Lekcja 7. Obsługa daty i czasu

Wyświetlanie daty i czasu

W trakcie pisania skryptów często będziemy chcieli uzyskać aktualną datę i (lub) czas. PHP dysponuje na szczęście odpowiednimi funkcjami, które pozwalają na wykonywanie tego typu operacji. Podstawową funkcją pozwalającą na uzyskiwanie informacji o dacie jest `date`. Jej wywołanie ma postać:

```
date(format[, znacznik_czasu])
```

Zwraca ona ciąg znaków we wskazanym formacie. Parametr *format* określa, jakie dane nas interesują, natomiast *znacznik_czasu* to argument opcjonalny[26] zawierający znacznik czasu Uniksa (ang. *timestamp*) i określający interesującą nas datę. Znacznik czasu w rzeczywistości określa liczbę sekund, które upłynęły od północy (godzina 0:00:00) 1 stycznia 1970 roku. Ten sposób określania daty jest powszechnie wykorzystywany w systemach z rodziny Unix. Na razie jednak nie musimy go stosować. Jeśli bowiem parametr *znacznik_czasu* zostanie pominięty, funkcja `date` będzie wykorzystywała czas bieżący. Skoro tak, najważniejszy dla nas w tej chwili jest parametr *format*, który jest ciągiem znaków mogącym zawierać znaczniki przedstawione w tabeli 2.19.

Tabela 2.19. *Znaczniki formatujące dla funkcji date*

Znacznik	Opis	Przykładowe wartości
`a`	Określenie przed południem (`am`) lub po południu (`pm`), małymi literami	`am`, `pm`
`A`	Określenie przed południem (`AM`) lub po południu (`PM`), wielkimi literami	`AM`, `PM`
`B`	Czas internetowy w formacie `Swatch`	od `000` do `999`
`c`	Data zgodna z formatem ISO 8601	`2019-10-12T15:19:21+00:00`
`d`	Dzień miesiąca w formacie dwucyfrowym (z zerem na początku)	od `01` do `31`
`e`	Identyfikator strefy czasowej	`UTC`, `CET`, `Europe/Warsaw`
`D`	Dzień tygodnia w formie skrótu trzyliterowego	od `Mon` do `Sun`
`F`	Pełna angielska nazwa miesiąca	od `January` do `December`
`g`	Godzina w formacie dwunastogodzinnym (bez zera na początku)	od `1` do `12`
`G`	Godzina w formacie dwudziestoczterogodzinnym (bez zera na początku)	od `1` do `24`
`h`	Godzina w formacie dwunastogodzinnym (z zerem na początku)	od `01` do `12`

[26] Argumenty czy też szerzej, wszelkie konstrukcje języka, które są opcjonalne, w opisach ujmuje się w nawias kwadratowy.

Tabela 2.19. *Znaczniki formatujące dla funkcji date — ciąg dalszy*

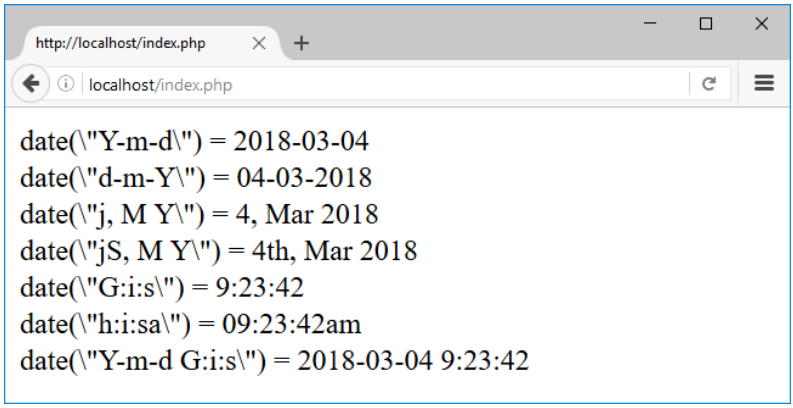
Znacznik	Opis	Przykładowe wartości
H	Godzina w formacie dwudziestoczterogodzinnym (z zerem na początku)	od 01 do 24
i	Liczba minut (z zerem na początku)	od 01 do 59
I	Znacznik czasu zimowego (1) i letniego (0)	0 lub 1
j	Dzień miesiąca (bez zera na początku)	od 1 do 31
l	Pełna nazwa dnia tygodnia	od Monday do Sunday
L	Znacznik roku przestępnego (1 oznacza rok przestępny)	0 lub 1
m	Miesiąc w postaci dwucyfrowej (z zerem na początku)	od 01 do 12
M	Skrót nazwy miesiąca (trzyliterowy)	od Jan do Dec
n	Miesiąc w postaci liczbowej (bez zera na początku)	od 1 do 12
N	Dzień tygodnia zgodnie ze standardem ISO-8601; znacznik dostępny od wersji 5.1.0	od 1 (poniedziałek) do 7 (niedziela)
o	Rok zgodnie ze standardem ISO-8601; znacznik dostępny od wersji 5.1.0	2016, 2019
O	Różnica w stosunku do czasu Greenwich (GMT)	+0200
P	Różnica w stosunku do czasu Greenwich (GMT) z dwukropkiem jako separatorem godzin i minut; znacznik dostępny od wersji 5.1.3	+02:00
r	Data w postaci zgodnej ze standardem opisanym w RFC 2822	Sun, 22 Dec 2019 16:01:07 +0200
s	Liczba sekund (z zerem na początku)	od 00 do 59
S	Przyrostek literowy dla liczby określającej dzień tygodnia (dla jęz. ang.)	st, nd, rd lub th
t	Liczba dni w podanym miesiącu	od 28 do 31
T	Strefa czasowa	np. CET, EST, MDT
u	Liczba mikrosekund; znacznik dostępny od wersji 5.2.2[27]	274258
U	Znacznik czasu Uniksa	liczba sekund, które upłynęły od 1 stycznia 1970 00:00:00 GMT
w	Dzień tygodnia w postaci numerycznej	od 0 (niedziela) do 6 (sobota)
W	Numer tygodnia w roku zgodnie ze standardem ISO-8601; znacznik dostępny od wersji 4.1.0.	od 1 do 52
y	Rok w postaci dwucyfrowej	np. 19
Y	Rok w postaci czterocyfrowej	np. 2019
z	Numer kolejnego dnia w roku (numeracja od 0)	od 0 do 365
Z	Przesunięcie strefy czasowej względem południka zerowego (w sekundach)	np. –43200, 43200

[27] Uwaga: funkcja date zawsze generuje ciąg zer dla mikrosekund. Właściwe mikrosekundy można uzyskać za pomocą wywołania DateTime::format (statyczna metoda format z klasy DateTime).

Jeżeli w ciągu formatującym znajdą się inne znaki niż wymienione w tabeli 2.19., zostaną one wyświetlone w niezmienionej postaci. Aby się przekonać, jakie efekty daje użycie różnych znaczników, można uruchomić skrypt widoczny na listingu 2.60. Wyświetla on przykładowe ciągi formatujące oraz odpowiadające im daty i (lub) czasy. Efekt działania kodu został zaprezentowany na rysunku 2.32.

Listing 2.60. *Przykładowe wywołania funkcji date*

```
<?php
  echo 'date(\"Y-m-d\") = ' . date("Y-m-d") . '<br>';
  echo 'date(\"d-m-Y\") = ' . date("d-m-Y") . '<br>';
  echo 'date(\"j, M Y\") = ' . date("j, M Y") . '<br>';
  echo 'date(\"jS, M Y\") = ' . date("jS, M Y") . '<br>';
  echo 'date(\"G:i:s\") = ' . date("G:i:s") . '<br>';
  echo 'date(\"h:i:sa\") = ' . date("h:i:sa") . '<br>';
  echo 'date(\"Y-m-d G:i:s\") = ' . date("Y-m-d G:i:s") . '<br>';
?>
```



Rysunek 2.32. *Efekt działania różnych ciągów formatujących w funkcji date*

Drugą funkcją pozwalającą na pobranie informacji dotyczących daty i czasu jest `getdate`. Jej wywołanie ma postać:

```
getdate([znacznik_czasu])
```

Również tutaj parametr *znacznik_czasu* jest opcjonalny, a jego użycie ma takie samo znaczenie jak w przypadku `date`. Wynikiem działania `getdate` nie jest jednak ciąg znaków, ale tablica asocjacyjna (lekcja 9.) zawierająca pobrane dane. Indeksy tej tablicy wraz z ich znaczeniami i przykładowymi wartościami wymieniono w tabeli 2.20.

Tabela 2.20. *Indeksy tablicy zwracanej przez funkcję getdate*

Nazwa indeksu	Znaczenie	Przykładowe wartości
`seconds`	Liczba sekund	od 0 do 59
`minutes`	Liczba minut	od 0 do 59
`hours`	Godzina	od 0 do 23
`mday`	Dzień miesiąca	od 1 do 31

Tabela 2.20. *Indeksy tablicy zwracanej przez funkcję getdate — ciąg dalszy*

Nazwa indeksu	Znaczenie	Przykładowe wartości
`wday`	Dzień tygodnia w postaci numerycznej	od `0` (niedziela) do `6` (sobota)
`mon`	Miesiąc w postaci liczbowej	od `1` do `12`
`year`	Rok w postaci czterocyfrowej	1995, 2018
`yday`	Numer kolejnego dnia w roku	od `0` do `365`
`weekday`	Dzień tygodnia w postaci tekstowej (dla jęz. ang.)	`Sunday`, `Monday`
`month`	Miesiąc w postaci tekstowej (dla jęz. ang.)	`January`, `February`
`0`	Aktualny znacznik czasu Uniksa	zazwyczaj od `-2147483648` do `2147483647`

W skrypcie widocznym na listingu 2.61 zobrazowano, w jaki sposób wykorzystać `getdate` do wyświetlenia informacji o aktualnej dacie. Tablica zwrócona przez funkcję jest przypisywana zmiennej `$data`, a wartości odczytane z indeksów `mday`, `mon` i `year` są przypisywane zmiennym pomocniczym `$dzien`, `$miesiac`, `$rok`. W przypadku gdy wartości zapisane w `$dzien` i `$miesiac` są mniejsze od `10`, jest do nich dodawany znak `0`, tak aby były zawsze prezentowane w formie dwucyfrowej (jednocześnie następuje wtedy konwersja na typ `string`). Ostatecznie wszystkie zmienne są łączone w jeden ciąg znaków i wysyłane do przeglądarki za pomocą instrukcji `echo`.

Listing 2.61. *Wykorzystanie funkcji getdate*

```
<?php
  $data = getdate();
  $dzien = $data["mday"];
  $miesiac = $data["mon"];
  $rok = $data["year"];
  if($dzien < 10) $dzien = "0" . $dzien;
  if($miesiac < 10) $miesiac = "0" . $miesiac;
  echo "Dziś jest $dzien-$miesiac-$rok r.";
?>
```

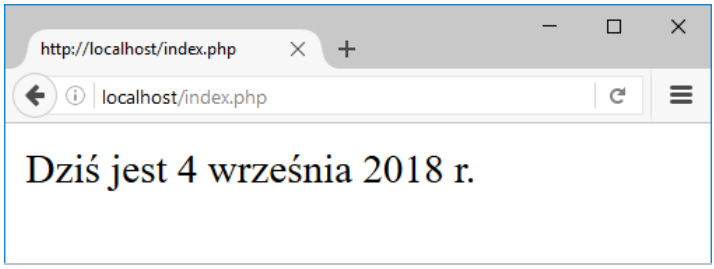
Jak wynika z tabeli 2.20, istnieje również możliwość uzyskania nazwy miesiąca w postaci tekstowej. Niestety dotyczy to jedynie języka angielskiego. Jeśli chcemy otrzymać polskie określenia, możemy samodzielnie przygotować odpowiednią procedurę. W tym celu wartość zwróconą przez wywołanie `getdate` pod indeksem `mon` należy przetworzyć tak, aby zawierała polską nazwę miesiąca. Jednym ze sposobów jest użycie instrukcji wyboru `switch…case`, tak jak zaprezentowano to na listingu 2.62. Zmienna `$miesiac` występuje tu w podwójnej roli — najpierw przechowuje numer miesiąca, a następnie (po wykonaniu instrukcji `switch`) — ciąg znaków określający jego nazwę. Przykładowy efekt działania skryptu zaprezentowano na rysunku 2.33.

Listing 2.62. *Uzyskanie polskich nazw miesięcy*

```
<?php
  $data = getdate();
  $dzien = $data["mday"];
  $miesiac = $data["mon"];
  $rok = $data["year"];
```

```
  switch($miesiac){
    case 1 : $miesiac = "stycznia"; break;
    case 2 : $miesiac = "lutego"; break;
    case 3 : $miesiac = "marca"; break;
    case 4 : $miesiac = "kwietnia"; break;
    case 5 : $miesiac = "maja"; break;
    case 6 : $miesiac = "czerwca"; break;
    case 7 : $miesiac = "lipca"; break;
    case 8 : $miesiac = "sierpnia"; break;
    case 9 : $miesiac = "września"; break;
    case 10 : $miesiac = "października"; break;
    case 11 : $miesiac = "listopada"; break;
    case 12 : $miesiac = "grudnia"; break;
  }

  echo "Dziś jest $dzien $miesiac $rok r.";
?>
```

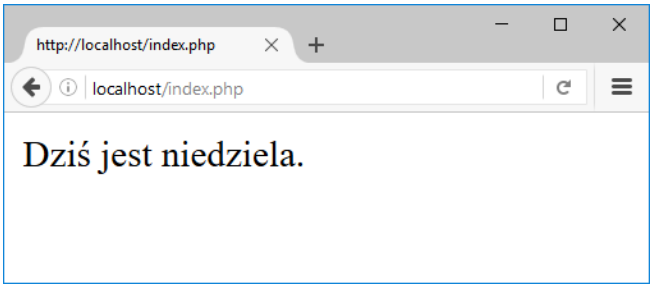
Rysunek 2.33. *Skrypt wyświetlający datę z polską nazwą miesiąca*



W podobny sposób uzyskamy polską nazwę aktualnego dnia tygodnia. Można w tym celu użyć zarówno funkcji `getdate`, jak i `date`. Tym razem skorzystamy więc z tej drugiej możliwości. Skrypt wykonujący takie zadanie jest widoczny na listingu 2.63, a przykładowy efekt jego działania — na rysunku 2.34. Podobnie jak w poprzednim przykładzie, pomocnicza zmienna `$dzienTygodnia` najpierw przechowuje numeryczną wartość określającą dzień, a następnie — ciąg znaków opisujący nazwę dnia.

Listing 2.63. *Wyświetlenie nazwy aktualnego dnia tygodnia*

```
<?php
  $dzienTygodnia = date("w");

  switch($dzienTygodnia){
    case 0 : $dzienTygodnia = "niedziela"; break;
    case 1 : $dzienTygodnia = "poniedziałek"; break;
    case 2 : $dzienTygodnia = "wtorek"; break;
    case 3 : $dzienTygodnia = "środa"; break;
    case 4 : $dzienTygodnia = "czwartek"; break;
    case 5 : $dzienTygodnia = "piątek"; break;
    case 6 : $dzienTygodnia = "sobota"; break;
  }

  echo "Dziś jest $dzienTygodnia.";
?>
```



Rysunek 2.34. *Efekt działania skryptu z listingu 2.63*

Do uzyskiwania danych opisujących datę i czas w językach narodowych można również wykorzystać funkcję `strftime`, która zwraca ciąg znaków sformatowany zgodnie z szablonem przekazanym jako argument. Jej wywołanie ma postać:

```
strftime(format[, timestamp])
```

Argument *format* to ciąg znaków, który może zawierać znaczniki przedstawione w tabeli 2.21. Parametr *timestamp* jest opcjonalny. Pozwala on na uzyskanie ciągu znaków odpowiadającego konkretnej dacie. Jeśli zostanie pominięty, zostanie użyty bieżący czas lokalny.

Tabela 2.21. *Znaczniki formatujące dla funkcji strftime*

Znacznik	Opis	Przykładowe wartości
%a	Nazwa dnia tygodnia w postaci skróconej	Pn, Wt, Śr
%A	Pełna nazwa dnia tygodnia	poniedziałek, wtorek
%b	Nazwa miesiąca w postaci skróconej	sty, lut, mar
%B	Pełna nazwa miesiąca	styczeń, luty, marzec
%c	Data i czas w formacie bieżącej lokalizacji	2008-02-17 18:35:46
%C	Określenie wieku; rok podzielony przez 100 i zaokrąglony do liczby całkowitej z zakresu od 00 do 99	20, 21
%d	Dzień miesiąca w postaci numerycznej dwucyfrowej (z zerem na początku, jeśli konieczne)	od 01 do 31
%D	Znaczenie identyczne z %m/%d/%y	02/17/14
%e	Dzień miesiąca w postaci numerycznej, liczby jednocyfrowe poprzedzane są spacją (format nieobsługiwany w systemach Windows)	od 1 do 31
%F	Znaczenie identyczne z %Y-%m-%d	2014-02-17
%g	Znaczenie takie jak %G, bez uwzględnienia wieku	08, 09
%G	Czterocyfrowy rok powiązany z numerem tygodnia według standardu ISO-8601:1988	2008, 2009
%h	Znaczenie takie samo jak %b	sty, lut, mar
%H	Godzina w formacie dwudziestoczterogodzinnym, z zerem na początku (jeśli konieczne)	od 01 do 24
%I	Godzina w formacie dwunastogodzinnym, z zerem na początku (jeśli konieczne)	od 01 do 12

Tabela 2.21. *Znaczniki formatujące dla funkcji strftime — ciąg dalszy*

Znacznik	Opis	Przykładowe wartości
`%j`	Numer dnia w roku w formacie trzycyfrowym, z zerami na początku, jeśli konieczne	od `001` do `366`
`%l`	Godzina w formacie dwunastogodzinnym ze spacją poprzedzającą pojedyncze cyfry	od `1` do `12`
`%m`	Miesiąc w postaci numerycznej dwucyfrowej, z zerem na początku (jeśli konieczne)	od `01` do `12`
`%M`	Bieżąca liczba minut w formacie dwucyfrowym	od `01` do `59`
`%n`	Znak nowej linii	N/A
`%p`	Znacznik przed południem – po południu (wielkimi literami)	`AM`, `PM`
`%P`	Znacznik przed południem – po południu (małymi literami)	`am`, `pm`
`%r`	Znaczenie identyczne z `%I:%M:%S %p`	`03:17:26 PM`
`%R`	Znaczenie identyczne z `%H:%M`	`15:17`
`%S`	Bieżąca liczba sekund w formacie dwucyfrowym	od `00` do `59`
`%t`	Znak tabulacji	N/A
`%T`	Znaczenie identyczne z `%H:%M:%S`	`15:18:53`
`%u`	Numer dnia tygodnia według standardu ISO-8601, 1 oznacza poniedziałek, 7 — niedzielę	od `1` do `7`
`%U`	Numer tygodnia w roku, zaczynając od pierwszej niedzieli jako pierwszego dnia pierwszego tygodnia	od `01` do `53`
`%V`	Numer tygodnia w roku zgodnie ze standardem ISO8601:1998	od `01` do `53`
`%w`	Numer dnia tygodnia, 0 oznacza niedzielę, 6 — sobotę	od `0` do `6`
`%W`	Numer tygodnia w roku, zaczynając od pierwszego poniedziałku jako pierwszego dnia pierwszego tygodnia	od `01` do `53`
`%x`	Reprezentacja daty zgodna z bieżącymi ustawieniami lokalizacyjnymi	`2014-02-17`
`%X`	Reprezentacja czasu zgodna z bieżącymi ustawieniami lokalizacyjnymi	`19:02:19`
`%y`	Rok w postaci dwucyfrowej	od `00` do `99`
`%Y`	Rok w postaci czterocyfrowej	od `1901` do `2038`
`%z`	Przesunięcie dla bieżącej strefy czasowej lub skrót nazwy strefy czasowej (zależnie od systemu operacyjnego, na którym działa PHP)[28]	`+0200`, `CET`
`%Z`	Określenie strefy czasowej niezwracane przez znacznik `%z`[29]	`+0200`, `CET`
`%%`	Znak `%`	`%`

[28] W praktyce może zostać zwrócona również pełna nazwa strefy czasowej.

[29] W praktyce może zostać zwrócony również ten sam wynik co przy znaczniku `%z`.

Dane zwracane przez `strftime`, a więc m.in. nazwy dni tygodnia oraz miesięcy, będą zgodne z bieżącymi ustawieniami lokalnymi (narodowymi), które mogą być zmieniane za pomocą funkcji `setlocale`. Wywołanie `setlocale` ma postać:

```
setlocale(kategoria, locale);
```

Parametr *`kategoria`* wskazuje, jakie informacje mają być pobierane z uwzględnieniem ustawień narodowych, natomiast *`locale`* określa, który zestaw narodowy ma być użyty. Możliwe wartości parametru *`kategoria`* zostały zaprezentowane w tabeli 2.22. Argument *`locale`* jest wyrażany w różny sposób w różnych systemach operacyjnych. W systemach uniksowych jest on oparty na normach opisanych w dokumentach RFC 1766 oraz ISO 639. Dla ustawień polskich należy użyć ciągu `pl_PL`. W systemach z rodziny Windows stosowana jest inna konwencja nazewnictwa i dla ustawień polskich należy użyć ciągu `plk` lub `polish`. Powodem takich różnic jest to, że funkcja `setlocale` zawarta w PHP bazuje na funkcjach systemu operacyjnego, które zachowują się różnie w zależności od systemu.

Tabela 2.22. *Wartości pierwszego parametru funkcji setlocale*

Parametr	Opis
`LC_ALL`	Wpływa na wszystkie parametry.
`LC_COLLATE`	Wpływa na porównywanie ciągów znaków.
`LC_CTYPE`	Wpływa na klasyfikację i konwersję znaków.
`LC_MONETARY`	Wpływa na postać ciągów numerycznych i walutowych.
`LC_NUMERIC`	Wpływa na postać separatora dziesiętnego.
`LC_TIME`	Wpływa na postać daty i czasu.
`LC_MESSAGES`	Wpływa ma komunikaty systemowe[30].

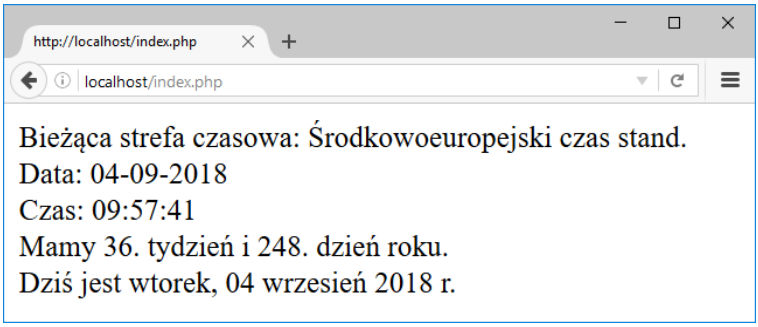
Warto zwrócić uwagę, że zachowanie `strftime` zależy również od tego, na jakim systemie operacyjnym działa PHP. O ile zatem przedstawione wcześniej na listingach 2.61 i 2.62 skrypty będą działały tak samo niezależnie od środowiska PHP, to w przypadku wykorzystania `strftime` już tak być nie musi. Po pierwsze zwracane wartości mogą być różne w zależności od systemu, po drugie, nie zawsze będą dostępne wszystkie znaczniki formatujące przedstawione w tabeli 2.21. Prosty skrypt wykorzystujący omawianą funkcję do wyświetlenia kilku podstawowych danych o dacie i czasie został zaprezentowany na listingu 2.64, a przykładowy efekt jego działania zilustrowano na rysunku 2.35 (widać, że przy automatycznym pobieraniu nazwy miesiąca nie zawsze uzyskamy poprawną polską odmianę, rysunek 2.31).

Listing 2.64. *Wykorzystanie funkcji setlocale*

```
<?php
  setlocale(LC_ALL, 'pl_PL');
  // W wersji dla systemu Windows należy użyć:
  // setlocale(LC_ALL, 'plk');
  echo strftime("Bieżąca strefa czasowa: %Z <br>");
  echo strftime("Data: %d-%m-%Y <br>");
```

[30] Opcja dostępna, jeśli PHP było kompilowane z biblioteką *libintl*.

```
    echo strftime("Czas: %H:%M:%S <br>");
    echo strftime("Mamy %U. tydzień i %j. dzień roku. <br>");
    echo strftime("Dziś jest %A, %d %B %Y r. <br>");
?>
```



Rysunek 2.35. *Efekt działania skryptu wykorzystującego funkcję strftime*

Tworzenie znacznika czasu

Do utworzenia znacznika czasu Uniksa, który może być wykorzystywany jako drugi parametr funkcji opisywanych w poprzedniej części lekcji, można wykorzystać funkcję `mktime`. W pełnej wersji przyjmuje ona aż sześć argumentów, chociaż w rzeczywistości wszystkie są opcjonalne. Schemat wywołania jest następujący:

```
mktime([godzina[, minuta[, sekunda[, miesiąc[, dzień[, rok]]]]]])
```

Znaczenie argumentów jest zgodne z ich nazwami. We wcześniejszych wersjach PHP istniał również siódmy parametr — `isdst`, który przyjmował wartość `1` dla czasu zimowego, `0` — dla letniego lub `-1`, jeżeli nie było wiadomo, jaki to jest czas. Od wersji PHP 5.1.0 został jednak uznany za schyłkowy (status *deprecated*), a w wersji 7 — całkowicie usunięty.

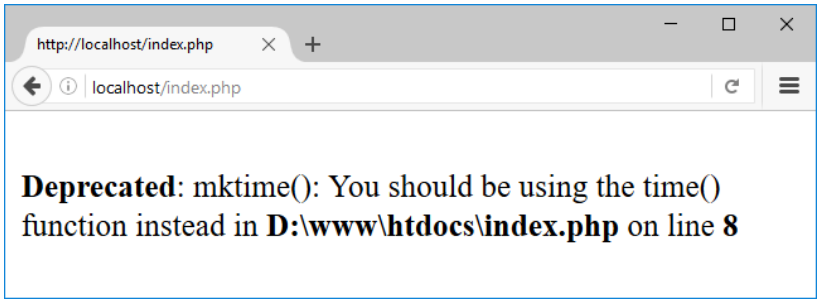
Parametry można pomijać od strony prawej do lewej, co oznacza, że jeśli chcemy zrezygnować z parametru *`miesiąc`*, to musimy również usunąć *`dzień`* i *`rok`*. Brakujące parametry są uzupełniane przez bieżący czas lokalny. Jeśli więc interesuje nas znacznik czasu odpowiadający godzinie 13:24:52 w aktualnym dniu, zastosujemy wywołanie:

```
mktime(13, 24, 52)
```

Jeśli zaś znacznik ma odpowiadać aktualnemu czasowi i dacie, należy pominąć wszystkie parametry:

```
mktime()
```

Należy przy tym zwrócić uwagę, że począwszy od PHP 5.1.0 pominięcie wszystkich argumentów (choć dopuszczalne) spowoduje wygenerowanie komunikatu sugerującego użycie funkcji `time` zamiast `mktime` (rysunek 2.36). Taki komunikat będzie jednak widoczny tylko wtedy, gdy w pliku konfiguracyjnym *php.ini* została włączona opcja raportowania błędów `E_STRICT` (zmienna *error_reporting* w sekcji *Error handling and logging*).



Rysunek 2.36. *Komunikat systemowy po użyciu funkcji mktime bez argumentów wywołania*

Parametr *rok* może być podawany w postaci dwu- lub czterocyfrowej. W pierwszym przypadku wartości od 0 do 69 są przekładane na lata 2000 – 2069, natomiast wartości od 70 do 99 — na lata 1970 – 1999. W tym miejscu trzeba zaznaczyć, że w większości systemów 32-bitowych maksymalny możliwy zakres dat to lata 1901 – 2038, co wynika z tego, że znacznik czasu jest reprezentowany jako 32-bitowa liczba ze znakiem. Może więc on przyjmować wartości od -2^{31} do $2^{31}-1$. Skoro taka wartość oznacza liczbę sekund, które upłynęły od 1 stycznia 1970 (od tzw. epoki Uniksa, ang. *Unix epoch*), zakres dat musi być ograniczony. Poza tym w niektórych wersjach systemu Windows nie są obsługiwane wartości ujemne znacznika, co w przypadku PHP w wersjach poniżej 5.1.0 ogranicza zakres do lat 1970 – 2038.

To, jak w praktyce można wykorzystać funkcję `mktime`, pokazuje skrypt widoczny na listingu 2.65. Pozwala on na obliczenie liczby dni występujących między dwiema datami. Przykładowo, pomiędzy 1 maja a 1 września 2018 roku.

Listing 2.65. *Skrypt obliczający liczbę dni pomiędzy dwiema datami*

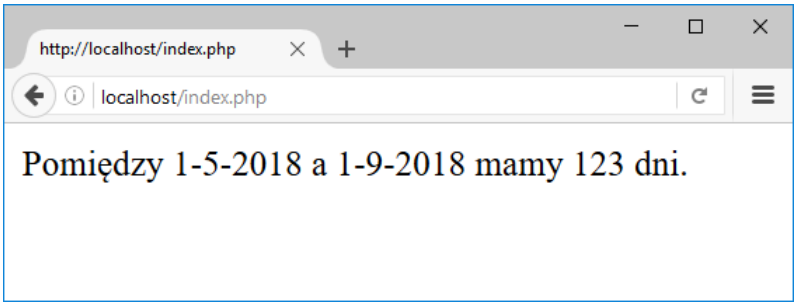
```
<?php
  $r1 = 2018;
  $m1 = 5;
  $d1 = 1;

  $r2 = 2018;
  $m2 = 9;
  $d2 = 1;

  $time1 = mktime(0, 0, 0, $m1, $d1, $r1);
  $time2 = mktime(0, 0, 0, $m2, $d2, $r2);
  $time = abs(ceil(($time1 - $time2) / 86400));

  echo "Pomiędzy $d1-$m1-$r1 a $d2-$m2-$r2 mamy $time dni.";
?>
```

Pierwsza data jest zapisywana w zmiennych `$r1`, `$m1`, `$d1`, a druga — `$r2`, `$m2`, `$d2`. Nietrudno się domyślić, że zmienne z literą `r` określają rok, z literą `m` — miesiąc, a z `d` — dzień. Obie daty są przetwarzane na odpowiadające im znaczniki czasu Uniksa, następnie jest obliczana ich różnica, czyli liczba występujących pomiędzy nimi sekund (`$time1 - $time2`). Po wyliczeniu tej wartości wystarczy zamienić sekundy na dni, aby odnaleźć ich szukaną liczbę. Taka zamiana jest prosta — dzielimy różnicę przez liczbę sekund w dobie, czyli 86 400 (60 sekund×60 minut×24 godziny). Ponieważ w za-

leżności od danych wejściowych uzyskana wartość może być ujemna oraz niecałkowita, stosujemy dodatkowo funkcję zaokrąglenia wyniku w górę (`ceil`) oraz funkcję podającą wartość bezwzględną argumentu (`abs`). Jeśli tak złożona instrukcja jest niezbyt czytelna, możemy ją rozbić na kilka etapów, np.:

```
$time = ($time1 - $time2) / 86400;
$time = ceil($time);
$time = abs($time);
```

Ostatecznie po uruchomieniu skryptu zobaczymy widok taki jak na rysunku 2.37.



Rysunek 2.37. *Liczba dni pomiędzy dwiema datami*

Znacznik czasu można również uzyskać poprzez konwersję ciągu znaków opisujących datę i (lub) czas. Służy do tego funkcja `strtotime`, której wywołanie jest następujące:

```
strtotime(format[, timestamp]);
```

Argument *`format`* to opis daty (ciąg znaków), dla której chcemy uzyskać znacznik czasu. Natomiast *`timestamp`* to opcjonalny znacznik czasu, który zostanie wykorzystany przy obliczaniu wyniku. Opis daty to angielski opis zgodny ze składnią GNU. Nie będziemy omawiać tu pełnej składni tego standardu, gdyż zajęłoby to zbyt wiele miejsca. Wszelkie niezbędne informacje można znaleźć w dokumentacji PHP w sekcji *Date and Time Related Extensions->Date/Time->Supported Date and Time Formats*. Pokazanych zostanie jednak kilka przykładów zastosowania. Jeśli więc chcemy uzyskać aktualny znacznik czasu, napiszemy:

```
strtotime("now");
```

Jeśli interesuje nas znacznik czasu dla 18 marca 1976 roku, napiszemy:

```
strtotime("18 march 1978");
```

Jeśli potrzebujemy znacznika czasu sprzed 12 dni, wpiszemy:

```
strtotime("-12 day");
```

Jeśli potrzebujemy znacznika czasu dla dnia, który nastąpi za 2 dni 4 godziny 15 minut i 12 sekund od aktualnej godziny, wpiszemy:

```
strtotime("+2 day 4 hours 15 minutes 12 seconds");
```

Jeśli chcemy uzyskać znacznik czasu dla następnego piątku, to wpiszemy:

```
strtotime("next friday")
```

Ostatnia funkcja, którą zajmiemy się w tym punkcie, to `microtime`. Ona również zwraca aktualny znacznik czasu, z tym że zawierający dodatkowo dane dotyczące liczby mikrosekund. Ogólne wywołanie ma postać:

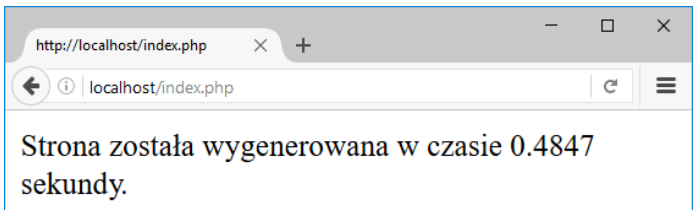
```
microtime([tryb])
```

Parametr *`tryb`* jest tu opcjonalnym parametrem typu `boolean`. W przypadku gdy występuje i jest równy `true`, zwracana jest wartość rzeczywista. W pozostałych sytuacjach zwracany jest ciąg w postaci `msec sec`, gdzie `msec` oznacza liczbę mikrosekund, a `sec` — liczbę sekund, które upłynęły od początku epoki Uniksa. Takie zachowanie pozwala np. na wykorzystanie tej funkcji do obliczenia czasu generowania strony. Sposób, w jaki można to zrobić, zaprezentowano w skrypcie widocznym na listingu 2.66.

Listing 2.66. *Skrypt obliczający czas generowania strony*

```
<?php
  $time1 = microtime(true);

  // Przykładowa treść skryptu
  for($i = 0; $i < 10000; $i++)
    for($j = 0; $j < 1000; $j++);

  $time2 = microtime(true);
  $time = round($time2 - $time1, 4);

  echo "Strona została wygenerowana w czasie $time sekundy.";
?>
```

Przed wykonaniem głównego kodu strony wywoływana jest funkcja `microtime`, a wynik jej działania jest przypisywany zmiennej `$time1`. Po wykonaniu kodu strony, który w tym przypadku jest symulowany przez zagnieżdżone pętle `for`, ponownie jest wywoływana funkcja `microtime`, a zwrócony wynik działania jest zapisywany w zmiennej `$time2`. Oznacza to, że czas generowania strony to nic innego jak `$time2 - $time1`. Wynik jest zaokrąglany za pomocą funkcji `round` do czterech miejsc po przecinku i wyświetlany w przeglądarce, tak jak widać to na rysunku 2.38.



Rysunek 2.38. *Wynik działania skryptu podającego czas generowania strony*

Pozostałe funkcje

Oprócz wspomnianych wcześniej istnieje jeszcze wiele innych funkcji służących do obsługi daty i czasu. Nie ma potrzeby, aby je wszystkie omawiać — w tej części lekcji opisanych zostanie jedynie kilka z nich. Będą to:

- checkdate,
- gmdate,
- localtime,
- time.

W razie potrzeby dodatkowe informacje można znaleźć w dokumentacji PHP w sekcji *Date and Time Related Extensions->Date/Time->Functions*.

Funkcja checkdate

Zadaniem funkcji `checkdate` jest sprawdzenie, czy dane przekazane jej w postaci argumentów tworzą poprawną datę. Jej wywołanie ma postać:

```
checkdate(miesiąc, dzień, rok)
```

Znaczenie argumentów jest zgodne z ich nazwami, wszystkie są typu `integer`. Funkcja zwraca wartość `true`, jeżeli podany dzień, miesiąc i rok tworzą poprawną datę, lub `false` — w przeciwnym razie. Należy zwrócić uwagę na kolejność argumentów — jest ona zgodna z amerykańskim formatem prezentacji daty, w którym miesiąc znajduje się na pierwszym miejscu. Przykład wywołania (dla 25 maja 2018 roku):

```
checkdate(5, 25, 2018)
```

Funkcja gmdate

Funkcja `gmdate` zwraca ciąg znaków sformatowany zgodnie z szablonem podanym jako parametr *`format`*. Jej wywołanie ma postać:

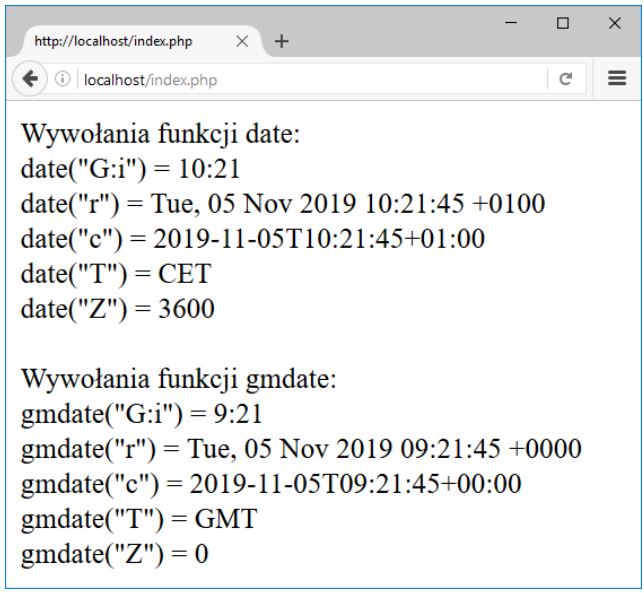
```
gmdate(format[, timestamp])
```

Jeśli podamy parametr *`timestamp`*, zostanie zwrócona odpowiadająca mu data. Jeżeli parametr ten zostanie pominięty, zostanie użyty aktualny czas. W odróżnieniu od omawianej wcześniej `date` zwracany jest czas GMT (UTC), a nie — lokalny. Parametr *`format`* może zawierać znaczniki formatujące podane w tabeli 2.19 przy opisie funkcji `date`. Różnice w wywołaniu `date` i `gmdate` zostaną pokazane po uruchomieniu skryptu z listingu 2.67. Przykładowy efekt jego działania jest widoczny na rysunku 2.39.

Listing 2.67. *Porównanie funkcji date z gmdate*

```
<?php
  echo "Wywołania funkcji date:<br>";
  echo "date(\"G:i\") = " . date("G:i") . "<br>";
  echo "date(\"r\") = " . date("r") . "<br>";
  echo "date(\"c\") = " . date("c") . "<br>";
  echo "date(\"T\") = " . date("T") . "<br>";
  echo "date(\"Z\") = " . date("Z") . "<br>";

  echo "<br>Wywołania funkcji gmdate:<br>";
  echo "gmdate(\"G:i\") = " . gmdate("G:i") . "<br>";
  echo "gmdate(\"r\") = " . gmdate("r") . "<br>";
```

```
    echo "gmdate(\"c\") = " . gmdate("c") . "<br>";
    echo "gmdate(\"T\") = " . gmdate("T") . "<br>";
    echo "gmdate(\"Z\") = " . gmdate("Z") . "<br>";
?>
```



Rysunek 2.39. *Wyniki działania funkcji date i gmdate*

Uwaga: aby efekt był prawidłowy, w konfiguracji PHP musi być ustawiona lokalna strefa czasowa inna niż UTC (np. *Europe/Warsaw*). Jeśli wyniki działania obu funkcji są takie same, oznacza to brak ustawionej strefy czasowej lub ustawienie strefy na UTC. W takim przypadku należy zmienić ustawienie opcji `date.timezone` w pliku konfiguracyjnym *php.ini* (np. `date.timezone = Europe/Warsaw`).

Funkcja localtime

Funkcja `localtime` zwraca tablicę, której poszczególne elementy zawierają dane dotyczące daty i czasu (zawartość tablicy jest taka sama jak struktury zwracanej po wywołaniu funkcji `localtime` w języku C). Wywołanie funkcji w PHP ma postać:

```
localtime([timestamp[, is_associative]])
```

Argumentem pierwszym jest uniksowy znacznik czasu. Jeżeli zostanie pominięty, to będzie uwzględniony bieżący czas lokalny. Ustawienie argumentu *`is_associative`* na `false` lub niepodanie go powoduje zwrócenie zwykłej tablicy indeksowanej liczbowo. Ustawienie na `true` powoduje z kolei zwrócenie tablicy asocjacyjnej o następujących kluczach:

- `tm_sec` — liczba sekund,
- `tm_min` — liczba minut,
- `tm_hour` — godzina,

- `tm_mday` — dzień miesiąca w postaci liczbowej,
- `tm_mon` — numer miesiąca w roku w postaci liczbowej,
- `tm_year` — liczba lat, które upłynęły od roku 1900,
- `tm_wday` — dzień tygodnia,
- `tm_yday` — dzień roku,
- `tm_isdst` — znacznik czasu letniego i zimowego.

Funkcja time

Funkcja `time` zwraca znacznik czasu Uniksa określający aktualną datę i czas. Wywołanie ma postać:

```
time()
```

Ćwiczenia do samodzielnego wykonania

Ćwiczenie 7.1. Korzystając z funkcji `date`, napisz skrypt wyświetlający następujące dane: nazwę dnia tygodnia (w jęz. polskim), pełną datę (rok w postaci czterocyfrowej), czas w formacie dwunastogodzinnym z oznaczeniem przed południem/po południu oraz różnicę w stosunku do czasu GMT (UTC).

Ćwiczenie 7.2. Napisz skrypt, który będzie wyświetlał liczbę dni pozostałych do końca bieżącego roku.

Ćwiczenie 7.3. Napisz skrypt pozwalający stwierdzić, którym dniem tygodnia był dowolnie wybrany dzień.

Ćwiczenie 7.4. Napisz skrypt obliczający czas generowania strony, w którym wartości sekund i mikrosekund są wyświetlane oddzielnie.

Ćwiczenie 7.5. Napisz skrypt, który umożliwi wczytywanie różnych wersji strony w zależności od pory dnia. Wykorzystaj funkcję `date` oraz instrukcję `include`.