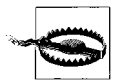
POGLAVLJE 1

Znakovni nizovi

1.0 Uvod

U PHP-u, znakovni nizovi (engl. *strings*) jesu nizovi bajtova – na primer, „Oče naš“ ili „Življaše jednom u nekoj zemlji“ ili čak „111211211“. Podaci koje čitate iz datoteke ili šaljete čitaču Weba predstavljeni su u obliku znakovnih nizova.

PHP-ovi znakovni nizovi su binarno bezbedni (tj. mogu da sadrže null bajtove) i šire se i skupljaju po potrebi. Njihova veličina je ograničena samo količinom memorije koja je PHP-u stavljena na raspolaganje.



PHP-ovi znakovni nizovi najčešće su ASCII znakovni nizovi. Ne-ASCII znakovne nizove, recimo one kodirane po UTF-8 ili drugim višebajtnim šemama za kodiranje, morate drugačije da tretirate; videti poglavlje 19.

PHP-ovi znakovni nizovi su po obliku i ponašanju slični onima u Perlu i Unixovim komandnim okruženjima. Možete ih inicijalizovati na tri načina: tako što ćete ih staviti u polunavodnike, navodnike, ili upotrebiti *heredoc* sintaksu. U znakovnim nizovima zatvorenim u polunavodnike, jedini posebni znakovi koje morate da pretvorite u izlazne sekvence (engl. *escape sequences*) jesu obrnuta kosa crta i sam polunavodnik. U primeru 1-1 prikazana su četiri znakovna niza navedena u polunavodnicima.

Primer 1-1. Znakovni nizovi u polunavodnicima

```
print 'I have gone to the store.';
print 'I\'ve gone to the store.';
print 'Would you pay $1.75 for 8 ounces of tap water?';
print 'In double-quoted strings, newline is represented by \n';
```

Kada se izvrši kôd iz primera 1-1, odštampaće se sledeće:

```
I have gone to the store.
I've gone to the store.
Would you pay $1.75 for 8 ounces of tap water?
In double-quoted strings, newline is represented by \n
```

Pošto PHP ne proverava postoje li u znakovnim nizovima navedenim u polunavodnicima interpolirane promenljive i izlazne sekvence (osim nekih), znakovni nizovi se na taj način definišu jednostavno i brzo.

U znakovnim nizovima datim između navodnika ne prepoznaju se polunavodnici pretvoreni u izlazne sekvence, ali se prepoznaju interpolirane promenljive i izlazne sekvence navedene u tabeli 1-1.

Tabela 1-1. Izlazne sekvence u znakovnim nizovima između navodnika

Izlazna sekvenca	Znak
\n	Novi red (Newline) (ASCII 10)
\r	Početak reda (Carriage return) (ASCII 13)
\t	Tabulator (ASCII 9)
\\	Obrnuta kosa crta
\$	Dolar
\"	Navodnici
Od \0 do \777	Oktalna vrednost
Od \x0 do \xFF	Heksadecimalna vrednost

U primeru 1-2 prikazano je nekoliko znakovnih nizova definisanih stavljanjem između navodnika.

Primer 1-2. Znakovni nizovi između navodnika

```
print "I've gone to the store.";
print "The sauce cost \$10.25.";
$cost = '$10.25';
print "The sauce cost $cost.";
print "The sauce cost \$\061\060.\x32\x35.";
```

Kada se izvrši kôd iz primera 1-2, odštampaće se sledeće:

```
I've gone to the store.
The sauce cost $10.25.
The sauce cost $10.25.
The sauce cost $10.25.
```

Poslednji red primera 1-2 ispravno štampa cenu umaka (engl. *sauce cost*) zato što je u ASCII tabeli kôd znaka 1 decimalno 49 odnosno oktalno 061. Kôd znaka 0 u ASCII tabeli je decimalno 48 odnosno oktalno 060; kôd znaka 2 u ASCII tabeli je decimalno 50 odnosno heksadecimalno 32; kôd znaka 5 u ASCII tabeli je decimalno 53 odnosno heksadecimalno 35.

Dokumenti definisani pomoću *heredoc* sintakse prepoznaju sve interpolacije i izlazne sekvence znakovnih nizova zatvorenih u navodnike (tabela 1-1), ali nije potrebno da se sami navodnici pretvore u izlazne sekvence. Dokumenti definisani pomoću *heredoc* sintakse počinju simbolom <<< iza kojeg sledi završna oznaka. Ta oznaka (bez ikakvih belina ispred ili iza nje) i znak tačka i zarez neposredno iza nje (ako treba, kako bi se pokazalo da je u pitanju kraj naredbe) označavaju kraj tako definisanog dokumenta. U primeru 1-3 pokazano je kako se dokument definiše pomoću *heredoc* sintakse.

Primer 1-3. Definisanje dokumenta pomoću heredoc sintakse

```
print <<< END
It's funny when signs say things like:
   Original "Root" Beer
   "Free" Gift
   Shoes cleaned while "you" wait
or have other misquoted words.
END;
```

Kada se izvrši kôd iz primera 1-3, odštampaće se sledeće:

```
It's funny when signs say things like:
   Original "Root" Beer
   "Free" Gift
   Shoes cleaned while "you" wait
or have other misquoted words.
```

U dokumentu definisanom pomoću *heredoc* sintakse, ostaju očuvani svi prelasci u novi red, razmaci i navodnici. Po konvenciji, identifikator završetka znakovnog niza (završnu oznaku) obično pišemo velikim slovima, pri čemu se razlika između velikih i malih slova uzima u obzir. Primer 1-4 prikazuje još dva validna dokumenta definisana pomoću *heredoc* sintakse.

Primer 1-4. Još dokumenata definisanih pomoću heredoc sintakse

```
print <<< PARSLEY
It's easy to grow fresh:
Parsley
Chives
on your windowsill
PARSLEY;

print <<< DOGS
If you like pets, yell out:
DOGS AND CATS ARE GREAT!
DOGS;
```

Dokumenti definisani pomoću *heredoc* sintakse posebno su podesni za ispisivanje HTML-a sa interpoliranim promenljivama, pošto navodnike koji se pojavljuju unutar HTML elemenata ne morate da pretvarate u izlazne sekvence. U primeru 1-5, HTML se ispisuje upravo uz korišćenje dokumenta definisanog pomoću *heredoc* sintakse.

Primer 1-5. Ispisivanje HTML-a uz korišćenje dokumenta definisanog pomoću heredoc sintakse

```
if ($remaining_cards > 0) {
    $url = '/deal.php';
    $text = 'Deal More Cards';
} else {
    $url = '/new-game.php';
    $text = 'Start a New Game';
}
print <<< HTML
There are <b>$remaining_cards</b> left.
<p>
<a href="$url">$text</a>
HTML;
```

U primeru 1-5, znak tačka i zarez (;) iza završne oznake (HTML) kazuje PHP-u da je došao do kraja naredbe. Međutim, u nekim slučajevima znak tačka i zarez ne treba pisati. Jedan od tih slučajeva prikazan je u primeru 1-6, gde se u dokumentu definisanom pomoću *heredoc* sintakse upotrebljava operator nadovezivanja znakovnih nizova (engl. *concatenation operator*).

Primer 1-6. Nadovezivanje znakovnih nizova u dokumentu definisanom pomoću heredoc sintakse

```
$html = <<< END
<div class="$divClass">
<ul class="$ulClass">
<li>
END
. $listItem . '</li></div>';

print $html;
```

Ukoliko promenljive `$divClass`, `$ulClass` i `$listItem` imaju razumne vrednosti, primer 1-6 će ispisati:

```
<div class="class1">
<ul class="class2">
<li> The List Item </li></div>
```

U primeru 1-6, izraz treba da se nastavi u sledećem redu, pa posle završne oznake END nismo pisali tačku i zarez. Vodite računa i o tome da smo operator nadovezivanja (tačku) morali da pišemo u novom redu iza završne oznake (END), kako bi PHP mogao da je prepozna.

Pojedinačne bajtove znakovnih nizova možete referencirati pomoću uglastih zagrada. Prvi bajt znakovnog niza u PHP-u uvek ima indeks 0. U primeru 1-7, izvadićemo iz znakovnog niza samo jedan bajt.

Primer 1-7. Vađenje pojedinačnog bajta iz znakovnog niza

```
$neighbor = 'Hilda';
print $neighbor[3];
```

Primer 1-7 će ispisati četvrti znak znakovnog niza `$neighbor`:

```
d
```

Za pristupanje pojedinačnom znaku (bajtu) znakovnog niza, možete upotrebiti i vitičaste zagrade. Dakle, `$neighbor{3}` isto je što i `$neighbor[3]`. Pisanje vitičastih zagrada novina je u PHP-u. Njime se postiže vizuelna razlika između indeksiranja znakovnih nizova i indeksiranja nizova (matrica).

1.1 Pristupanje podnizovima

Problem

Želite da utvrdite da li znakovni niz sadrži određeni podniz. Primera radi, želite da utvrdite da li adresa e-pošte sadrži znak `@`.

Rešenje

Upotrebite funkciju `strpos()`, kao u primeru 1-8.

Primer 1-8. Pronalaženje podniza funkcijom strpos()

```
<?php

if (strpos($_POST['email'], '@') === false) {
    print 'There was no @ in the e-mail address!';
 }

?>
```

Objašnjenje

Povratna vrednost funkcije `strpos()` jeste indeks prve pojave podniza („igle“) u nizu („plastu sena“). Ukoliko `strpos()` uopšte ne nađe iglu u plastu sena, vratiće `false`. Ako je igla na početku plasta sena, `strpos()` će vratiti 0, pošto indeks 0 predstavlja početak (prvi znak) znakovnog niza. Da biste razlikovali povratne vrednosti 0 i `false`, morate upotrebiti operator identiteta `(===)` ili neidentiteta `(!==)` umesto običnog operatora jednakosti `(==)` ili nejednakosti `(!=)`. U primeru 1-8, operatorom `===` utvrdićemo da li funkcija `strpos()` vraća vrednost `false`. Ta provera će uspeti samo ako strpos vrati `false`, a ne ako vrati 0 ili bilo koji drugi broj.

Videti i

Dokumentaciju funkcije `strpos()` na adresi *http://www.php.net/strpos*.

1.2 Izdvajanje podnizova

Problem

Iz znakovnog niza hoćete da izdvojite jedan njegov deo, počev od određenog mesta u nizu. Na primer, izdvojili biste prvih osam znakova korisničkog imena upisanog u obrazac.

Rešenje

Za izdvajanje podniza upotrebite funkciju `substr()`, kao u primeru 1-9.

Primer 1-9. Izdvajanje podniza funkcijom substr()

```
<?php
$substring = substr($string,$start,$length);
$username = substr($_GET['username'],0,8);
?>
```

Objašnjenje

Ako su vrednosti parametara `$start` i `$length` pozitivne, `substr()` će iz znakovnog niza izvaditi `$length` znakova, počev od znaka čiji je indeks `$start`. Prvi znak znakovnog niza ima indeks 0. U primeru 1-10, parametri `$start` i `$length` su pozitivni.

Primer 1-10. Upotreba funkcije substr() s pozitivnim parametrima $start i $length

```
print substr('watch out for that tree',6,5);
```

Primer 1-10 će ispisati:

```
out f
```

Ako izostavite parametar `$length`, `substr()` će kao rezultat vratiti podniz od indeksa `$start` do kraja prvobitnog znakovnog niza, kao u primeru 1-11.

Primer 1-11. Upotreba funkcije substr() s pozitivnim parametrom $start i bez parametra $length

```
print substr('watch out for that tree',17);
```

Primer 1-11 će ispisati:

```
t tree
```

Ukoliko je vrednost parametra `$start` veća od dužine znakovnog niza, `substr()` vraća `false`.

Ako zbir vrednosti parametara `$start` i `$length` premašuje dužinu znakovnog niza, funkcija `substr()` vraća ceo podniz od indeksa `$start` do kraja niza, kao u primeru 1-12.

Primer 1-12. Upotreba funkcije substr() kada dužina podniza premašuje kraj znakovnog niza

```
print substr('watch out for that tree',20,5);
```

Primer 1-12 će ispisati:

```
ree
```

Ukoliko je parametar `$start` negativan, `substr()` broji unazad od kraja znakovnog niza i tako utvrđuje početak podniza koji treba izdvojiti, kao u primeru 1-13.

Primer 1-13. Upotreba funkcije substr() uz negativnu vrednost parametra $start

```
print substr('watch out for that tree',-6);
print substr('watch out for that tree',-17,5);
```

Rezultat izvršavanja primera 1-13 biće

```
t tree
out f
```

Ukoliko negativna vrednost parametra `$start` premašuje početak znakovnog niza (recimo, kada `$start` iznosi -27, a dužina znakovnog niza je 20), `substr()` se ponaša kao da je `$start` jednak 0.

U slučaju da je vrednost parametra `$length` negativna, `substr()` broji unazad od kraja znakovnog niza i tako utvrđuje znak kojim se završava podniz koji treba izvaditi, kao u primeru 1-14.

Primer 1-14. Upotreba funkcije substr() uz negativnu vrednost parametra $length

```
print substr('watch out for that tree',15,-2);
print substr('watch out for that tree',-4,-1);
```

Primer 1-14 će ispisati:

```
hat tr
tre
```

Videti i

Dokumentaciju funkcije `substr()` na adresi *http://www.php.net/substr*.

1.3 Zamena podnizova

Problem

Jedan podniz želite da zamenite drugim. Primera radi, pre štampanja hoćete da sakrijete sve cifre broja kreditne kartice sem poslednje četiri.

Rešenje

Upotrebite funkciju `substr_replace( )`, kao u primeru 1-15.

Primer 1-15. Zamena podniza pomoću funkcije substr_replace()

```
// Sve od indeksa $start do kraja znakovnog niza $old_string
// postaje $new_substring
$new_string = substr_replace($old_string,$new_substring,$start);

// $length znakova, počev od indeksa $start, postaje znakovni niz $new_substring
$new_string = substr_replace($old_string,$new_substring,$start,$length);
```

Objašnjenje

Kada ne dobije argument `$length`, funkcija `substr_replace()` zamenjuje sve, počev od indeksa `$start` do kraja znakovnog niza. Kada je parametar `$length` specificiran, zamenjuje se samo zadati broj znakova:

```
print substr_replace('My pet is a blue dog.','fish.',12);
print substr_replace('My pet is a blue dog.','green',12,4);
$credit_card = '4111 1111 1111 1111';
print substr_replace($credit_card,'xxxx ',0,strlen($credit_card)-4);

My pet is a fish.
My pet is a green dog.
xxxx 1111
```

Ako je vrednost parametra `$start` negativna, novi podniz biće smešten `$start` znakova od kraja znakovnog niza `$old_string`, a ne od njegovog početka:

```
print substr_replace('My pet is a blue dog.','fish.',-9);
print substr_replace('My pet is a blue dog.','green',-9,4);

My pet is a fish.
My pet is a green dog.
```

Ako su `$start` i `$length` jednaki 0, novi podniz biće umetnut na početak znakovnog niza `$old_string`:

```
print substr_replace('My pet is a blue dog.','Title: ',0,0);

Title: My pet is a blue dog.
```

Funkcija `substr_replace()` podesna je kada je tekst predug da bi se sav prikazao odjednom, pa želite da prikažete samo deo teksta i vezu ka ostatku. Primer 1-16 prikazuje prvih 25 znakova poruke i tri tačke, kao vezu ka stranici koja prikazuje ostatak teksta.

Primer 1-16. Prikazivanje dugačkog teksta pomoću znaka tri tačke

```
$r = mysql_query("SELECT id,message FROM messages WHERE id = $id") or die();
$ob = mysql_fetch_object($r);
printf('<a href="more-text.php?id=%d">%s</a>',
       $ob->id, substr_replace($ob->message,' ...',25));
```

U primeru 1-16 referencirana je stranica *more-text.php*, koja preuzima i prikazuje celu poruku čiji je ID (identifikator) prosleđen u znakovnom nizu upita (engl. *query*).

Videti i

Dokumentaciju funkcije `substr_replace()` na adresi *http://www.php.net/substr-replace*.

1.4 Pojedinačna obrada svakog znaka (bajta) znakovnog niza

Problem

Svaki bajt (znak) znakovnog niza morate da obradite pojedinačno.

Rešenje

Petljom `for` pojedinačno izdvojte svaki znak (bajt) znakovnog niza. U primeru 1-17 prebrojaćemo samoglasnike znakovnog niza funkcijom `strstr()`.

Primer 1-17. Obrada svakog bajta znakovnog niza pojedinačno

```
<?php
$string = "This weekend, I'm going shopping for a pet chicken.";
$vowels = 0;
for ($i = 0, $j = strlen($string); $i < $j; $i++) {
    if (strstr('aeiouAEIOU',$string[$i])) {
       $vowels++;
    }
}
?>
```

Objašnjenje

Pojedinačnom obradom svakog znaka znakovnog niza lako ćemo izračunati sekvencu tipa „Pregledaj i saopšti“ (engl. *look and say*), kao u primeru 1-18.

Primer 1-18. Sekvenca „Pregledaj i saopšti“

```
<?php
function lookandsay($s) {
    // inicijalizovaćemo povratnu vrednost praznim znakovnim nizom
    $r = '';
    // $m sadrži znak koji prebrojavamo; kao početnu vrednost dodelićemo mu
    // prvi znak znakovnog niza
    $m = $s[0];
    // $n je broj dosad prebrojanih znakova kao $m, inicijalizovaćemo ga na 1
    $n = 1;
    for ($i = 1, $j = strlen($s); $i < $j; $i++) {
        // ako je ovaj znak jednak prethodnom
        if ($s[$i] == $m) {
            // povećaj broj ponavljanja ovog znaka za 1
            $n++;
        } else {
            // u protivnom, dodaj povratnoj vrednosti broj ponavljanja i znak
            $r .= $n.$m;
            // zadaj da znak koji tražimo bude tekući znak
            $m = $s[$i];
            // i resetuj broj ponavljanja na 1
            $n = 1;
        }
    }
    // vrati napravljen znakovni niz, poslednji znak i broj ponavljanja
    return $r.$n.$m;
}

for ($i = 0, $s = 1; $i < 10; $i++) {
    $s = lookandsay($s);
    print "$s <br/>\n";
}
```

Primer 1-18 će ispisati:

```
1
11
21
1211
111221
312211
13112221
1113213211
31131211131221
13211311123113112211
```

Takva sekvenca se naziva „Pregledaj i saopšti“ pošto svaki njen element dobijate pregledom prethodnog elementa i imenovanjem njegovog sadržaja. Na primer, prvi element je 1, što čitamo kao „jedna jedinica“. Zato je drugi element „11“. To su dve jedinice, pa je treći element „21“. Slično kao pre, to su jedna dvojka i jedna jedinica, pa je četvrti element jednak „1211“ itd.

Videti i

Dokumentaciju petlje `for` na adresi *http://www.php.net/for*; više informacija o sekvencama „Pregledaj i saopšti“ na adresi *http://mathworld.wolfram.com/LookandSaySequence.html*.

1.5 Obrtanje znakovnog niza reč po reč ili znak po znak (bajt po bajt)

Problem

Želite da obrnete redosled reči ili znakova (bajtova) znakovnog niza.

Rešenje

Za obrtanje redosleda bajtova znakovnog niza upotrebite funkciju `strrev()`, kao u primeru 1-19.

Primer 1-19. Obrtanje redosleda bajtova znakovnog niza

```
<?php
print strrev('This is not a palindrome.');
?>
```

Primer 1-19 će ispisati:

```
.emordnilap a ton si sihT
```

Da biste obrnuli redosled reči znakovnog niza, razdelite znakovni niz funkcijom `explode()` po granicama reči, obrnite redosled tih reči funkcijom `array_reverse()` i zatim ih spojite u nov znakovni niz, kao u primeru 1-20.

Primer 1-20. Obrtanje redosleda reči znakovnog niza

```
<?php
$s = "Once upon a time there was a turtle.";
// podeli znakovni niz na reči
$words = explode(' ',$s);
// obrni niz reči
$words = array_reverse($words);
// ponovo napravi znakovni niz
$s = implode(' ',$words);
print $s;
?>
```

Primer 1-20 će ispisati:

```
turtle. a was there time a upon Once
```

Objašnjenje

Za obrtanje redosleda reči znakovnog niza funkcijom `implode()` dovoljan je jedan red koda, kao u primeru 1-21.

Primer 1-21. Sažeto obrtanje redosleda reči znakovnog niza

```
<?php
$reversed_s = implode(' ',array_reverse(explode(' ',$s)));
?>
```

Videti i

Recept 23.7, gde se razmatraju implikacije korišćenja neke druge granice reči umesto razmaka; dokumentaciju funkcije `strrev()` na adresi *http://www.php.net/strrev*, a funkcije `array_reverse()` na adresi *http://www.php.net/array-reverse*.

1.6 Povećavanje i smanjivanje širine tabulatora (kolona)

Problem

U znakovnom nizu, želite da zamenite razmake tabulatorima (ili tabulatore razmacima), a da tekst ostane poravnat u kolone. Na primer, formatiran tekst treba da prikažete korisnicima na standardizovan način.

Rešenje

Za zamenu razmaka tabulatorima ili tabulatora razmacima upotrebite funkciju `str_replace()`, kao u primeru 1-22.

Primer 1-22. Uzajamna zamena tabulatora i razmaka

```
<?php
$r = mysql_query("SELECT message FROM messages WHERE id = 1") or die();
$ob = mysql_fetch_object($r);
$tabbed = str_replace(' ',"\t",$ob->message);
$spaced = str_replace("\t",' ',$ob->message);

print "With Tabs: <pre>$tabbed</pre>";
print "With Spaces: <pre>$spaced</pre>";
?>
```

Međutim, pri konverziji razmaka u tabulatore pomoću funkcije `str_replace()` ne uzima se u obzir širina kolona. Ukoliko želite da kolone počinju na svakih osam znakova, a da red počinje rečju od pet slova i tabulatorom, taj tabulator treba zameniti trima razmacima, a ne jednim. Da biste tabulatore pretvorili u razmake uz očuvanje širine kolona, upotrebite funkciju `pc_tab_expand()`, kao u primeru 1-23.

Primer 1-23. Funkcija pc_tab_expand()

```
<?php
function pc_tab_expand($text) {
    while (strstr($text,"\t")) {
      $text = preg_replace_callback('/^([^\t\n]*)(\t+)/m','pc_tab_expand_helper',
              $text);
    }
    return $text;
}

function pc_tab_expand_helper($matches) {
    $tab_stop = 8;

    return $matches[1] .
    str_repeat(' ',strlen($matches[2]) *
                         $tab_stop - (strlen($matches[1]) % $tab_stop));
}


$spaced = pc_tab_expand($ob->message);
?>
```

Za pretvaranje razmaka u tabulatore (uz očuvanje širine kolona) upotrebite funkciju `pc_tab_unexpand()`, kao u primeru 1-24.

Primer 1-24. Funkcija pc_tab_unexpand()

```
<?php
function pc_tab_unexpand($text) {
    $tab_stop = 8;
    $lines = explode("\n",$text);
    foreach ($lines as $i => $line) {
        // Zamena svih tabulatora razmacima
        $line = pc_tab_expand($line);
        $chunks = str_split($line, $tab_stop);
        $chunkCount = count($chunks);
```

```
        // Pregledaj sve sem poslednjeg parčeta
        for ($j = 0; $j < $chunkCount - 1; $j++) {
            $chunks[$j] = preg_replace('/ {2,}$/',"\t",$chunks[$j]);
        }
        // Ako poslednje parče ima razmake za jednu kolonu,
        // pretvori ga u tabulator; u protivnom, ostavi ga onakvo kakvo je.
        if ($chunks[$chunkCount-1] == str_repeat(' ', $tab_stop)) {
            $chunks[$chunkCount-1] = "\t";
        }
        // Ponovo spoji parčiće u red:
        $lines[$i] = implode('',$chunks);
    }
    // Ponovo spoji redove u znakovni niz:
    return implode("\n",$lines);
}

$tabbed = pc_tab_unexpand($ob->message);
?>
```

Obe funkcije primaju znakovni niz kao argument i vraćaju znakovni niz modifikovan na odgovarajući način.

Objašnjenje

U svim funkcijama se pretpostavlja da je svaka kolona (znak tabulatora) široka kao osam razmaka, ali to se može promeniti tako što se zada druga vrednost za promenljivu `$tab_stop`.

Regularan izraz u funkciji `pc_tab_expand()` odgovara i grupi tabulatora i celom tekstu tog reda pre te grupe tabulatora. Tekst pre tabulatora mora da se uzme u obzir zato što od njegove dužine zavisi broj razmaka kojima treba zameniti te tabulatore, da bi dalji tekst bio poravnat sa sledećom kolonom (znakom tabulatora). Funkcija se ne ograničava samo na to da svaki tabulator zameni sa osam razmaka, već tekst nakon znaka tabulatora poravnava u kolone.

Slično tome, ni funkcija `pc_tab_unexpand()` ne ograničava se samo na to da traži osam uzastopnih razmaka i da ih zameni jednim znakom tabulatora, već svaki red deli na parčad od po osam znakova i zatim završnu belinu svakog parčeta (od najmanje dva razmaka) zamenjuje tabulatorom. Time ne samo da se zadržava poravnanje teksta u kolone, nego se i štedi prostor u znakovnom nizu.

Videti i

Dokumentaciju funkcije `str_replace()` na adresi *http://www.php.net/str-replace*, funkcije `preg_replace_callback()` na adresi *http://www.php.net/preg_replace_callback*, i funkcije `str_split()` na adresi *http://www.php.net/str_split*. O funkciji `preg_replace_callback()` govorićemo i u receptu 22.10.

1.7 Pretvaranje velikih slova u mala i obrnuto

Problem

Želite da pretvorite mala slova u velika, velika u mala ili da na neki drugi način modifikujete veličinu slova znakovnog niza. Na primer, početna slova imena hoćete da pretvorite u velika, a ostala slova u mala.

Rešenje

Za pretvaranje prvog slova jedne, odnosno više reči u veliko slovo, upotrebite funkciju `ucfirst()` odnosno funkciju `ucwords()`, kao u primeru 1-25.

Primer 1-25. Pretvaranje malih slova u velika

```
<?php
print ucfirst("how do you do today?");
print ucwords("the prince of wales");
?>
```

Primer 1-25 će ispisati:

```
How do you do today?
The Prince Of Wales
```

Za pretvaranje velikih slova u mala (ili obrnuto) u celom znakovnom nizu, upotrebite funkcije `strtolower()` odnosno `strtoupper()`, kao u primeru 1-26.

Primer 1-26. Pretvaranje velikih slova u mala (i obrnuto) u celom znakovnom nizu

```
print strtoupper("i'm not yelling!");
// Sadržaj oznaka mora biti ispisan malim slovima da bi bio kompatibilan sa XHTML-om
print strtolower('<A HREF="one.php">one</A>');
```

Primer 1-26 će ispisati:

```
I'M NOT YELLING!
<a href="one.php">one</a>
```

Objašnjenje

Za pretvaranje prvog slova reči u veliko slovo upotrebite funkciju `ucfirst()`:

```
<?php
print ucfirst('monkey face');
print ucfirst('1 monkey face');
?>
```

Time će se ispisati:

```
Monkey face
1 monkey fac
```

Obratite pažnju na to da druga rečenica ne glasi „1 Monkey face“.

Funkciju `ucwords()` upotrebite za pretvaranje prvog slova svake reči znakovnog niza u veliko slovo:

```
<?php
print ucwords('1 monkey face');
print ucwords("don't play zone defense against the philadelphia 76-ers");
?>
```

Time će se ispisati:

```
1 Monkey Face
Don't Play Zone Defense Against The Philadelphia 76-ers
```

Kao što smo očekivali, funkcija `ucwords()` ne pretvara u veliko slovo „t“ u reči „don't“. Ali ona ne pretvara u veliko slovo ni „e“ u reči „76-ers“. Za funkciju `ucwords()`, reč je svaka sekvenca znakova koji ostavljaju trag (nisu beline), a slede iza jednog ili više znakova koji ne ostavljaju trag, tj. iza belina. Pošto i znak `'` i znak `-` ostavljaju trag, funkcija `ucwords()` ne smatra ni „t“ u reči „don't“ niti „e“ u reči „76-ers“ znakovima kojima su započete nove reči.

Mala slova koja nisu prva u reči ne menjaju u velika ni funkcija `ucfirst()` niti funkcija `ucwords()`:

```
<?php
print ucfirst('macWorld says I should get an iBook');
print ucwords('eTunaFish.com might buy itunaFish.Com!');
?>
```

Time će se ispisati:

```
MacWorld says I should get an iBook
ETunaFish.com Might Buy ItunaFish.Com!
```

Funkcije `strtolower()` i `strtoupper()` deluju na ceo znakovni niz, ne samo na pojedinačne znakove. `strtolower()` menja sva slova abecede u mala, a `strtoupper()` sva slova abecede u velika:

```
<?php
print strtolower("I programmed the WOPR and the TRS-80.");
print strtoupper('"since feeling is first" is a poem by e. e. cummings.');
?>
```

Ispisaće se sledeće:

```
i programmed the wopr and the trs-80.
"SINCE FEELING IS FIRST" IS A POEM BY E. E. CUMMINGS.
```

Prilikom pretvaranja malih slova u velika, te funkcije uzimaju u obzir parametre koji opisuju regionalne (lokalne) specifičnosti zadate na računaru.

Videti i

Više informacija o parametrima koji opisuju regionalne specifičnosti potražite u poglavlju 19; dokumentaciju o funkciji `ucfirst()` na adresi *http://www.php.net/ucfirst*, o funkciji `ucwords()` na adresi *http://www.php.net/ucwords*, o funkciji `strtolower()` na adresi *http://www.php.net/strtolower*, i o funkciji `strtoupper()` na adresi *http://www.php.net/strtoupper*.

1.8 Interpoliranje rezultata funkcija i izraza unutar znakovnih nizova

Problem

Rezultate izvršavanja funkcije ili izračunavanja izraza želite da ubacite u znakovni niz.

Rešenje

Ako određenu vrednost ne možete da ubacite u znakovni niz, upotrebite operator (.) za nadovezivanje znakovnih nizova, kao u primeru 1-27.

Primer 1-27. Nadovezivanje znakovnih nizova

```
<?php
print 'You have '.($_REQUEST['boys'] + $_REQUEST['girls']).' children.';
print "The word '$word' is ".strlen($word).' characters long.';
print 'You owe '.$amounts['payment'].' immediately';
print "My circle's diameter is ".$circle->getDiameter().' inches.';
?>
```

Objašnjenje

Promenljive, svojstva objekata i elemente nizova (čiji indeksi nisu unutar navodnika) možete da navedete neposredno u znakovnom nizu zatvorenom u navodnike:

```
<?php
print "I have $children children.";
print "You owe $amounts[payment] immediately.";
print "My circle's diameter is $circle->diameter inches.";
?>
```

Interpolacija u znakovnom nizu zatvorenom u navodnike nameće neka ograničenja sintakse onoga što se može interpolirati. U prethodnom primeru, `$amounts['payment']` morali smo da napišemo kao `$amounts[payment]` da bi se pravilno interpoliralo. Za interpolaciju komplikovanijih izraza u znakovni niz upotrebite vitičaste zagrade. Na primer:

```
<?php
print "I have less than {$children} children.";
print "You owe {$amounts['payment']} immediately.";
print "My circle's diameter is {$circle->getDiameter()} inches.";
?>
```

Direktna interpolacija i nadovezivanje znakovnih nizova mogući su i u dokumentima definisanim pomoću *heredoc* sintakse. Interpolacija uz nadovezivanje znakovnih nizova u dokumentima definisanim pomoću *heredoc* sintakse ume čudno da izgleda, zato što završna oznaka dokumenta definisanog pomoću *heredoc* sintakse i operator nadovezivanja znakovnih nizova moraju biti u zasebnim redovima:

```
<?php
print <<< END
Right now, the time is
END
. strftime('%c') . <<< END
 but tomorrow it will be
END
. strftime('%c',time() + 86400);
?>
```

Sem toga, ako interpolirate u dokumentu definisanom pomoću *heredoc* sintakse, povedite računa o razmacima potrebnim da bi ceo znakovni niz izgledao kako treba. U prethodnom primeru, `Right now the time` mora se završiti jednim razmakom, a deo `but tomorrow it will be` mora imati po jedan razmak na početku i na kraju.

Videti i

Za sintaksu interpoliranja vrednosti promenljivih (kao što je `${"amount_$i"}`), pogledajte recept 5.4; dokumentaciju operatora nadovezivanja znakovnih nizova potražite na adresi *http://www.php.net/language.operators.string*.

1.9 Uklanjanje belina s početka i kraja znakovnih nizova

Problem

Želite da uklonite beline s početka ili kraja znakovnog niza, recimo zato da biste očistili korisnički unos pre nego što ga podvrgnete validaciji.

Rešenje

Upotrebite funkcije `ltrim()`, `rtrim()` ili `trim()`. Funkcija `ltrim()` uklanja belinu s početka znakovnog niza, `rtrim()` s kraja znakovnog niza, a `trim()` i s početka i s kraja znakovnog niza:

```
<?php
$zipcode = trim($_REQUEST['zipcode']);
$no_linefeed = rtrim($_REQUEST['text']);
$name = ltrim($_REQUEST['name']);
?>
```

Objašnjenje

Za te funkcije, beline su sledeći znakovi: novi red (engl. *newline*), početak reda (engl. *carriage return*), razmak, horizontalni i vertikalni tabulator, i znak null.

Uklanjanjem belina ispred i iza znakovnih nizova štedi se memorijski prostor, a omogućava se i preciznije prikazivanje formatiranih podataka i teksta unutar, recimo, oznaka `<pre>`. Ukoliko korisnički unos upoređujete s nečim, najpre uklonite beline ispred i iza njega, kako onaj ko greškom unese „98052“ kao svoj poštanski broj ne bi morao da sve piše iz početka. Uklanjanjem belina pre poređenja s tačnim tekstom obezbeđuje se i da se, na primer, „salami\n“ podudari sa „salami“. Takođe, znakovni niz treba da normalizujete tako što ćete ukloniti beline pre nego što ga smestite u bazu podataka.

Funkcije `trim()` mogu da uklanjaju iz znakovnih nizova i znakove koje zada korisnik. Znakove koje treba ukloniti zadajte kao drugi argument. Ukoliko treba ukloniti ceo opseg znakova [od ..do], zadajte ga pomoću dve tačke između prvog i poslednjeg znaka opsega:

```
<?php
// Uklanjanje cifara i razmaka s početka reda
print ltrim('10 PRINT A$',' 0..9');
// Uklanjanje znaka tačka i zarez s kraja reda
print rtrim('SELECT * FROM turtles;',';');
?>
```

Ispisaće se:

```
PRINT A$
SELECT * FROM turtles
```

PHP ima i funkciju `chop()`, što je samo drugo ime za `rtrim()`. Međutim, bolje je da upotrebljavate `rtrim()` umesto PHP-ove funkcije `chop()`, zato što se ona ponaša drukčije od Perlove funkcije `chop()` (kojoj bi u Perlu ionako trebalo pretpostaviti funkciju `chomp`), a mogla bi da zbuni ostale kada budu čitali vaš kôd.

Videti i

Dokumentaciju funkcije `trim()` na adresi *http://www.php.net/trim*, funkcije `ltrim()` na adresi *http://www.php.net/ltrim* i funkcije `rtrim()` na adresi *http://www.php.net/rtrim*.

1.10 Generisanje podataka razdvojenih zarezima

Problem

Želite da formatirate podatke kao vrednosti razdvojene zarezima (engl. *comma-separated values*, CSV), da biste ih mogli uvesti u tabelarni proračun ili bazu podataka.

Rešenje

Za generisanje reda vrednosti razdvojenih zarezima od niza (engl. *array*) podataka upotrebite funkciju `fputcsv()`. U primeru 1-28, podaci iz niza `$sales` upisuju se u datoteku.

Primer 1-28. Generisanje podataka razdvojenih zarezima

```
<?php

$sales = array( array('Northeast','2005-01-01','2005-02-01',12.54),
                array('Northwest','2005-01-01','2005-02-01',546.33),
                array('Southeast','2005-01-01','2005-02-01',93.26),
                array('Southwest','2005-01-01','2005-02-01',945.21),
                array('All Regions','--','--',1597.34) );

$fh = fopen('sales.csv','w') or die("Can't open sales.csv");
foreach ($sales as $sales_line) {
    if (fputcsv($fh, $sales_line) === false) {
        die("Can't write CSV line");
    }
}
fclose($fh) or die("Can't close sales.csv");

?>
```

Objašnjenje

Za ispisivanje podataka razdvojenih zarezima na izlaz umesto upisivanja u datoteku, upotrebite poseban izlazni tok `php://output`, kao u primeru 1-29.

Primer 1-29. Ispisivanje podataka razdvojenih zarezima

```
<?php

$sales = array( array('Northeast','2005-01-01','2005-02-01',12.54),
                array('Northwest','2005-01-01','2005-02-01',546.33),
                array('Southeast','2005-01-01','2005-02-01',93.26),
                array('Southwest','2005-01-01','2005-02-01',945.21),
                array('All Regions','--','--',1597.34) );

$fh = fopen('php://output','w');
foreach ($sales as $sales_line) {
    if (fputcsv($fh, $sales_line) === false) {
        die("Can't write CSV line");
    }
}
fclose($fh);
?>
```

Ukoliko podatke razdvojene zarezima hoćete da upišete u znakovni niz, a ne da ih ispišete na izlaz odnosno upišete u datoteku, kombinujte tehniku iz primera 1-29 i baferisanje izlaza funkcijom `ob_get_contents()`, kao u primeru 1-30.

Primer 1-30. Upisivanje podataka razdvojenih zarezima u znakovni niz

```
<?php

$sales = array( array('Northeast','2005-01-01','2005-02-01',12.54),
                array('Northwest','2005-01-01','2005-02-01',546.33),
                array('Southeast','2005-01-01','2005-02-01',93.26),
                array('Southwest','2005-01-01','2005-02-01',945.21),
                array('All Regions','--','--',1597.34) );

ob_start();
$fh = fopen('php://output','w') or die("Can't open php://output");
foreach ($sales as $sales_line) {
    if (fputcsv($fh, $sales_line) === false) {
        die("Can't write CSV line");
    }
}
fclose($fh) or die("Can't close php://output");
$output = ob_get_contents();
ob_end_clean();
?>
```

Videti i

Dokumentaciju funkcije `fputcsv()` na adresi *http://www.php.net/fputcsv* te više informacija o baferisanju izlaza.

1.11 Raščlanjivanje podataka razdvojenih zarezima

Problem

Imate podatke u formatu vrednosti razdvojenih zarezima (CSV) — recimo, datoteku uvezenu iz Excela ili neke baze podataka; podatke želite da raščlanite (engl. *parse*) na zapise i polja, i to u formatu s kojim možete raditi u PHP-u.

Rešenje

Ukoliko su CSV podaci u datoteci (ili dostupni na nekoj URL adresi), otvorite datoteku funkcijom `fopen()` i učitajte podatke funkcijom `fgetcsv()`. U primeru 1-31, ispisuju se CSV podaci neke HTML tabele.

Primer 1-31. Čitanje CSV podataka iz datoteke

```
<?php
$fp = fopen('sample2.csv','r') or die("can't open file");
print "<table>\n";
while($csv_line = fgetcsv($fp)) {
    print '<tr>';
    for ($i = 0, $j = count($csv_line); $i < $j; $i++) {
        print '<td>'.htmlentities($csv_line[$i]).'</td>';
    }
    print "</tr>\n";
```

```
}
print '</table>\n';
fclose($fp) or die("can't close file");
?>
```

Objašnjenje

U PHP-u 4, kao drugi argument funkcije `fgetcsv()` morate da zadate vrednost veću od maksimalne dužine reda CSV datoteke. (Ne zaboravite da uračunate beline na kraju redova.) U PHP-u 5, dužina reda je opcioni argument. Ako ga ne zadate, `fgetcsv()` će učitati ceo red podataka. (U PHP-u 5.0.4 i kasnijim verzijama, isto ćete postići ako prosledite vrednost 0 kao dužinu reda.) U slučaju da je prosečna dužina reda veća od 8192 bajta, program će se izvršavati brže ukoliko dužinu reda zadate eksplicitno, umesto da pustite PHP da je pogađa.

Funkciji `fgetcsv()` možete proslediti i opcioni treći argument, graničnik koji umesto zareza razdvaja podatke. Međutim, upotrebom drugačijeg graničnika donekle se gubi smisao korišćenja CSV datoteka kao jednostavnog sredstva za razmenu tabelarnih podataka.

Ne padajte u iskušenje da zaobiđete `fgetcsv()` tako što ćete učitati podatke i na sve zareze primeniti funkciju `explode()`. CSV je previše komplikovan za takav pristup, zato što njegova polja mogu sadržati, na primer, prave, doslovne zareze koje ne treba tretirati kao graničnike polja. Funkcija `fgetcsv()` štiti vas i vaš kôd od takvih „sitnih“ grešaka.

Videti i

Dokumentaciju funkcije `fgetcsv()` na adresi *http://www.php.net/fgetcsv*.

1.12 Generisanje zapisa s poljima fiksne širine

Problem

Želite da formatirate zapise podataka tako da svako polje zauzme dati broj znakova.

Rešenje

Upotrebite funkciju `pack()` sa znakovnim nizom za formatiranje (engl. *format string*) koji specificira sekvencu znakovnih nizova nadopunjenih razmacima. U primeru 1-32, pretvorićemo niz podataka u zapise fiksne širine.

Primer 1-32.Generisanje zapisa s poljima fiksne širine

```
<?php

$books = array( array('Elmer Gantry', 'Sinclair Lewis', 1927),
                array('The Scarlatti Inheritance','Robert Ludlum',1971),
                array('The Parsifal Mosaic','William Styron',1979) );
```

```
foreach ($books as $book) {
    print pack('A25A15A4', $book[0], $book[1], $book[2]) . "\n";
}

?>
```

Objašnjenje

Znakovni niz za formatiranje, `A25A14A4`, kazuje funkciji `pack()` da argumente koji slede pretvori u: znakovni niz od 25 znakova nadopunjen razmacima, znakovni niz od 14 znakova nadopunjen razmacima i znakovni niz od 4 znaka nadopunjen razmacima. Funkcija `pack()` daje koncizno rešenje za polja nadopunjena razmacima u zapisima fiksne širine.

Ukoliko polja želite da nadopunite nekim drugim znakom, a ne razmacima, upotrebite funkciju `substr()` koja obezbeđuje da vrednosti u poljima ne budu predugačke, odnosno funkciju `str_pad()` koja obezbeđuje da vrednosti u poljima ne budu prekratke. U primeru 1-33, pretvorićemo niz zapisa u zapise fiksne širine, čija će polja do pune širine biti nadopunjena znakom tačka.

Primer 1-33. Generisanje zapisa s poljima fiksne širine, bez funkcije pack()

```
<?php

$books = array( array('Elmer Gantry', 'Sinclair Lewis', 1927),
                array('The Scarlatti Inheritance','Robert Ludlum',1971),
                array('The Parsifal Mosaic','William Styron',1979) );

foreach ($books as $book) {
    $title = str_pad(substr($book[0], 0, 25), 25, '.');
    $author = str_pad(substr($book[1], 0, 15), 15, '.');
    $year = str_pad(substr($book[2], 0, 4), 4, '.');
    print "$title$author$year\n";
}

?>
```

Videti i

Dokumentaciju funkcije `pack()` na adresi *http://www.php.net/pack* i funkcije `str_pad()` na adresi *http://www.php.net/str_pad*. Znakovni nizovi funkcije `pack()` za formatiranje, detaljnije su razmotreni u receptu 1.16.

1.13 Raščlanjivanje zapisa s poljima fiksne širine

Problem

Zapise fiksne širine hoćete da raščlanite u znakovne nizove.

Rešenje

Upotrebite funkciju `substr()`, kao u primeru 1-34.

Primer 1-34. Raščlanjivanje zapisa fiksne širine funkcijom substr()

```
<?php
$fp = fopen('fixed-width-records.txt','r') or die ("can't open file");
while ($s = fgets($fp,1024)) {
    $fields[1] = substr($s,0,10);  // prvo polje: prvih 10 znakova u redu
    $fields[2] = substr($s,10,5);  // drugo polje: sledećih 5 znakova u redu
    $fields[3] = substr($s,15,12); // treće polje: sledećih 12 znakova u redu
    // neka funkcija koja obrađuje polja
    process_fields($fields);
}
fclose($fp) or die("can't close file");
?>
```

Ili upotrebite funkciju `unpack()`, kao u primeru 1-35.

Primer 1-35. Raščlanjivanje zapisa fiksne širine funkcijom unpack()

```
<?php
$fp = fopen('fixed-width-records.txt','r') or die ("can't open file");
while ($s = fgets($fp,1024)) {
    // asocijativni niz čiji su ključevi "title", "author" i "publication_year"
    $fields = unpack('A25title/A14author/A4publication_year',$s);
    // neka funkcija koja obrađuje polja
    process_fields($fields);
}
fclose($fp) or die("can't close file");
?>
```

Objašnjenje

Evo kako izgleda spisak naslova knjiga, imena njihovih autora i datuma objavljivanja, ispisan tako da je svakom polju dodeljen fiksan broj znakova; svaki zapis zauzima jedan red:

```
<?php
$booklist=<<<END
Elmer Gantry             Sinclair Lewis1927
The Scarlatti InheritanceRobert Ludlum 1971
The Parsifal Mosaic      Robert Ludlum 1982
Sophie's Choice          William Styron1979
END;
?>
```

U svakom redu, na naslov odlazi prvih 25 znakova, na ime autora sledećih 14 znakova, a na datum objavljivanja sledeća 4 znaka. Znajući te širine polja, lako ćete funkcijom `substr()` izdvojiti svako polje i upisati ga u niz:

```
<?php
$books = explode("\n",$booklist);
```

```
for($i = 0, $j = count($books); $i < $j; $i++) {
  $book_array[$i]['title'] = substr($books[$i],0,25);
  $book_array[$i]['author'] = substr($books[$i],25,14);
  $book_array[$i]['publication_year'] = substr($books[$i],39,4);
}
?>
```

Nakon podele dokumenta `$booklist` u niz redova, istu petlju možemo upotrebiti i za jedan znakovni niz i za više redova učitanih iz datoteke.

Petlja će biti prilagodljivija ukoliko imena i širine polja zadamo u zasebnom nizu koji se funkciji za raščlanjivanje može proslediti, kao u funkciji `pc_fixed_width_substr()` u primeru 1-36.

Primer 1-36. Funkcija pc_fixed_width_substr()

```
<?php
function pc_fixed_width_substr($fields,$data) {
  $r = array();
  for ($i = 0, $j = count($data); $i < $j; $i++) {
    $line_pos = 0;
    foreach($fields as $field_name => $field_length) {
      $r[$i][$field_name] = rtrim(substr($data[$i],$line_pos,$field_length));
      $line_pos += $field_length;
    }
  }
  return $r;
}

$book_fields = array('title' => 25,
                     'author' => 14,
                     'publication_year' => 4);

$book_array = pc_fixed_width_substr($book_fields,$books);
?>
```

Promenljiva `$line_pos` prati početak svakog polja. Dok petlja prolazi kroza svaki red, ta promenljiva se povećava za širinu prethodnog polja. Za uklanjanje beline iza svakog polja upotrebite funkciju `rtrim()`.

Kao zamenu za `substr()`, za izdvajanje polja možete upotrebiti funkciju `unpack()`. Umesto da specificirate imena i širine polja u obliku asocijativnog niza, za `unpack()` treba da napravite znakovni niz za formatiranje. U primeru 1-37 imate funkciju `pc_fixed_width_unpack()`, koja pomoću funkcije `unpack()` izdvaja polja fiksne širine.

Primer 1-37. Funkcija pc_fixed_width_unpack()

```
<?php

function pc_fixed_width_unpack($format_string,$data) {
  $r = array();
  for ($i = 0, $j = count($data); $i < $j; $i++) {
    $r[$i] = unpack($format_string,$data[$i]);
  }
```

```
  return $r;
}

$book_array = pc_fixed_width_unpack('A25title/A14author/A4publication_year',
                                    $books);

?>
```

Pošto funkcija `unpack()` shvata format A kao „znakovni niz nadopunjen razmacima“, nema potrebe da uklanjate završne razmake funkcijom `rtrim()`.

Nakon što bilo koja od prethodnih funkcija izdvoji polja u niz `$book_array`, podaci se mogu ispisati, na primer, u obliku HTML tabele:

```
<?php
$book_array = pc_fixed_width_unpack('A25title/A14author/A4publication_year',
                                    $books);
print "<table>\n";
// ispiši red zaglavlja
print '<tr><td>';
print join('</td><td>',array_keys($book_array[0]));
print "</td></tr>\n";
// ispiši sve redove podataka
foreach ($book_array as $row) {
    print '<tr><td>';
    print join('</td><td>',array_values($row));
    print "</td></tr>\n";
}
print '</table>\n';
?>
```

Zadavanjem argumenata `</td><td>` funkciji `join()` koja elemente niza (engl. *array*) nadovezuje u jedan znakovni niz (engl. *string*) povezujući ih prvim argumentom (u ovom slučaju znakovima `</td><td>`), dobijamo red HTML tabele bez prve oznake `<td>` i poslednje oznake `</td>`. Upotpunićemo red tabele tako što ćemo ispisati oznake `<tr><td>` pre tako povezanih podataka, odnosno `</td></tr>` iza tako povezanih podataka.

Funkcije `substr()` i `unpack()` imaju jednake mogućnosti kada su u poljima znakovni nizovi, ali je `unpack()` bolje rešenje kada elementi polja nisu samo znakovni nizovi.

U slučaju da su sva polja iste širine, funkcija `str_split()` služi za lako cepanje ulaznih podataka. Taj novitet PHP-a 5 vraća izlazni niz sastavljen od delova ulaznog znakovnog niza. U primeru 1-38, funkcijom `str_split()` delimo znakovni niz na delove širine 32 bajta.

Primer 1-38. Podela znakovnog niza na delove jednake širine funkcijom str_split()

```
<?php
$fields = str_split($line_of_data,32);
// $fields[0] sadrži bajtove 0–31
// $fields[1] sadrži bajtove 32–63
// i tako dalje
```

Videti i

Više informacija o funkciji `unpack()` potražite u receptu 1.16 i na adresi *http://www.php.net/unpack*; dokumentaciju funkcije `str_split()` na adresi *http://www.php.net/str_split*; funkcija `join()` razmotrena je u receptu 4.8.

1.14 Podela znakovnog niza na delove

Problem

Treba da podelite znakovni niz na delove. Na primer, hoćete da pristupite svakom redu teksta koji je korisnik upisao u polje `<textarea>` na obrascu.

Rešenje

Ako su delovi razgraničeni istim nizom znakova kao graničnikom, upotrebite funkciju `explode()`:

```
<?php
$words = explode(' ','My sentence is not very complicated');
?>
```

U slučaju da graničnik morate opisati regularnim izrazom kompatibilnim s POSIX-om ili Perlom, upotrebite funkciju `split()` ili `preg_split()`:

```
<?php
$words = split(' +','This sentence has some extra whitespace in it.');
$words = preg_split('/\d\. /','my day: 1. get up 2. get dressed 3. eat toast');
$lines = preg_split('/[\n\r]+/',$_REQUEST['textarea']);
?>
```

Funkciju `spliti()`, odnosno funkciju `preg_split()` sa indikatorom `/i` upotrebite i ukoliko želite da se, prilikom ispitivanja podudaranja znakova s graničnikom, razlika između velikih i malih slova ne uzima u obzir:

```
<?php
$words = spliti(' x ','31 inches x 22 inches X 9 inches');
$words = preg_split('/ x /i','31 inches x 22 inches X 9 inches');
?>
```

Objašnjenje

Najjednostavnije rešenje od svih daje funkcija `explode()`. Prosledite joj niz znakova koji definiše graničnik, znakovni niz koji teba izdeliti i, opciono, broj elemenata koje treba da vrati kao svoj rezultat:

```
<?php
$dwarves = 'dopey,sleepy,happy,grumpy,sneezy,bashful,doc';
$dwarf_array = explode(',',$dwarves);
?>
```

Time bismo dobili `$dwarf_array` kao niz od sedam elemenata, pa bi naredba `print_r($dwarf_array)` ispisala:

```
Array
(
    [0] => dopey
    [1] => sleepy
    [2] => happy
    [3] => grumpy
    [4] => sneezy
    [5] => bashful
    [6] => doc
)
```

Ako se specificira broj elemenata rezultata manji od broja mogućih delova, poslednji deo će obuhvatiti ostatak znakovnog niza:

```
<?php
$dwarf_array = explode(',',$dwarves,5);
print_r($dwarf_array);
?>
```

Rezultat bi bio:

```
Array
(
    [0] => dopey
    [1] => sleepy
    [2] => happy
    [3] => grumpy
    [4] => sneezy,bashful,doc
)
```

Funkcija `explode()` doslovno shvata zadati graničnik. Ako kao graničnik zadate zarez i razmak, ona će znakovni niz podeliti samo tamo gde nađe i zarez i razmak, a ne tamo gde nađe samo zarez ili samo razmak.

Funkcija `split()` je prilagodljivija. Umesto da graničnik zadajete doslovno, njoj zadajete POSIX regularan izraz:

```
<?php
$more_dwarves = 'cheeky,fatso, wonder boy, chunky,growly, groggy, winky';
$more_dwarf_array = split(', ?',$more_dwarves);
?>
```

Ovaj regularan izraz cepa ulazni znakovni niz tamo gde nađe zarez i opciono razmak iza njega, čime se svi novi elementi (niz `$more_dwarves`) tretiraju ispravno. Neće biti pocepan element koji u imenu sadrži razmak, ali će ulazni znakovni niz biti pocepan na svim mestima gde stoji "," ili ", ". Naredba `print_r($more_dwarf_array()` ispisaće:

```
Array
(
    [0] => cheeky
    [1] => fatso
```

```
    [2] => wonder boy
    [3] => chunky
    [4] => growly
    [5] => groggy
    [6] => winky
)
```

Slična funkciji `split()` jeste funkcija `preg_split()`, ali se njoj zadaje regularan izraz kompatibilan s Perlom, a ne POSIX-ov. Uz `preg_split()` možete da koristite razna proširenja Perlovih regularnih izraza, i trikove kao što je uključivanje teksta graničnika u rezultujući niz znakovnih nizova:

```
<?php
$math = "3 + 2 / 7 - 9";
$stack = preg_split('/ *([+\-\/*]) */',$math,-1,PREG_SPLIT_DELIM_CAPTURE);
print_r($stack);
?>
```

Ispisaće se:

```
Array
(
    [0] => 3
    [1] => +
    [2] => 2
    [3] => /
    [4] => 7
    [5] => -
    [6] => 9
)
```

Regularan izraz graničnika podudara se sa četiri matematička operatora (+, -, /, *), ispred i iza kojih su opcioni razmaci. Indikator `PREG_SPLIT_DELIM_CAPTURE` kazuje funkciji `preg_split()` da u rezultujući niz znakovnih nizova uključi i znakove koji se podudaraju s regularnim izrazom graničnika navedenim u malim zagradama. Pošto su u njima samo znakovi matematičkih operatora, u vraćenom nizu neće biti razmaka.

Videti i

Regularni izrazi su detaljnije razmotreni u poglavlju 22; dokumentaciju funkcije `explode()` potražite na adresi *http://www.php.net/explode*, funkcije `split()` na adresi *http://www.php.net/split*, i funkcije `preg_split()` na adresi *http://www.php.net/preg-split*.

1.15 Prelamanje teksta na redove određene dužine

Problem

Hoćete da prelomite znakovni niz u više redova. Primera radi, tekst želite da prikažete između oznaka `<pre>/</pre>`, ali tako da ostane unutar prozora čitača Weba uobičajene veličine.

Rešenje

Upotrebite funkciju `wordwrap()`:

```
<?php
$s = "Four score and seven years ago our fathers brought forthø
on this continent a new nation, conceived in liberty andø
dedicated to the proposition that all men are created equal.";

print "<pre>\n".wordwrap($s)."\n</pre>";
?>
```

Time će se ispisati:

```
<pre>
Four score and seven years ago our fathers brought forth on this continent
a new nation, conceived in liberty and dedicated to the proposition that
all men are created equal.
</pre>
```

Objašnjenje

Funkcija `wordwrap()` podrazumevano prelama tekst u redove dužine 75 znakova. Opcionim drugim argumentom drugu dužinu reda:

```
<?php
print wordwrap($s,50);
?>
```

Ispisaće se:

```
Four score and seven years ago our fathers brought
forth on this continent a new nation, conceived in
liberty and dedicated to the proposition that all
men are created equal.
```

Za prelazak u novi red mogu se upotrebiti i drugi znakovi, ne samo `\n`. Kada želite dvostruki prored, zadajte `"\n\n"`:

```
<?php
print wordwrap($s,50,"\n\n");
?>
```

Ispisaće se:

```
Four score and seven years ago our fathers brought

forth on this continent a new nation, conceived in

liberty and dedicated to the proposition that all

men are created equal.
```

Opcioni četvrti argument funkcije `wordwrap()` određuje tretman reči dužih od specificirane dužine reda. Ukoliko je taj argument 1, te reči će biti prelomljene. U protivnom, biće ispisane neprekinute, iako se time premašuje zadata dužina reda:

```
<?php
print wordwrap('jabberwocky',5);
print wordwrap('jabberwocky',5,"\n",1);
?>
```

Ispisaće se:

```
jabberwocky

jabbe
rwock
y
```

Videti i

Dokumentaciju funkcije `wordwrap()` na adresi *http://www.php.net/wordwrap*.

1.16 Smeštanje binarnih podataka u znakovne nizove

Problem

Želite da raščlanite znakovni niz koji sadrži vrednosti kodirane u obliku binarne strukture ili da upišete kodirane vrednosti u znakovni niz. Na primer, hoćete da uskladištite brojeve predstavljene binarno, a ne kao sekvence ASCII znakova.

Rešenje

Za upisivanje binarnih podataka u znakovni niz, upotrebite funkciju `pack()`:

```
<?php
$packed = pack('S4',1974,106,28225,32725);
?>
```

Za izdvajanje binarnih podataka iz znakovnog niza, upotrebite funkciju `unpack()`:

```
<?php
$nums = unpack('S4',$packed);
?>
```

Objašnjenje

Prvi argument funkcije `pack()` je oznaka formata koja opisuje način kodiranja podataka prosleđenih u ostalim argumentima. Oznaka formata S4 kazuje funkciji `pack()` da od ulaznih podataka napravi četiri 16-bitna broja tipa short, bez predznaka, po redosledu bajtova važećem na računaru na kojem se program izvršava. Ako su kao ulazni podaci dati brojevi 1974, 106, 28225 i 32725, a na računaru se najpre piše

manje značajan (engl. *little-endian*) bajt broja, funkcija će kao rezultat dati osam bajtova: 182, 7, 106, 0, 65, 110, 213 i 127. Svaki par bajtova odgovara jednom od ulaznih brojeva: 7 * 256 + 182 daje 1974; 0 * 256 + 106 daje 106; 110 * 256 + 65 = 28225; 127 * 256 + 213 = 32725.

I prvi argument funkcije `unpack()` je oznaka formata, a drugi argument su podaci koje treba dekodirati. Kada joj kao oznaku formata prosledimo `S4`, osmobajtna sekvenca koju je dala funkcija `pack()` daće niz od četiri elementa s prvobitnim brojevima. Naredba `print_r($nums)` ispisala bi:

```
Array
(
    [1] => 1974
    [2] => 106
    [3] => 28225
    [4] => 32725
)
```

U funkciji `unpack()`, iza oznake formata i broja elemenata možete navesti znakovni niz od kojeg treba napraviti ključ niza. Na primer:

```
<?php
$nums = unpack('S4num',$packed);
print_r($nums);
?>
```

Ispisaće se:

```
Array
(
    [num1] => 1974
    [num2] => 106
    [num3] => 28225
    [num4] => 32725
)
```

U funkciji `unpack()`, više oznaka formata mora biti razdvojeno kosom crtom `/`:

```
<?php
$nums = unpack('S1a/S1b/S1c/S1d',$packed);
print_r($nums);
?>
```

Ispisaće se:

```
Array
(
    [a] => 1974
    [b] => 106
    [c] => 28225
    [d] => 32725
)
```

U tabeli 1-2 navedene su oznake formata koje se mogu upotrebljavati u funkcijama `pack()` i `unpack()`.

Tabela 1-2. Oznake formata za funkcije pack() i unpack()

Oznaka formata	Tip podataka
a	Znakovni niz dopunjen vrednostima NULL
A	Znakovni niz dopunjen razmacima
h	Heksadecimalni znakovni niz, prvo niži nibl (polubajt)
H	Heksadecimalni znakovni niz, prvo viši nibl (polubajt)
c	`signed char` (označen znakovni tip)
C	`unsigned char` (neoznačen znakovni tip)
s	`signed short` (16 bitova, redosled bajtova zavisi od mašine)
S	`unsigned short` (16 bitova, redosled bajtova zavisi od mašine)
n	`unsigned short` (16 bitova, redosled bajtova big endian)
v	`unsigned short` (16 bitova, redosled bajtova little endian)
i	`signed int` (veličina i redosled bajtova zavise od mašine)
I	`unsigned int` (veličina i redosled bajtova zavise od mašine)
l	`signed long` (32 bita, redosled bajtova zavisi od mašine)
L	`unsigned long` (32 bita, redosled bajtova zavisi od mašine)
N	`unsigned long` (32 bita, redosled bajtova big endian)
V	`unsigned long` (32 bita, redosled bajtova little endian)
f	`float` (veličina i način predstavljanja zavise od mašine)
d	`double` (veličina i način predstavljanja zavise od mašine)
x	Bajt vrednosti nula
X	Arhiviranje/dearhiviranje jednog bajta
@	Popunjavanje vrednošću NUL do apsolutno zadate pozicije

U formatima `a`, `A`, `h` i `H`, broj iza oznake formata pokazuje dužinu znakovnog niza. Primera radi, `A25` označava razmacima nadopunjen znakovni niz dužine 25 znakova. U ostalim oznakama formata, broj iza oznake formata pokazuje koliko je takvih tipova uzastopno u znakovnom nizu. Za ostatak dostupnih podataka upotrebite oznaku `*`.

Funkcija `unpack()` služi i za konverziju tipova podataka. U narednom primeru, niz `$ascii` popunićemo ASCII kodom svakog znaka u `$s`:

```
<?php
$s = 'platypus';
$ascii = unpack('c*',$s);
print_r($ascii);
?>
```

Ispisaće se:

```
Array
(
    [1] => 112
    [2] => 108
```

```
    [3] => 97
    [4] => 116
    [5] => 121
    [6] => 112
    [7] => 117
    [8] => 115
)
```

Videti i

Dokumentaciju funkcije `pack()` na adresi *http://www.php.net/pack* i funkcije un`pack()` na adresi *http://www.php.net/unpack*.

1.17 Program: CSV datoteka za preuzimanje

Kombinacija funkcije `header()` za menjanje tipa sadržaja izlaza PHP programa i funkcije `fputcsv()` za formatiranje podataka, omogućava slanje CSV datoteka čitačima Weba, koji ih automatski predaju programu za obradu tabelarnih proračuna (odnosno aplikaciji koja je u konkretnom klijentskom sistemu zadužena za rad sa CSV datotekama). U primeru 1-39, formatiraćemo rezultate SQL upita `SELECT` kao CSV datoteku i dati joj odgovarajuća zaglavlja tako da je čitač može ispravno obraditi.

Primer 1-39. CSV datoteka za preuzimanje

```
<?php

require_once 'DB.php';
// Povezivanje s bazom podataka
$db = DB::connect('mysql://david:haxOr@localhost/phpcookbook');

// Vađenje podataka iz baze
$sales_data = $db->getAll('SELECT region, start, end, amount FROM sales');
// Otvaranje datoteke za funkciju fputcsv()
$output = fopen('php://output','w') or die("Can't open php://output");
$total = 0;

// Saopšti čitaču da očekuje CSV datoteku
header('Content-Type: application/csv');
header('Content-Disposition: attachment; filename="sales.csv"');

// Ispiši red zaglavlja
fputcsv($output,array('Region','Start Date','End Date','Amount'));
// Ispiši sve redove podataka i povećaj $total
foreach ($sales_data as $sales_line) {
    fputcsv($output, $sales_line);
    $total += $sales_line[3];
}
// Ispiši red sa $total i zatvori datoteku
fputcsv($output,array('All Regions','--','--',$total));
fclose($output) or die("Can't close php://output");

?>
```

Program u primeru 1-39 šalje dva zaglavlja kako bi se obezbedilo da čitač ispravno obradi CSV izlaz. Prvo zaglavlje, `Content-Type`, kazuje čitaču da tip izlaza nije HTML nego CSV. Drugo zaglavlje, `Content-Disposition`, kazuje čitaču da izlaz ne prikazuje sam, već da pokuša da učita spoljni program koji će to uraditi. Atribut `filename` ovog zaglavlja daje čitaču podrazumevano ime koje može upotrebiti za preuzetu datoteku.

Ukoliko iste podatke želite da prikažete na različite načine, možete da kombinujete kôd za formatiranje na jednoj stranici i da promenljivom upita (engl. *query string variable*) odredite koju vrstu formatiranja treba obaviti. U primeru 1-40, promenljiva upita nazvana `format` određuje da li će se rezultati SQL upita `SELECT` vratiti u obliku HTML tabele ili CSV datoteke.

Primer 1-40. Dinamička odluka: CSV ili HTML

```
<?php

$db = new PDO('sqlite:/usr/local/data/sales.db');

$query = $db->query('SELECT region, start, end, amount FROM sales', PDO::FETCH_NUM);
$sales_data = $db->fetchAll();
$total = 0;
$column_headers = array('Region','Start Date','End Date','Amount');
// Odluči koji format ćeš upotrebiti
$format = $_GET['format'] == 'csv' ? 'csv' : 'html';

// Ispiši početak koji odgovara izabranom formatu
if ($format == 'csv') {
    $output = fopen('php://output','w') or die("Can't open php://output");
    header('Content-Type: application/csv');
    header('Content-Disposition: attachment; filename="sales.csv"');
    fputcsv($output,$column_headers);
 } else {
    echo '<table><tr><th>';
    echo implode('</th><th>', $column_headers);
    echo '</th></tr>';
 }

foreach ($sales_data as $sales_line) {
    // Ispiši red koji odgovara izabranom formatu
    if ($format == 'csv') {
        fputcsv($output, $sales_line);
    } else {
        echo '<tr><td>' . implode('</td><td>', $sales_line) . '</td></tr>';
    }
     $total += $sales_line[3];
}
$total_line = array('All Regions','--','--',$total);

// Ispiši podnožje koje odgovara izabranom formatu
if ($format == 'csv') {
```

```
        fputcsv($output,$total_line);
        fclose($output) or die("Can't close php://output");
    } else {
        echo '<tr><td>' . implode('</td><td>', $total_line) . '</td></tr>';
        echo '</table>';
    }
?>
```

Ako u primeru 1-40 pristupite programu uz `format=csv` u upitu, on će kao svoj izlaz vratiti CSV datoteku. Za sve druge vrednosti promenljive `format` u upitu, vratiće HTML izlaz. Istom logikom koja promenljivoj `$format` zadaje vrednosti CSV ili HTML, mogli biste zadati i druge izlazne formate, recimo XML. Ukoliko iste podatke želite da ponudite na preuzimanje na više mesta i u više formata, upakujte kôd iz primera 1-40 u funkciju koja prihvata niz podataka i specifikator formata, i zatim prikazuje odgovarajuće rezultate.