Bab 3

Memanipulasi dan Mengkonversi Data

Setiap kali berurusan dengan data, hampir selalu harus melakukan konversi untuk mengubah data dari satu bentuk atau format ke bentuk lain. Di level paling bawah, bahasa scripting seperti PHP sudah memberikan kenyamanan seperti konversi otomatis antara angka menjadi string dan sebaliknya. Tapi ada banyak tugas lain yang sehari-hari tetap perlu kita lakukan, seperti mengonversi data tanggal dari atau ke format string atau mengubah angka menjadi teks dalam bahasa Indonesia. Resep-resep pada bab ini berurusan dengan konversi dan manipulasi yang umum dilakukan dari satu bentuk ke bentuk lain.

Resep 3-1: Konversi Tanggal dan Waktu

Banyak pemula yang mengalami kesulitan dalam memanipulasi data waktu ataupun tanggal. Ini dikarenakan dua hal. Pertama, waktu dan tanggal adalah data yang berstruktur. Waktu terdiri dari "jam:menit:detik", sementara tanggal terdiri dari "hari/bulan/tahun". Kedua, sistem waktu dan tanggal tidak berbasis desimal. Waktu misalnya, berbasis 60 untuk menit dan detik. Minggu berbasis 7, bulan berbasis 28 hingga 31, dan seterusnya.

Belum lagi ada beberapa format tampilan tanggal dan waktu yang digunakan, sehingga untuk mem-parse string berisi tanggal atau waktu memang tidak sederhana.

Prinsip pertama dalam melakukan perhitungan waktu adalah mengonversi dulu dalam satuan timestamp. Misalnya Unix timestamp yang formatnya dalam satuan detik dan berupa angka. Sehingga dapat ditambah dan dikurangi. Keuntungan lainnya, adalah bebas ketergantungan zona waktu. Unix timestamp didefinisikan sebagai jumlah detik setelah 1 Jan 1970 00:00 UTC.

Jika ada 2 waktu memiliki timestamp yang sama, maka kedua waktu itu adalah sama, tak peduli pada zona waktu. Ini berbeda dengan string waktu "10:10:10" misalnya, yang waktu absolutnya tidaklah sama di zona waktu yang berbeda.

Dua fungsi utama di PHP untuk manipulasi waktu adalah: `strtotime()`. Fungsi ini dapat mem-parse waktu dalam berbagai bentuk tampilan menjadi Unix timestamp. Fungsi kedua adalah `date()` atau `gmdate()` yang akan mengonversi balik Unix timestamp ke string berformat tertentu.
Ingat, semua aritmetika waktu sendiri dilakukan dalam format Unix timestamp dengan aritmetika angka biasa.

Mengurangi atau menambah tanggal dengan jumlah durasi tertentu

Misalnya, Anda memiliki string tanggal "10 Des 2003 10:10:10" dan ingin tahu berapa tanggal dan waktu 7 hari 8 jam 9 menit dari tanggal tersebut.

```
$tanggal = 10 Dec 2003 10:10:10";
if (($timestamp = strtotime($tanggal)) == -1)
  die("Gagal memparse tanggal!");
echo "7 hari 8 jam 9 menit dari $tanggal adalah: ",
  date("d M Y H:i:s", $timestamp + 7*86400 + 8*3600 + 9*60);
```


Hasilnya adalah:

```
17 Dec 2003 18:19:10
```

Fungsi `strtotime()` menerima sebuah string tanggal/waktu dan akan mengembalikan timestamp Unix; jika gagal (mis: kalau format tanggal/waktu tidak benar, seperti "-1 Des 2003") maka fungsi akan mengembalikan nilai -1. Kita lalu menambah pada timestamp ini sejumlah waktu dalam satuan detik (86400 detik = 1 hari; 3600 detik = 1 jam; 60 detik = 1 menit).

Kalau ingin mengetahui tanggal berapa 7 hari 8 jam 9 menit dari *sekarang*, Anda bisa menggunakan:

```
echo "7 hari 8 jam 9 menit dari sekarang adalah: ",
  date("d M Y H:i:s", time() + 7*86400 + 8*3600 + 9*60);
```

Ini karena `time()` sudah mengembalikan waktu saat ini dalam format Unix timestamp.

Mengetahui day of week

Misalnya ingin mengetahui hari jatuhnya tanggal 25 Des 2004, konversi tanggal ke timestamp lalu format menggunakan `date()` dengan lambang format "l" (huruf l kecil).

```
$tanggal = "25 Dec 2004";

$timestamp = strtotime($tanggal);
echo "$tanggal = ".date("l Ymd His", $timestamp);
```

Hasilnya:

```
25 Dec 2004 = Saturday
```

Supaya hasilnya berbahasa Indonesia, gunakan `strftime()`, karena `date()` tidak mendukung locale:

```
$tanggal = "25 Dec 2004";

if (setlocale(LC_ALL, "id_ID")===false)
  die("Gagal setlocale!");
$timestamp = strtotime($tanggal);
echo "$tanggal = ".strftime("%A", $timestamp);
```

Hasilnya:

```
25 Dec 2004 = Sabtu
```

Mengekstrak elemen waktu

Misalnya ingin mengetahui tanggal 25 Des terdekat tinggal berapa hari lagi, contohnya dalam membuat skrip "countdown", maka Anda tidak bisa melakukan aritmetika biasa seperti pada contoh sebelumnya. Karena tanggal akhir tidak diketahui lengkap dan tahun tidak diketahui. Untuk mengatasi hal ini maka perlu dilakukan ekstraksi tanggal. Fungsi yang dapat digunakan untuk ekstraksi tanggal adalah `getdate()`.

Sebetulnya dengan `date()` atau `strftime()` pun bisa, namun `getdate()` memang khusus digunakan untuk ekstraksi tunggal ini.

```
<?

// ekstraksi tanggal/bulan/tahun dari tanggal saat ini
$now = time();
$timeelements = getdate($now);

// sekarang tentukan apakah ingin countdown ke 25 des tahun ini
atau tahun
// depan.
if ($timeelements['mon'] = 12 && $timeelements['mday'] >= 25)
  $year = $timeelements['year']+1;
else
  $year = $timeelements['year'];

// bentuk timestamp
$timestamp = mktime(0, 0, 0, 12, 25, $year);

echo "Saat ini ", date("d M Y H:i:s", $now), "<br>";
echo ($timestamp-$now), " detik lagi hingga Natal $year!";

?>
```

Contoh hasil:

```
Saat ini 10 Dec 2003 17:24:46
1233314 detik lagi hingga Natal 2003!
```


Fungsi `getdate()` menerima timestamp yang defaultnya adalah `time()`, dan mengembalikan array berisi elemen seperti `year`, `month`, `mday`, `hours`, `minutes`, `seconds`, dan sebagainya. Kita lalu menggunakan elemen tahun untuk pengujian.

Terakhir, kita bentuk kembali timestamp baru dari balik elemen-elemen waktu menggunakan fungsi `mktime()`.

Jika ingin menampilkan durasi waktu dalam satuan yang lebih human-friendly, gunakan fungsi berikut:

```
 1|<?
  |
 3|//
 4|// duration2text.php
 5|//
  |
 7|function duration2text($dur) {
 8|  $is_neg = $dur < 0 ? 1:0; $dur = abs($dur);
  |
10|  $years = floor($dur / (365*86400)); $dur -= $years *
  365*86400;
11|  $months = floor($dur / (30*86400)); $dur -= $months *
  30*86400;
12|  $days = floor($dur / 86400); $dur -= $days * 86400;
13|  $hours = floor($dur / 3600); $dur -= $hours * 3600;
14|  $minutes = floor($dur / 60); $dur -= $minutes * 60;
15|  $secs = $dur;
  |
17|  return join(" ", array(
18|    ($years ? "$years tahun" : ""),
19|    ($months ? "$months bulan" : ""),
20|    ($days ? "$days hari" : ""),
21|    ($hours ? "$hours jam" : ""),
22|    ($minutes ? "$minutes menit" : ""),
23|    "$secs detik",
24|    ($is_neg ? "lalu" : "lagi")
25|  ));
26|}
  |
28|?>
```

Hasilnya jika baris:

```
echo ($timestamp-$now), " detik lagi hingga Natal $year!";
```

diganti menjadi:

```
require_once "duration2text.php";
echo duration2text($timestamp-$now), " detik lagi hingga Natal
$year!";
```

adalah:

```
Saat ini 10 Dec 2003 17:36:27
14 hari 6 jam 23 menit 33 detik lagi detik lagi hingga Natal 2003!
```

Ada juga kasus lain. Misalnya Anda ingin mengetahui berapa jam, menit dan detik lagi hingga jam 06:00:00 esok hari, sedangkan durasi tidak diketahui secara pasti. Maka perlu mengekstrak data besok terlebih dahulu.

Gunakan contoh berikut ini:

```
<?

// peroleh elemen-elemen waktu untuk jam saat ini besok
$now = time();
$timeelements = getdate($now + 86400);

// ganti jam:menit:detik menjadi 06:00:00
$timestamp = mktime(6, 0, 0, $timeelements['mon'],
                    $timeelements['mday'],
 $timeelements['year']);

include_once "duration2text.php";
echo "Besok pukul 06:00:00 = ", duration2text($timestamp-$now);

?>
```

Hasilnya:

```
Besok pukul 06:00:00 = 12 jam 15 menit 42 detik lagi
```


Resep 3-2: Konversi Zona Waktu

Zona waktu juga adalah salah satu hal yang sering menjebak programer PHP jika bekerja dengan dua buah mesin berbeda. Misalnya komputer pribadi diset zona waktu lokal (Jakarta, GMT+7) sementara server tempat skrip PHP di-upload menggunakan zona waktu lokal setempat, misalnya Los Angeles, Pacific Time, GMT-8). Maka hal ini kana menyebabkan banyak kesalahan perhitungan waktu dan tanggal.

Ditambah lagi dengan jenis data `DATETIME` atau `TIMESTAMP` di MySQL yang tidak mengandung informasi zona waktu, maka skrip dan data yang di-upload bisa tampil tidak benar. Misalnya, skrip PHP menyapa pengunjung Indonesia "selamat pagi" padahal di Jakarta dan sekitarnya sudah malam.

Solusi teraman tentu saja selalu menggunakan fungsi seperti `gmdate()` dan `gmmktime()` dan bukannya `date()` dan `mktime()`. Namun fungsi ini tidak nyaman karena mengharuskan bekerja dengan GMT, sementara waktu lokal bukan GMT.

Jika Anda menggunakan `date()` maka untuk mengonversi sebuah tanggal/waktu dari zona waktu yang satu ke yang lain adalah sebagai berikut:

```
<?

$SERVER_TIMEZONE = date("Z"); // zona waktu di server
$WANTED_TIMEZONE = 7*3600;    // zona waktu yg ingin
 ditampilkan, GMT+7

// timestamp yang ingin dikonversi
$timestamp = time();

// tampilkan tanggal dalam zona waktu $WANTED_TIMEZONE
echo "Waktu saat ini: ",date("d M Y H:i:s",
  $timestamp - $SERVER_TIMEZONE + $WANTED_TIMEZONE);

?>
```

Hasilnya dalam waktu Indonesia, tak peduli tempat server pagi atau malam:

```
Waktu saat ini: 10 Dec 2003 18:03:47
```

Resep 3-3: Terbilang

Setiap kali kita berurusan dengan aplikasi billing atau invoicing atau akuntansi, hampir selalu kebutuhan untuk mengonversi angka 12.500 menjadi "dua belas ribu lima ratus [rupiah]" muncul.

Berikut ini implementasinya dalam PHP:

```
<?

//
// terbilang.php
//

function terbilang($num) {
  $digits = array(
    0 => "nol",
    1 => "satu",
    2 => "dua",
    3 => "tiga",
    4 => "empat",
    5 => "lima",
    6 => "enam",
    7 => "tujuh",
    8 => "delapan",
    9 => "sembilan");
  $orders = array(
     0 => "",
     1 => "puluh",
     2 => "ratus",
     3 => "ribu",
     6 => "juta",
     9 => "miliar",
    12 => "triliun",
    15 => "kuadriliun");

  $is_neg = $num < 0; $num = "$num";

  //// angka di kiri desimal

  $int = ""; if (preg_match("/^[+-]?(\d+)/", $num, $m)) $int
```



```
   = $m[1];
  $mult = 0; $wint = "";

  // ambil ribuan/jutaan/dst
  while (preg_match('/(\d{1,3})$/', $int, $m)) {

    // ambil satuan, puluhan, dan ratusan
    $s = $m[1] % 10;
    $p = ($m[1] % 100 - $s)/10;
    $r = ($m[1] - $p*10 - $s)/100;

    // konversi ratusan
    if ($r==0) $g = "";
    elseif ($r==1) $g = "se$orders[2]";
    else $g = $digits[$r]." $orders[2]";

    // konversi puluhan dan satuan
    if ($p==0) {
      if ($s==0);
      elseif ($s==1) $g = ($g ? "$g ".$digits[$s] :
                                ($mult==0 ? $digits[1] :
"se"));
      else $g = ($g ? "$g ":"") . $digits[$s];
    } elseif ($p==1) {
      if ($s==0) $g = ($g ? "$g ":"") . "se$orders[1]";
      elseif ($s==1) $g = ($g ? "$g ":"") . "sebelas";
      else $g = ($g ? "$g ":"") . $digits[$s] . " belas";
    } else {
      $g = ($g ? "$g ":"").$digits[$p]." puluh".
           ($s > 0 ? " ".$digits[$s] : "");
    }

    // gabungkan dengan hasil sebelumnya
    $wint = ($g ? $g.($g=="se" ? "":" ").$orders[$mult]:"").
            ($wint ? " $wint":"");

    // pangkas ribuan/jutaan/dsb yang sudah dikonversi
    $int = preg_replace('/\d{1,3}$/', '', $int);
    $mult+=3;
  }
  if (!$wint) $wint = $digits[0];
```

```
   |
 74|  //// angka di kanan desimal
   |
 76|  $frac = ""; if (preg_match("/\.(\d+)/", $num, $m)) $frac =
   $m[1];
 77|  $wfrac = "";
 78|  for ($i=0; $i<strlen($frac); $i++) {
 79|    $wfrac .= ($wfrac ? " ":"").$digits[substr($frac,$i,1)];
 80|  }
   |
 82|  return ($is_neg ? "minus ":"").$wint.($wfrac ? "koma
   $wfrac":"");
 83|}
   |
 85|?>
```

Contoh penggunaan:

```
<?

//
// pakai-terbilang.php
//

require_once "terbilang.php";

foreach  (array(0,1,9,10,11,13,20.030,21,55,99,101,
               110,-1100,10000031,9090909090) as $num) {
  echo "$num = ", terbilang($num), "<br>\n";
}

?>
```

Hasilnya:

```
0 = nol
1 = satu
9 = sembilan
10 = sepuluh
11 = sebelas
13 = tiga belas
20.03 = dua puluh koma nol tiga
21 = dua puluh satu
55 = lima puluh lima
99 = sembilan puluh sembilan
101 = seratus satu
110 = seratus sepuluh
-1100 = minus seribu seratus
10000031 = sepuluh juta tiga puluh satu
9090909090 = sembilan miliar sembilan puluh juta sembilan ratus
sembilan ribu sembilan puluh
```


Algoritma Terbilang cukup sederhana: untuk angka di belakang koma, kita cukup mengkonversi tiap digit menjadi kata yang sesuai (1 menjadi "satu", 2 menjadi "dua", dan seterusnya). Untuk angka di kiri koma, perlu dibagi-bagi digit tiga-tiga menjadi ribuan, lalu jutaan, lalu miliaran, dan seterusnya. Ini karena penyebutan tiap kelompok tiga digit tersebut sama. Misalnya 123 = "seratus dua puluh tiga", 123.000 = "seratus dua puluh tiga ribu"). Di dalam mengonversi kelompok-tiga-tiga menjadi teks, kita perlu memperhatikan beberapa kasus khusus. Misalnya 10 = "sepuluh", bukan "satu puluh", 11 = "sebelas", bukan "sepuluh satu", dan bilangan lain yang sejenis. Langkah terakhir tinggal menggabungkan teks untuk angka di kiri koma dan di kanan koma, dan menambahkan kata "minus" di depan teks jika angkanya negatif.

Resep 3-4: Konversi Basis Bilangan

Konversi antara basis desimal, oktal (basis 8), heksadesimal (16), dan biner (2) juga merupakan sesuatu yang sering dilakukan programer, terutama saat berurusan dengan hal-hal yang berbau sistem/teknis. Untungnya, di PHP mudah sekali mengkonversi antarbasis berapa pun dengan fungsi `base_convert()`.

```
// mengubah desimal ke heksadesimal
echo base_convert(1234, 10, 16);              // "4d2"
echo base_convert("01234", 10, 16);           // "4d2"

// mengubah heksadesimal ke desimal
echo base_convert("04d2", 16, 10);            // "1234"

// mengubah desimal ke oktal
```

```
echo base_convert(1000, 10, 8);              // "1750"

// mengubah oktal ke desimal
echo base_convert(1750, 10, 8);              // "1000"
echo base_convert("1750", 10, 8);            // "1000"

// mengubah desimal ke biner
echo base_convert(999, 10, 2);               // "1111100111"

// mengubah biner ke desimal
echo base_convert("1111110111", 2, 10);      // "999"

// basis lain
echo base_convert(123456, 10, 36);           // "2n9c"
```

Selain `base_convert()`, terdapat fungsi-fungsi lain yang lebih spesifik, mis `hexdec()` untuk mengubah heksadesimal ke desimal, `dechex()` untuk mengubah desimal ke heksadesimal, `bindec()`, `decbin()`, `octdec()`, dan `decoct()`. Menggunakan fungsi-fungsi spesifik ini kadang membuat program lebih jelas dibaca. Untuk mengkonversi bilangan desimal ke string heksadesimal, biner, atau oktal, bisa juga digunakan fungsi `sprintf()` dengan simbol `%x`, `%b`, dan `%o`. Kelebihan menggunakan `sprintf()`, Anda bisa memeriksa jumlah digit yang ingin dikeluarkan:

```
// tampilkan sebagai 16-digit biner
echo sprintf("%016b", 170);          // "0000000010101010"
```

Untuk heksadesimal, juga bisa memilih digit "a" hingga "f" dalam huruf kecil atau huruf kapital. Meskipun untuk melakukan yang sama dengan `dechex()` atau `base_convert()` tinggal ditambahkan aja `strtoupper()` atau `strtolower()`).

Resep 3-5: Mengurutkan Array

Array adalah struktur data yang biasanya digunakan untuk menampung koleksi dan hampir selalu pengurutan dilakukan terhadap array ini. Karena itu PHP sudah menyediakan berbagai fungsi untuk mengurutkan array. Resep-resep di bawah bisa menjadi referensi singkat cara melakukan berbagai macam pengurutan.

```
// contoh data
$buah = array("mangga", "apel", "jambu", "kedondong");
$orang = array("Budi", "de Beers", "Wati", "Anita");
```



```
$file1 = array(".qmail", ".forward", "#tmp#", "SATU.TXT",
"dua.bat");
$file2 = array("img1.jpg", "img11.jpg", "img2.jpg");
$angka = array(10,20,4,20,55,9);

// - mengurutkan secara ASCII ("Z" sebelum "a", "10" sebelum "2")
// - in-place (tidak menghasilkan array baru, memodifikasi
langsung array semula)
sort($buah, SORT_STRING);
  # hasil: apel, jambu, kedondong, mangga
  # biasanya diinginkan
sort($orang, SORT_STRING);
  # hasil: "Anita", "Budi", "Wati", "de Beers"
  # biasanya tidak diinginkan
sort($angka, SORT_STRING);
  # hasil: 10,20,20,4,55,9
  # biasanya tidak diinginkan

// - mengurutkan secara ASCII terbalik
// - in-place
rsort($orang, SORT_STRING);
  # hasil: "de Beers", "Wati", "Budi", "Anita"
rsort($angka, SORT_STRING);
  # hasil: 9,55,4,20,20,10

// - mengurutkan secara ASCII tanpa membedakan huruf besar kecil
// - in-place
function sortfunc_case($a, $b) { return strcasecmp($a,$b); }
usort($orang, 'sortfunc_case');
  # hasil: "Anita", "Budi", "de Beers", "Wati"
  # biasanya diinginkan

// - mengurutkan secara ASCII tanpa membedakan huruf besar kecil,
terbalik
// - in-place
function sortfunc_rcase($a, $b) { return strcasecmp($b,$a); }
usort($orang, 'sortfunc_rcase');
  # hasil: "Wati", "de Beers", "Budi", "Anita"

// - mengurutkan secara numerik ("10" sesudah "2")
// - in-place
```

```
sort($angka, SORT_NUMERIC);
  # hasil: 4,9,10,20,20,55
  # biasanya diinginkan, tapi hanya berguna untuk array berisi
angka saja

// - mengurutkan secara numerik terbalik
// - in-place
rsort($angka, SORT_NUMERIC);
  # hasil: 55,20,20,10,9,4

// - mengurutkan secara "natural" (antara angka dan huruf yang
bergabung tetap
//   diperlakukan terpisah)
// - in-place
natsort($file2); // membedakan huruf besar/kecil, besar selalu
sebelum kecil
natcasesort($file2); // tidak membedakan huruf besar kecil
  # hasil: img1.jpg, img2.jpg, img11.jpg
  # dengan sort() biasa, hasilnya: img1.jpg, img11.jpg, img2.jpg
  #                                (biasanya tidak diinginkan)

// - mengurutkan berdasarkan panjang string (pendek sebelum
panjang)
function sortfunc_strlen($a,$b) {
  $la=strlen($a); $lb=strlen($b);
  if ($la<$lb) return -1; elseif ($la>$lb) return +1; else return
0;
}
usort($buah, "sortfunc_strlen");
  # hasil: apel, jambu, mangga, kedondong
  # biasanya dipakai untuk tujuan khusus saja

// - mengurutkan berdasarkan panjang string, terbalik (panjang
sebelum pendek)
function sortfunc_rstrlen($a,$b) {
  $la=strlen($a); $lb=strlen($b);
  if ($la<$lb) return +1; elseif ($la>$lb) return -1; else return
0;
}
usort($buah, "sortfunc_rstrlen");
  # hasil: kedondong, mangga, jambu, apel
```



```

// - urutkan tanpa mengacuhkan simbol ("#A...", "$A..." diurutkan
sbg "A...")
//   dan tanpa mempedulikan huruf besar kecil
function sortfunc_casewordsonly($a,$b) {
  $sa=preg_replace("/[^A-Za-z0-9]+/","",$a);
  $sb=preg_replace("/[^A-Za-z0-9]+/","",$b);
  return strcasecmp($sa,$sb);
}
usort($file1, "sortfunc_casewordsonly");
  # hasil: dua.bat, .forward, .qmail, SATU.TXT, #tmp#
  # biasanya berguna dalam membuat entri indeks atau mengurutkan
judul lagu
  # atau buku (yang kadang mengandung kutip, dsb) atau nama file
  # dengan sort biasa, hasilnya: #tmp#, .forward, .qmail,
SATU.TXT, dua.txt
  # dengan sort biasa tanpa membedakan huruf besar-kecil,
                        hasilnya: #tmp#, .forward, .qmail,
dua.txt, SATU.TXT

// - urutkan berdasarkan ukuran file (kecil sebelum besar)
function sortfunc_filesize($a,$b) {
  $sa=filesize($a); $b=filesize($b);
  if ($sa<$sb) return -1; elseif ($sa>$sb) return +1; else return
0;
}
usort($file1, "sortfunc_filesize");

// - urutkan berdasarkan tanggal modifikasi file (lama sebelum
baru)
function sortfunc_filemtime($a,$b) {
  $ma=filemtime($a); $mb=filemtime($b);
  if ($ma<$mb) return -1; elseif ($ma>$mb) return +1; else return
0;
}
usort($file1, "sortfunc_filemtime");

// - urutkan berdasarkan nama file, tapi folder selalu sebelum
file
// - tidak membedakan huruf besar/kecil
// - ini pengurutan 2 kriteria, nama file kriteria sekunder
```

```
function sortfunc_filetype_filename($a,$b) {
  $fa=is_dir($a); $fb=is_dir($b);
  if ($fa && !$fb) return -1;
  if ($fb && !$fa) return 1;
  return strcasecmp($a,$b);
}
usort($files, "sortfunc_filetype_filename");

// - bonus :-)
// - urutkan berdasarkan rand(), efeknya adalah mengacak/shuffle
array
function sortfunc_rand($a,$b) { return rand(-1,1); }
usort($buah, "sortfunc_rand");
  # hasil berbeda2 dan acak...
  # tapi untuk mengacak array, shuffle() lebih baik
```