Relative Zeitangaben mit der eingebauten PHP-Funktion strtotime() erstellen

Dr. Volker Thormählen, 11. Juni 2019

Inhalt

Abbildungen

Listings

Tabellen

1 Relative Zeitangaben nutzen

In diesem Beitrag wird beschrieben, wie *relative Zeitangaben* genutzt werden können. Die Skriptsprache PHP enthält die eingebaute Funktion *strtotime()*, die sich dafür geeignet. Sie erwartet bis zu zwei Parameter, die nachfolgend erörtert werden.

Anhand zahlreicher Beispiele wird des Weiteren gezeigt, wie die genannte Funktion in der Praxis angewandt werden kann.

Grundkenntnisse der Skriptsprache PHP werden vorausgesetzt.

2 Parameter der PHP-Funktion strtotime()

Die eingebaute PHP-Funktion *strtotime()*[1] kann verwendet werden, um *relative* Zeitangaben zu generieren. Sie wandelt ein beliebiges in englischer Schreibweise angegebenes Datum in einen UNIX-Zeitstempel um. Sie benötigt bis zu zwei Parameter:

- Der erste Parameter enthält den gewünschten zeitlichen Abstand zum Bezugszeitpunkt, zum Beispiel "+1 week".
- Im zweiten Parameter wird ein Bezugszeitpunkt in Form eines Zeitstempels definiert. Falls dieser fehlt, wird die aktuelle Systemzeit verwendet.

3 Zeitlichen Abstand definieren

Der zeitliche Abstand wird beispielsweise wie folgt definiert (s. Listing 1, grau hinterlegte engl. Texte):

```
<?php                                                    // Ergebnis
echo date('d.m.Y', strtotime("now")), "\n";              // 11.06.2019 (=> heute)
echo date('d.m.Y', strtotime("14 September 2000")), "\n"; // 14.09.2000
echo date('d.m.Y', strtotime("+1 day")), "\n";           // 12.06.2019
echo date('d.m.Y', strtotime("+1 week")), "\n";          // 18.06.2019
echo date('d.m.Y', strtotime("+1 week 2 days")), "\n";   // 20.06.2019
echo date('d.m.Y', strtotime("next Friday")), "\n";      // 14.06.2019
echo date('d.m.Y', strtotime("last Monday")), "\n";      // 10.06.2019
?>
```

Listing 1: Zeitlichen Abstand definieren

4 Bezugszeitpunkt erstellen

Mit der eingebauten PHP-Funktion *mktime()* kann der Bezugszeitpunkt *$heute* erstellt werden, beispielsweise für den 11.06.2019 mit dieser Syntax: *mktime*("stunden", "minuten", "sekunden", "monat", "tag", "jahr") (s. Listing 2):

```
// Datum $heute erzeugen mit der eingebauten PHP-Funktion mktime().
$tm = mktime(0,0,0,6,11,2019);
$heute = date('d.m.Y',$tm);
echo $heute.'<br>';     // Ergebnis: 11.06.2019
```

[1] Die PHP-Funktion *strtotime()* arbeitet mit dem sog. *timestamp.* Er wird seit dem 1.1.1970 im Sekundentakt hochgezählt. Deshalb können Daten davor nicht ermittelt werden.

```
// Datum $jetzt erzeugen mit der eingebauten PHP-Funktion time().
date_default_timezone_set("Europe/Berlin");
$tm = time();
$jetzt = date('d.m.Y',$tm);
```

Listing 2: Bezugsdaten erzeugen

Alternativ kann der Bezugszeitpunkt *$jetzt* mit der eingebauten PHP-Funktion *time()* erzeugt werden (s. Listing 2).

5 Praktische Beispiele

5.1 Englische Text-Schnipsel

Die Webseite [1] beschreibt verschiedene relative Datums- /Zeitangaben, die der sog. Parser der eingebauten PHP-Funktion *strtotime()* versteht. Tabelle 1 enthält praktische Beispiele:

Zeitstempel erzeugen mit ...	**strtotime('...')**[2]
Erster Mittwoch im Dezember 2015	first wednesday 2015-12
Letzter Freitag	last friday
Erster Tag im Folgemonat	first day of next month
Nächster Donnerstag	next thursday
Erster Montag im Januar 2015	first monday 2015-01
Erster Montag im Januar 2016	first monday 2016-01
Erster Montag im Folgemonat	first monday of next month
Letzter Tag im Vormonat	last day of last month
Letzter Tag in diesem Monats	last day of this month
Zweiter Freitag im Vormonat	second friday of last month
Letzter Freitag im aktuellen Monat	last friday of this month

Tabelle 1: Praktische Beispiele zur Erzeugung von Zeitstempeln

5.2 Erstes Beispiel

Angenommen, das aktuelle Systemdatum ist der 12.06.2019. Mit folgendem PHP-Code kann der in Tabelle 1, Sp. 2, letzte Zeile, enthaltene englische Text angewandt werden:

```
echo date('D, d.m.Y', strtotime('last friday of this month'));
```

Listing 3: PHP-Code zur Ermittlung des letzten Freitags im aktuellen Monat

Das Ergebnis ist: Fri, 28.06.2019.

5.3 Zweites Beispiel

Angenommen, das aktuelle Systemdatum ist wiederum der 12.06.2019. Mit folgendem PHP-Code kann der in Tabelle 1, Sp. 2, zweitletzte Zeile, enthaltene englische Text angewandt werden:

```
echo date('D, d.m.Y', strtotime('second friday of last month'));
```

Listing 4: PHP-Code zur Ermittlung des zweiten Freitags im Vormonat

Das Ergebnis ist: Fri, 10.05.2019

[2] Quelle: In Anlehnung an [3]

6 Terminliste für einen Verein erzeugen

6.1 Scenario

Angenommen, ein beliebiger Verein (z. B. Kegler, Wanderer, Sänger oder dergleichen) trifft sich regelmäßig am 1. Freitag eines Monats. Für das Jahr 2019 soll dafür eine Terminliste angefertigt werden. Diese soll wie folgt aussehen (s. Abbildung 1):

Wir treffen uns im Jahr 2019 am:
Freitag, den 04.01.2019
Freitag, den 08.02.2019
Freitag, den 08.03.2019
Freitag, den 05.04.2019
Freitag, den 03.05.2019
Freitag, den 07.06.2019
Freitag, den 05.07.2019
Freitag, den 02.08.2019
Freitag, den 06.09.2019
Freitag, den 04.10.2019
Freitag, den 08.11.2019
Freitag, den 06.12.2019

Abbildung 1: Muster einer Terminliste für das Jahr 2019

6.2 Lösung

6.2.1 Benutzerdefinierte PHP-Funktion erstellen

Benötigt wird eine benutzerdefinierte PHP-Funktion mit dem Namen '*get_first_friday()*'. Dieser Funktion wird das relevante Jahr in der PHP-Variablen $yr übergeben, s. Listing 5:

```
function get_first_friday($yr) {
   // Aufgabe: Den ersten Freitag in einem Monat des übergebenen Jahres ermitteln.
   //Datum und Zeitzone auf Deutsch einstellen.
   setlocale(LC_TIME,'de_DE','German_Germany');
   date_default_timezone_set('Europe/Berlin');
   // Associativen Datenbereich für dt. Wochentage einrichten.
   $weekday =
   array('Mon'=>'Montag','Tue'=>'Dienstag','Wed'=>'Mittwoch','Thu'=>'Donnerstag',
         'Fri'=>'Freitag','Sat'=>'Samstag','Sun'=>'Sonntag');
   echo "Wir treffen uns im Jahr ". $yr ." am:<br>";
   for ($m=1;$m<=12;$m++) {                  // Alle Monate 'm' des Jahres 'yr' durchlaufen.
      $tm  = mktime(0,0,0,$m,1,$yr);         // Zeitstempel: Erster Tag im Monat des Jahres.
      $fri = strtotime('first friday this month'.$yr.'-'.$m);
      $dt  = date('d.m.Y', $fri);            // Freitagsdatum als Text.
      $dow = date('D',strtotime($dt));       // Kürzel des jeweiligen Wochentags (Mon, ... , Sun).
      echo $weekday[$dow] . ', den '.$dt;    // dt. Wochentag und Freitagsdatum.
      echo "<br>";
   }
}
```

Listing 5: Benutzerdefinierte PHP-Funktion 'get_first_friday()'

6.2.2 Benutzerdefinierte PHP-Funktion aufrufen

Der Aufruf der PHP-Funktion *'get_first_friday()*' erfordert zwei PHP-Codezeilen (s. Listing 6):

```
<?php
$yr = 2019;                 // Kalenderjahr
get_first_friday($yr);  //Aufruf dieser Funktion
?>
```

Listing 6: Aufruf der benutzerdefinierten PHP-Funktion 'get_first_friday()'

7 Start- und Enddatum einer Kalenderwoche im Jahr bestimmen

7.1 Benutzerdefinierte PHP-Funktion erstellen

Das Start- und Enddatum einer bestimmten Woche eines Kalenderjahres kann mit der benutzerdefinierten PHP-Funktion '*Kalenderwoche_von_bis()*' bestimmt werden. An diese Funktion müssen ein Kalenderjahr und eine gültige Kalenderwoche übergeben werden. Doppelt zur Anwendung kommen dabei zwei eingebauten PHP-Funktionen (s. Listing 7):

- *date* Formatiert ein Datums- oder eine Zeitangabe.
- *strtotime* Wandelt ein beliebiges in englischer Schreibweise angegebenes Datum in einen Zeitstempel (engl. *timestamp*) um. Das Kürzel 'W' hinter dem Bindestrich symbolisiert eine best. Kalenderwoche (engl. *Week*) im Jahr. Diese muss zweistellig sein.

```
function Kalenderwoche_von_bis($jahr, $woche) {
   return  'Montag, den ' .date("d.m.Y",strtotime("{$jahr}-W{$woche}")).' bis '.
           'Sonntag, den '.date("d.m.Y",strtotime("{$jahr}-W{$woche}-7"));
}
```

Listing 7: Datumsbereich einer Kalenderwoche ermitteln und zurückgeben

Bei Vorgabe der 1. Kalenderwoche des Jahres 2019 lautet der Rückgabewert der Funktion:

```
Die Kalenderwoche 01 des Jahres 2019 fällt in diesen Datumsbereich:
Montag, den 31.12.2018 bis Sonntag, den 06.01.2019
```

Abbildung 2: Rückgabe der benutzerdefinierten PHP-Funktion 'Kalenderwoche_von_bis()'

7.2 Benutzerdefinierte PHP-Funktion aufrufen

Zum Aufruf der oben genannten Funktion (s. Listing 7) sind folgende PHP-Anweisungen erforderlich:

```
<?php
// Das relevante Kalenderjahr bestimmen.
$jahr = date('Y',strtotime('this year'));
// Zahl der Wochen im Kalenderjahr bestimmen:
// Der 28. Dezember liegt immer in der letzten Kalenderwoche eines Jahres.
$kw_im_jahr = date('W', strtotime('28 December $jahr'));
// Kalenderwoche vorgeben.
$kw = '52';
// Vorgegebene Kalenderwoche auf Gültigkeit prüfen und ggf. berichtigen.
$kw = ($kw > $kw_im_jahr) ? $kw_im_jahr : $kw;
// Start- und Enddatum der gewählten Kalenderwoche ausgeben.
echo 'Die Kalenderwoche '.$kw .' des Jahres '.$jahr.' fällt in diesen
Datumsbereich: ';
echo Kalenderwoche_Von_bis($jahr, $kw);
?>
```

Zum benutzerfreundlichen Aufruf dieser benutzerdefinierten PHP-Funktion bietet sich die Erstellung eines HTML-Formular mit zwei Eingabefeldern und entsprechenden Plausibilitätsprüfungen an. Darauf wird hier aus Platzgründen verzichtet.

8 Jahre mit 53 Kalenderwochen ermitteln

8.1 Kalenderwochen listen

In diesem und im folgenden Gliederungspunkt (s. 8.2, S. 5) wird die eingebaute PHP-Funktion *'strtotime()*' erneut angewandt.

Mit einer Zählschleife von 2009 bis 2021 sollen alle Jahre gelistet werden, die 53 Kalenderwochen umfassen:

```
<?php
for ($yr=2009;$yr<=2021;$yr++){
   $date_string = $yr."-12-28";
   $kw = date("W",strtotime($date_string));
   if ($kw == '53') {
      echo "Das Jahr ".$yr." hat ".$kw.(" Kalenderwochen<br>";
   }
}
?>
```

Listing 8: Jahre mit 53 Kalenderwochen ermitteln und ausgeben

Bei Verwendung der PHP-Funktion *strtotime()* ergibt sich:

Das Jahr 2009 hat 53 Kalenderwochen
Das Jahr 2015 hat 53 Kalenderwochen
Das Jahr 2020 hat 53 Kalenderwochen

Abbildung 3: Jahre mit 53 Kalenderwochen

8.2 Kalenderwochen überprüfen

Überprüfung, ob die Kalenderwoche 53 des Jahres 2020 am Montag, den 28.12.2020 beginnt:

```
<?php
$kw = new DateTime('2020-W53');    // Die Kalenderwoche muss zweistellig sein!
$dt = date_format($kw,'d.m.Y');    // Zugehöriges Datum ermitteln und formatieren.
$parts = explode('.',$dt);         // Datum in seine 3 Bestandteile zerlegen.
// Zeitstempel (engl. timestamp) mit der PHP-Funktion 'strtotime' erzeugen.
$timestamp = strtotime($parts[2].'-'.$parts[1].'-'.$parts[0]);
// Ergebnis: Mon, den 28.12.2020.
echo date('D',$timestamp).", den ".$dt;
?>
```

Listing 9: Starttag und -datum der KW 53 des Jahres 2020 überprüfen

Alternativ kann die Überprüfung mit dieser benutzerdefinierten PHP-Funktion erfolgen (s. Listing 10):

```
<?php
function StartAndEndOfWeek($jahr,$kw){
   $dt = new DateTime();   // DateTime-Objekt erstellen.
   // Die PHP-Funktion setISODate() benötigt 3 Argumente: Jahr, Woche u. Tag.
   return "Erster Wochentag: ".$dt->setISODate($jahr,$kw,"1")->format('d.m.Y')
      ."; Letzter Wochentag: ".$dt->setISODate($jahr,$kw,"7")->format('d.m.Y');
}
?>
```

Listing 10: Start- und Enddatum einer Kalenderwoche im Jahr ermitteln

Die PHP-Anweisung `echo StartAndEndOfWeek(2020,53)` bestätigt das erwartete Ergebnis:

```
Erster Wochentag: 28.12.2010; Letzter Wochentag: 03.01.2021
```

Abbildung 4: Start- und Enddatum der Kalenderwoche 53 im Jahr 2010

9 Literatur

[1] o.V., „Relative Formats,“ [Online]. Available: https://www.php.net/manual/de/datetime.formats.relative.php. [Zugriff am 11 06 2019].

[2] o. V., „Start und Enddatum einer Woche ermitteln,“ 02 12 2011. [Online]. Available: https://www.php-space.info/php-tutorials/113-start,und,enddatum,einer,woche,ermitteln.html. [Zugriff am 11 06 2019].

[3] o. V., „PHP: Get the first Monday of a month (and more),“ [Online]. Available: https://thisinterestsme.com/php-get-first-monday-of-month/. [Zugriff am 11 06 2019].