%d	Tag im Monat als zweistellige Zahl
%D	Ausgabe woe m/d/y
%e	Tag des Monats als Dezimalwert
%G	Jahr als 4-stellige Zahl wie ISO-Wochennummer
%g	wie G, jedoch ohne Jahrhundert
%h	Monatsname, abgekürzt
%H	Stunde als Zahl (00 bis 23)
%I	Stunde als Zahl (00 bis 12)
%j	Tag im Jahr (001 bis 366)
%m	Monat als Zahl (01 bis 12)
%M	Minute als Zahl (00 bis 59)
%n	Zeilenumbruch
%p	am oder pm
%R	Zeit im 24 Stdunden Format
%S	Sekunden als Zahl (00 bis 59)
%t	Tabulator
%T	aktuelle Zeit
%u	Wochentag als Zahl von 1 bis 7
%U	Nummer der Woche des aktuellen Jahres
%V	Nummer der Woche, erste Woche mit min. 4 Tagen
%w	Wochentag als Zahl (Sonntag = 0)
%W	Woche im aktuellen Jahr, Woche 01 beginnt am ersten Montag
%x	Datum ohne Zeit (entsprechend locale-Einstellung)
%X	Zeit ohne Datum (entsprechend locale-Einstellung)
%y	Jahr als Zahl (00 bis 99 ohne Jh.)
%Y	Jahr als Zahl (mit Jahrhundert)
%Z	Zeitzone, Name oder ene Abkürzung
%%	Prozentzeichen

Tabelle 1: Parameter für date() und strftime()

Folgendes Beispiel zeigt dir das Ausgeben von einem Datum, hier im Format *17th December 2006*.

```
echo date('jS F Y');
```

Wenn date() als zweiter Parameter ein Timestamp übergeben wird, gibt die Funktion statt des aktuellen Datums das Datum des Timestamps aus.

Allerdings hat date() im deutschen Sprachraum einen entscheidenden Nachteil, die Ausgabe des Datums ist auf Englisch. Das heißt, statt *Dezember* wird *December* ausgegeben. Um das zu umgehen, gibt es die beiden Funktionen

setlocale() und **strftime()**. setlocale() setzt, wie der Name schon sagt, lokale Informationen. Um das Datumsformat auf Deutsch zu setzen, schreibst du diesen Code in dein Script:

```
setlocale(LC_TIME, 'German_Germany');
```

Um das Datum auszugeben, benutzt du statt date() strftime(). strftime() erwartet genau wie date() als Parameter einen Parameter.

Beispielcode:

```
setlocale(LC_TIME, 'German_Germany');
echo 'Heute ist ' . strftime('%A, der %d. %B %Y');
echo '<br />';
echo 'Es ist ' . strftime('%H:%M') . ' Uhr';
```

Der Unix-Zeitstempel

Ein **Zeitstempel** (Timestamp) speichert die Anzahl der Sekunden, die seit Mitternacht des *1. Januar 1970 GMT* verstrichen sind (Unix-Epoche). Dies sieht auf den ersten Blick vielleicht ziemlich umständlich aus, ist aber sehr praktisch, da sich mit einfachen Zahlen deutlich besser rechnen lässt. Wenn du z.B. die nach dem Eintragungsdatum sortierten Einträge eines Gästebuches auslesen willst, kannst du die Timestamps miteinander vergleichen. Je jünger der Eintrag desto höher die Zahl. Bei direkten Uhrzeiten wäre dieser Vergleich deutlich aufwendiger.

Des Weiteren haben Timestamps, im Gegensatz zu andern Datumsformaten, kein Jahr-2000-Problem. Allerdings gibt es andere Probleme: Timestamps werden in einem 32-Bit-Integer gespeichert, dadurch können sie nur eine bestimmte Zeitspanne speichern (1902 bis 2038). Unter Windows und einigen anderen Systemen sind keine negativen Timestamps möglich deshalb kannst du nicht mit Datumswerten vor 1970 rechnen.

time()

Die einfachste Methode wie du einen solchen Zeitstempel erzeugen kannst, ist der Befehl **time()**.

```
$timestamp = $time();
```

mktime()

Da häufig mit normalen Daten anstatt timestamps umgegangen wird, ist die Konvertierung zwischen beiden Formaten nötig. Die Umwandlung von einem normalen Datum in einen timestamp erreichst du über den Befehl **mktime()**. Der grundlegende Befehlsaufbau erfolgt wie im folgenden Codebeispiel.

```
int mktime ( [int Stunde [, int Minute [, int Sekunde [, int Monat [, int Tag [, int Jahr
[, int is_dst]]]]]]] )
```

Der letzte Parameter *is_dst* stellt klar, ob das Datum in der Sommer- (**1**) oder Winterzeit (**0**) liegt. Beim Vorgabewert (**-1**) ist nicht bekannt, ob das Datum in der Sommer- oder Winterzeit liegt.

getdate()

Die Umkehrfunktion zu mktime() lautet **getdate()**. Dabei greift der Befehl auf einen bestehenden timestamp zurück, und gibt ein Array zurück. Die einzelnen Arrays kannst du über **Schlüssel** aufrufen, welche Schlüssel du benutzen kannst zeigt dir die *Tabelle 2*.

Schlüssel	Beschreibung
seconds	Anzahl der Sekunden (00 bis 59)
minutes	Anzahl der Minuten (00 bis 59)
hours	Anzahl der Stunden (00 bis 23)
mday	Numerischer Tag des Monats (1 bis 31)
wday	Numerischer Wochentag (0 bis 6)
mon	Monatszahl (1 bis 12)
year	Vierstellige Jahreszahl
yday	Numerischer Tag des Jahres (0 bis 365)
weekday	Ausgeschriebener Wochentag
month	Ausgeschriebener Monatsname, wie Januar oder März

Tabelle 2: Schlüssel für getdate()

Datumswerte überprüfen

Oft ist es nötig, die Gültigkeit eines Datums zu überprüfen. Dafür benutzt du in PHP die Funktion **checkdate()**. Diese erwartet als Parameter den Monat, den Tag und das Jahr (in dieser Reihenfolge) und gibt TRUE bzw. FALSE zurück.

Beispiel:

//Liefert TRUE zurück:

```
checkdate(6, 10, 1989);
```

//Liefert FALSE zurück:

```
checkdate(2, 30, 2006);
```

Weblinks

Suchst du weitere Informationen zu den gezeigten Befehlen oder Zeitstempeln, kannst du auf diesen Seiten nachschlagen:

http://www.php.net/manual/de/ref.datetime.php
http://www.php-faq.de/ch/ch-datetime.html