Po prostu PHP. Techniki zaawansowane

Autor: Larry Ullman
Tłumaczenie: Radosław Meryk
ISBN: 83-7197-775-1
Tytuł oryginału: PHP Advanced For The World Wide Web. Visual QuickPro Guide
Format: B5, stron: 492

Język programowania PHP stanowi dla wielu osób przepustkę w świat pisania aplikacji działających po stronie serwera WWW. Łatwość z jaką przychodzi nauczenie się tego języka, sprawiła, że używają go setki tysięcy amatorów i profesjonalistów na całym świecie.

Po pewnym czasie wiedza wyniesiona z podręczników opisujących podstawy PHP języka przestaje wystarczać. Niniejsza książka pomoże Ci w wykonaniu kolejnego kroku: kroku w kierunku pisania zaawansowanych aplikacji. Dzięki niej wzbogacisz swoją wiedzę i staniesz się prawdziwym ekspertem programowania w PHP, poszukiwanym na rynku pracy.

- Poznasz tajniki programowania obiektowego.
- Nauczysz się korzystać z baz danych.
- Dowiesz się, w jaki sposób zabezpieczać stworzone przez siebie aplikacje.
- Napiszesz własny sklep internetowy, korzystając z sesji i bazy SQL.
- Poznasz sposoby uruchamiania programów PHP w oderwaniu od serwera WWW.
- Nauczysz się generować nie tylko strony WWW, ale także grafikę i pliki PDF.
- Dowiesz się, jak i po co używać języka XML.
- Skorzystasz z wielu rozszerzeń języka, które ułatwiają rozwiązywanie złożonych problemów.

Pomoże Ci w tym prosty język w jakim napisana jest książka oraz liczne przykłady kodu, a także osoba autora, doświadczonego programisty i wykładowcy PHP na Uniwersytecie Kalifornijskim w Berkeley.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Zaawansowane programowanie w PHP

1

Na najprostszym poziomie dobre programowanie wyraża się tym, czy aplikacja lub skrypt działa zgodnie z zamiarem. Początkujący programiści pozostaną na tym poziomie i nie ma w takim podejściu nic złego. Jednakże zaawansowany programista będzie próbował pójść nieco dalej. Będzie dążył do zapewnienia lepszej wydajności, niezawodności, bezpieczeństwa i przenośności. Ta książka nauczy nas, w jaki sposób rozwinąć umiejętności zaawansowanego programisty PHP.

Ten rozdział opisuje niektóre nowe funkcje i właściwości języka PHP w wersji 4., techniki, jakie będą stosowane w tej książce, oraz kilka wskazówek i sztuczek rzemiosła. Chociaż zapewne już wiemy, w jaki sposób korzystać z tablic, to prawdopodobnie jeszcze nie znamy konstrukcji `foreach` lub starszej, ale wciąż bardzo użytecznej funkcji `array_walk()`. Prawdopodobnie mieliśmy już okazję zapisać własną funkcję, ale być może nie do końca wiemy, w jaki sposób wykorzystywać rekurencję oraz zmienne statyczne. W rozdziale tym opiszemy te elementy, a także inne podstawowe informacje, jak: dokumentowanie kodu, tworzenie jego struktury, stałe oraz powiązania. Wyjaśnimy też różnice pomiędzy wykorzystaniem funkcji `print()` a funkcji `echo()`, a także w jaki sposób tworzyć aliasy zmiennych — technikę, która jest nowa dla języka PHP w wersji 4. Ostatecznie w procesie pisania przykładowych skryptów dowiemy się, że można tworzyć dynamiczne aplikacje WWW, stosując bazę danych w prostym pliku tekstowym.

W rozdziale tym utworzymy kilka skryptów, służących do utworzenia i zarządzania ośrodkiem totalizatora sportowego online, gdzie użytkownicy odgadują zwycięskie zespoły. W każdym tygodniu oraz w całym sezonie obliczany jest współczynnik poprawnych typów dla każdego z użytkowników. Wymaga to wykonania niewielu czynności administracyjnych. Wybrałem ten przykład nie ze względu na to, że odpowiada tematowi, ale również dlatego, że sięga on czasów moich początków w PHP. Nauczyłem się tego języka (po przejściu z języka Perl) przy okazji tworzenia podobnej aplikacji. Oczywiście wtedy zakończyłem pracę, kiedy tylko skrypt zaczął działać. Nie muszę dodawać, że skrypty zaprezentowane w tym rozdziale są bardziej wydajne, niezawodne, bezpieczne oraz przenośne niż te skrypty, które opracowałem wtedy.

Struktura i dokumentowanie kodu

Właściwa struktura kodu oraz jego dokumentowanie, mimo że nie ma wpływu na działanie aplikacji, stanowi podstawę zaawansowanego programowania w PHP. Właściwa struktura oraz dokumentowanie jest elementem, który należy zastosować dla wygody własnej oraz naszych klientów i współpracowników, a także dla programisty, który w przyszłości będzie być może zmuszony poprawiać nasz kod.

Kiedy piszę o strukturze kodu, mam na myśli sposób fizycznej organizacji kodu PHP w samym dokumencie (W rozdziale 5., „Projektowanie aplikacji WWW" omówię zagadnienie szersze — strukturę witryny). Porównajmy skrypty 1.1 oraz 1.2. Obydwa są poprawnymi dokumentami PHP, ale który z nich wolelibyśmy przeglądać i poprawiać?

Podstawy budowania struktury kodu odnoszą się do sposobu stosowania wcięć w kodzie, pozostawiania pustych wierszy, używania nawiasów zwykłych i klamrowych itp. Ogólne zasady są następujące:

- należy stosować wcięcia bloków kodu (np. wyników instrukcji warunkowych lub zawartość funkcji) o szerokości jednej tabulacji lub czterech spacji (w zasadzie powinno się stosować spacje, a nie tabulatory, ale programiści stosują wygodniejsze tabulacje);
- określone fragmenty kodu należy oddzielać pustym wierszem, aby wyróżnić je wizualnie;
- należy wprowadzać spacje pomiędzy słowami, argumentami funkcji, operatorami itp. (generalnie, choć nie zawsze, w języku PHP liczba spacji nie ma znaczenia);
- funkcje należy umieszczać na początku dokumentu.

Skrypt 1.1. *Ten skrypt będzie działać prawidłowo, ale znacznie trudniej go poprawiać i zrozumieć niż skrypt 1.2*

Kod

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//PL"
2   "http://www.w3.org/TR/2000/REC-xhtml1-
    20000126/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml">
4   <head>
5   <title>Totalizator piłkarski</title>
6   </head>
7   <body>
8   <?php # script_01_01.php
9   define ("W", $w);
10  function write_data ($v, $k, $fp) {
11  static $fl;
12  if (!$fl) {
13  fputs ($fp, "Wiersz atrapa\n");
14  $fl = TRUE;
15  }
16  fputs ($fp, "$v\n");
17  }
18  $f = '2001/picks_' . W . '_Winners_.txt'; /
19  if ($fp = @fopen ($f, "w")) {
20  array_walk ($w, 'write_data', $fp);
21  fclose ($fp);
22  echo 'Zapisano zwycięzców!<br></br>';
23  } else {
24  echo "Nie można otworzyć pliku $f!
    Należy upewnić się, że zmienna ma wartość.
     <br></br>";
25  }
26  function read_picks ($dp, $d) {
27  global $p;
28  if ($fn = readdir ($dp)) {
29  if (substr($fn, 0, 6) == "picks_") {
30  $pfn = explode ("_", $fn);
31  if (($pfn[1] == W) and ($pfn[2] !=
    "Winners")) { /
32  $un = $pfn[2];
33  $tf = $d . $fn;
34  $up = file ($tf);
35  $p[$un] = $up;
36  }
37  }
38  read_picks ($dp, $d);
39  }
40  }
41  $p = array();
42  $d = '2001/';
```

Skrypt 1.1. *ciąg dalszy*

Kod

```
43  $dp = opendir ($d);
44  read_picks ($dp, $d);
45  closedir ($dp);
46
47  while (list ($k, $v) = each ($p)) {
48  $ws = 0;
49  $ls = 0;
50  while (list ($k2, $v2) = each ($v)) {
51  $v2 = substr ($v2, 0, (strlen($v2) - 1));
52  if (($v2 == $w[$k2]) and ($k2 != 0)) {
53  $ws++;
54  } elseif ($k2 != 0 ) {
55  $ls++;
56  }
57  }
58  $r[$k] = array ('ws' => $ws, 'ls' => $ls);
59  }
60  ksort ($r);
61  function write_results ($v, $k, $fp) {
62  static $fl2;
63  if (!$fl2) {
64  fputs ($fp, "Użytkownik\tws\tls\n");
65  $fl2 = TRUE;
66  }
67  $d = implode ("\t", $v);
68  fputs ($fp, "$k\t$d\n");
69
70  $f2 = '2001/results_' . W . '_.txt';
71  if ($fp2 = @fopen ($f2, "w")) {
72  array_walk ($r, 'write_results', $fp2); .
73  fclose ($fp2); // Zamknięcie pliku.
74  echo 'Zapisano wyniki.<br></br>';
75  } else {
76  echo "Nie można utworzyć pliku
    z tygodniowymi wynikami, $f2.<br></br>";
77  }
78  ?>
79  </body>
80  </html>
```

Po utworzeniu struktury naszego kodu powinniśmy zapewnić, aby był on dobrze udokumentowany. Dokumentowanie kodu jest programowym odpowiednikiem umieszczania notatek na żółtych, samoprzylepnych karteczkach. Można zaryzykować stwierdzenie, że dokumentowania pracy nigdy za dużo. Należy opisać działanie funkcji, zmiennych, fragmentów kodu oraz całych stron. W skrypcie 1.3 pokazałem, jak wyglądałby skrypt 1.2, gdyby zawierał komentarze. Oto kilka rad dotyczących dokumentowania kodu:

- zapisujmy przeznaczenie zmiennych, chyba że ich przeznaczenie nie jest oczywiste nawet dla początkujących programistów;
- opisujmy zastosowanie funkcji;
- wyjaśniajmy, jaki jest cel działania fragmentów kodu;
- wymieniajmy pliki, z którymi program się komunikuje, których wymaga itp.;
- oznaczajmy zamykające nawiasy klamrowe dla złożonych funkcji oraz struktur sterujących (instrukcji warunkowych, pętli itp.).

Skrypt 1.2. *Wstawienie pustych wierszy i tabulatorów znacznie zwiększa czytelność skryptu*

Kod

```
1    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
     1.0 Transitional//PL"
2    "http://www.w3.org/TR/2000/
     REC-xhtml1-20000126/DTD/xhtml1-
     transitional.dtd">
3    <html xmlns="http://www.w3.org/1999/xhtml">
4    <head>
5       <title> Totalizator piłkarski </title>
6    </head>
7    <body>
8    <?php
9
10   define ("WEEK", $week);
11
12   function write_data ($value, $key, $fp) {
13
14      static $first_line;
15
16      if (!$first_line) {
17         fputs ($fp, "Wiersz atrapa \n");
18         $first_line = TRUE;
19      }
20
21      fputs ($fp, "$value\n");
22   }
23
24   $file = '2001/picks_' . WEEK .
     '_Winners_.txt'; /
25
26   if ($fp = @fopen ($file, "w")) {
27
28      array_walk ($winners, 'write_data',
        $fp);
29      fclose ($fp);
30
31      echo 'Zapisano zwycięzców!
        <br></br>';
32
33   } else {
34      echo "Nie można otworzyć pliku $f!
        Należy upewnić się, że zmienna ma
        wartość.
        <br></br>";
35   }
36
37   function read_picks ($dp, $directory) {
38
39      global $picks;
40
41      if ($file_name = readdir ($dp)) {
42
43         if (substr($file_name, 0, 6) ==
           "picks_") {
```

Skrypt 1.2. *ciąg dalszy*

Kod

```
44
45            $parse_file_name = explode
              ("_", $file_name);
46
47            if (($parse_file_name[1] ==
              WEEK) and ($parse_file_name[2] !=
              "Winners")) { /
48
49               $username =
                 $parse_file_name[2];
50               $the_file = $directory .
                 $file_name;
51               $users_picks = file
                 ($the_file);
52               $picks[$username] =
                 $users_picks;
53            }
54         }
55
56         read_picks ($dp, $directory);
57      }
58   }
59
60   $picks = array();
61
62   $directory = '2001/';
63   $dp = opendir ($directory);
64   read_picks ($dp, $directory);
65   closedir ($dp);
66
67
68   while (list ($key, $value) = each
     ($picks)) {
69
70      $wins = 0;
71      $losses = 0;
72
73      while (list ($key2, $value2) = each
        ($value)) {
74
75         $value2 = substr ($value2, 0,
           (strlen($value2) - 1));
76
77         if (($value2 == $winners[$key2])
           and ($key2 != 0)) {
78            $wins++;
79         } elseif ($key2 != 0 ) {
80            $losses++;
81         }
82      }
83
84      $results[$key] = array ('wins' =>
        $wins, 'losses' => $losses);
85   }
```

Skrypt 1.2. *ciąg dalszy*

Kod

```
86
87   ksort ($results);
88
89
90   function write_results ($value, $key,
     $fp) {
91
92      static $first_line2;
93
94      if (!$first_line2) {
95         fputs ($fp, "Użytkownik\tTypy
           poprawne\tTypy błędne\n");
96         $first_line2 = TRUE;
97      }
98
99      $data = implode ("\t", $value);
100     fputs ($fp, "$key\t$data\n");
101
102
103  $file2 = '2001/results_' . WEEK .
     '_.txt';
104
105  if ($fp2 = @fopen ($file2, "w")) {
106
107     array_walk ($results,
        'write_results', $fp2); .
108     fclose ($fp2); // Zamknięcie pliku.
109     echo 'Zapisano wyniki.<br></br>';
110
111  } else {
112
113     echo "Nie można otworzyć pliku
        z tygodniowymi wynikami,
        $file2.<br></br>";
114  }
115  ?>
116  </body>
117  </html>
```

Skrypt 1.3. *Ta wersja skryptu 1.2 została właściwie udokumentowana, ale oczywiście jest miejsce na dodanie dodatkowych notatek*

Kod

```
1    <?php # script_01_03.php
2    // ******************************
3    // Sieciowy system totalizatora
     piłkarskiego
4
5    // Opracował: Larry E. Ullman
6
7    // Zapisano  : Sierpień 2001
8    // Poprawiono: 14 sierpnia 2001
9
10   // Kontact: php@DMCinsights.com
11
12   // Strona process_winners.php obsługuje
     stronę pick_winners.php page.
     Określa ona skuteczność typów
     poszczególnych graczy i zapisuje te
     informacje w pliku tekstowym.
13
14   // ******************************
15   ?>
16   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
     1.0 Transitional//PL"
17   "http://www.w3.org/TR/2000/REC-xhtml1-
     20000126/DTD/xhtml1-transitional.dtd">
18   <html xmlns="http://www.w3.org/1999/xhtml">
19   <head>
20      <title>Totalizator piłkarski</title>
21   </head>
22   <body>
23   <?php
24
25   // Ustawienie stałej week.
26   define ("WEEK", $week);
27
28   /* ---------------------------- */
29   /* Zapis zwycięzców w pliku.    */
30   /* ---------------------------- */
31
32   function write_data ($value, $key, $fp) {
     // Funkcja write_data zapisuje typy
     w pliku.
33
34      static $first_line;
35
36      if (!$first_line) { // Zapisanie
        pierwszego wiersza-atrapy, ale tylko
        raz.
37         fputs ($fp, "Wiersz atrapa\n");
38         $first_line = TRUE;
39      }
40
```

Skrypt 1.3. *ciąg dalszy*

Kod

```
41       fputs ($fp, "$value\n"); // Dodanie
         znaku return, aby każdy zapis znalazł
         się w osobnym wierszu.
42    }
43
44    $file = '2001/picks_' . WEEK .
      '_Winners_.txt'; // Wskazanie nazwy
      pliku w formacie 'picks_1_Winners_.txt'.
45
46    if ($fp = @fopen ($file, "w")) {
      // Przeczytanie pliku do tablicy.
47
48       array_walk ($winners, 'write_data',
         $fp); // Wysłanie całej tablicy
         do funkcji write_data.
49       fclose ($fp); // Zamknięcie pliku.
50
51       echo 'Zapisano zwycięzców!<br></br>';
52
53    } else { // Wyświetlenie błędu w przypadku,
      gdyby otwarcie pliku było niemożliwe.
54       echo "Nie można otworzyć pliku $file!
         Proszę upewnić się, że zmienna week
         ma wartość.<br></br>";
55    }
56
57
58    /* ---------------------------- */
59    /* Pobranie typów graczy.       */
60    /* ---------------------------- */
61
62    function read_picks ($dp, $directory) {
      //Czynność tę wykonuje funkcja read_picks.
63
64       global $picks;
65
66       if ($file_name = readdir ($dp)) {
         // Sprawdzenie każdego pliku w katalogu.
67
68          if (substr($file_name, 0, 6) ==
            "picks_") { // Kontynuacja,
            w przypadku odnalezienia pliku
            z typami.
69
70             $parse_file_name = explode
               ("_", $file_name);
               // Przekształcenie nazwy pliku
               na odpowiedni format.
71
72             if (($parse_file_name[1] ==
               WEEK) and ($parse_file_name[2]
               != "Winners")) { // Jeżeli to
               nie jest plik z danymi
               o zwycięzcach, a numer tygodnia
               jest właściwy...
```

Skrypt 1.3. *ciąg dalszy*

Kod

```
73
74                $username =
                  $parse_file_name[2];
75                $the_file = $directory .
                  $file_name;
76                $users_picks = file
                  ($the_file);
77                $picks[$username] =
                  $users_picks;
78             }
79          }
80
81          read_picks ($dp, $directory); //
            Kolejna iteracja funkcji.
82       }
83    }
84
85    $picks = array(); // Zainicjowanie
      głównej zmiennej.
86
87    $directory = '2001/';
88    $dp = opendir ($directory);
89    read_picks ($dp, $directory);
90    closedir ($dp);
91
92
93    /* ------------------------------- */
94    /* Obliczenie tygodniowych wyników. */
95    /* ------------------------------- */
96
97    while (list ($key, $value) = each
      ($picks)) { // Pętla dla wyników
      w głównej tablicy.
98
99       $wins = 0; // Ustawienie na wartość 0
         dla każdej tablicy.
100      $losses = 0; // Ustawienie na wartość
         0 dla każdej tablicy.
101
102      while (list ($key2, $value2) = each
         ($value)) { // Pętla dla każdej
         tablicy użytkownika.
103
104         $value2 = substr ($value2, 0,
            (strlen($value2) - 1));
            // Usunięcie znaku return.
105
106         if (($value2 == $winners[$key2])
            and ($key2 != 0)) { // Jeżeli typ
            użytkownika jest właściwy,
            przyznajemy mu punkt, ale nie
            liczymy wiersza nr 1.
107            $wins++;
108         } elseif ($key2 != 0 ) {
109            $losses++;
```

Skrypt 1.3. *ciąg dalszy*

Kod

```
110         }
111     }
112
113     $results[$key] = array ('wins' =>
        $wins, 'losses' => $losses);
        // Zapisanie wyników każdego
        z użytkowników w tablicy.
114 }
115
116 ksort ($results); // Uporządkowanie
    tablicy alfabetycznie.
117
118
119 /* ------------------------------ */
120 /* Zapisanie tygodniowych wyników. */
121 /* ------------------------------ */
122
123 function write_results ($value, $key,
    $fp) { // Zapisuje wyniki do pliku.
124
125     static $first_line2;
126
127     if (!$first_line2) { // Zapisanie
        pierwszego wiersza-atrapy, ale tylko
        raz.
128         fputs ($fp, "Gracz\tpoprawne\
            tbłędne\n"); // Dodanie znaku
            return tak, aby każdy zapis
            znalazł się w osobnym wierszu.
129         $first_line2 = TRUE;
130     }
131
132     $data = implode ("\t", $value);
133     fputs ($fp, "$key\t$data\n");
        // Dodanie znaku return tak, aby
        każdy zapis znalazł się w osobnym
        wierszu.
134 }
135
136 $file2 = '2001/results_' . WEEK .
    '_.txt'; // Wskazanie nazwy pliku
    w formacie 'results_1_.txt'.
137
138 if ($fp2 = @fopen ($file2, "w")) {
    // Otwarcie pliku do zapisu.
139
140     array_walk ($results,
        'write_results', $fp2); // Wysłanie
        zawartości całej tablicy do funkcji
        write_results.
141     fclose ($fp2); // Zamknięcie pliku.
142     echo 'Wyniki zapisano.<br></br>';
143
```

Skrypt 1.3. *ciąg dalszy*

Kod

```
144 } else { // Wyświetlenie komunikatu
    o błędzie, jeżeli otwarcie pliku
    nie powiodło się.
145
146     echo "Nie można otworzyć pliku
        z tygodniowymi wynikami,
        $file2.<br></br>";
147 }
148 ?>
149 </body>
150 </html>
```

Poruszyliśmy już sporo zagadnień, dotyczących dobrej struktury kodu oraz właściwego jego dokumentowania. Zwięzły dokument poświęcony temu tematowi — Standard kodowania w PHP (*PHP Coding Standard*) można znaleźć pod adresem *www.DMCinsights.com/phpadv /coding_standard.php* (rysunki 1.1 oraz 1.2).

W dokumencie tym opisano formalne reguły zapisywania kodu w PHP wraz z ich uzasadnieniem. W tej książce będę wskazywał miejsca, gdzie przestrzegam konwencji, i te, gdzie je łamię. Należy jednak pamiętać o trzech sprawach.

Po pierwsze, chciałbym podkreślić, że bez względu na to, czy będziemy przestrzegać standardu kodowania PHP, czy moich przyzwyczajeń, najważniejsze jest zachowanie spójności. Niespójność nie tylko jest na bakier z każdym standardem, ale prowadzi do błędów i z pewnością spowoduje konieczność spędzenia dodatkowych godzin w czasie prób uruchamiania. Poza tym, jeżeli pracujemy w zespole, musimy opracować plan, którego powinni przestrzegać wszyscy członkowie zespołu. Jeżeli programujemy dla klienta, powinniśmy pozostawić przejrzystą, spójną dokumentację, tak aby klient lub inny programista mógł z łatwością zrozumieć naszą pracę.

Po drugie, struktura kodu i jego dokumentowanie jest elementem, który powinniśmy stosować od momentu rozpoczęcia kodowania i kontynuować w czasie pracy. Próby tworzenia komentarzy po zapisaniu kodu nigdy nie będą tak skuteczne, a często w ogóle nie będą podjęte. To, co wydaje się oczywiste w trakcie tworzenia, stanie się niejasne trzy miesiące później. Jeżeli jeszcze tego nie doświadczyliśmy, to z pewnością zdarzy się to nam pewnego dnia.

Po trzecie, ze względu na ograniczenia formatu książkowego, skrypty opracowane w tej książce od tego momentu nie będą tak dobrze ułożone, ani udokumentowane, jak to być powinno. Istnieje przecież limit wykorzystania cennej przestrzeni książkowej na zapiski typu `//Developed by: Larry E. Ullman.`

Osobiście zawsze stosuję się do trzech wymienionych niżej konwencji (będę to robił także w tej książce):


PHP Coding Standard

Last Modified: 2002-03-04

The PHP Coding Standard is with permission **based on** Todd Hoff's C++ Coding Standard.
Rewritten for PHP by Fredrik Kristiansen / DB Medialab, Oslo 2000-2001.

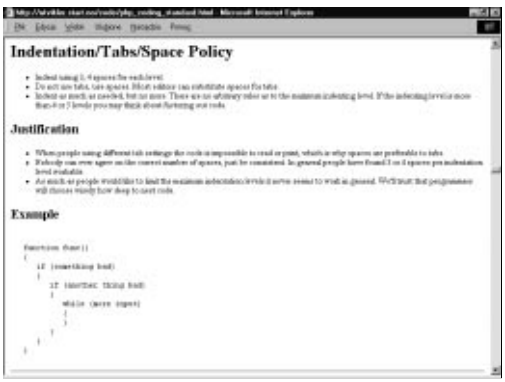
Using this Standard. If you want to make a local copy of this standard and use it as your own, you are perfectly free to do so. That's why we made it! If you find any errors or make any improvements please email me the changes so I can merge them in. Recent Changes

Before you start please verify that you have the most recent document.
You can also download a thin standard as a word document (maintained by Chris Hubbard).

Contents

- Introduction
  - Standardization is Important
  - Interpretation
  - Accepting an Idea
- Names
  - Make Names Fit
  - No All Upper Case Abbreviations
  - Class Names
  - Class Library Names
  - Method Names
  - Class Attribute Names
  - Method Argument Names
  - Variable Names
  - Array Element
  - Reference Variables and Functions Returning References
  - Global Variables
  - Define Names and Global Constants

Rysunek 1.1. *„PHP Coding Standard" (Standard kodowania w PHP) jest przewodnikiem „gramatycznym" programowania w PHP*


Indentation/Tabs/Space Policy

Justification

Example

Rysunek 1.2. *„PHP Coding Standard" (Standard kodowania w PHP) doskonale opisuje reguły, uzasadnia je i demonstruje, w jaki sposób przestrzegać tych zasad podczas tworzenia kodu*

Rysunek 1.3. *Program PHPDoc opracowano w celu zautomatyzowania procesu dokumentowania kodu. Program wykorzystuje wyrażenia regularne do analizowania kodu i wstawiania opisów*

Rysunek 1.4. *Jeżeli programujemy z wykorzystaniem biblioteki PEAR, powinniśmy przestrzegać specyficznych zasad dokumentowania i tworzenia struktury kodu (http://pear.php.net/manual/en/standards.php).*

- Nazwy zmiennych powinny być zapisywane małymi literami ze znakiem podkreślenia jako separatorem słów. Niektórzy zalecają, aby nazwy zmiennych globalnych zaczynały się na literę *g*, ale ja nie stosuję tej zasady.
- Ponieważ w języku XML skrócone wersje znaczników (<? oraz ?>) są wykorzystywane, zatem jeżeli korzystamy z XML-a, powinniśmy stosować formalne znaczniki PHP (<?php ?>). Przekonamy się o tym w rozdziale 9., „XML". Zalecam, aby zawsze stosować znaczniki formalne, ponieważ jest to najlepszy sposób zachowania zgodności pomiędzy serwerami.
- Dla stron, które mają być interpretowane jako skrypty PHP, zaleca się wykorzystywanie rozszerzenia *.php* (dla plików włączanych, jak np. klas i stron konfiguracyjnych, można wykorzystywać inne rozszerzenia). Dopuszczalne rozszerzenia plików są określone przez konfigurację serwera WWW, ale w społeczności PHP domyślnie przyjmuje się rozszerzenie *.php* (rozszerzenie stosowane w PHP w wersji 3. — *.php3* będzie działać w większości serwerów z PHP4, ale wydaje się, że ze względu na istnienie nowych wersji, stosowanie tego rozszerzenia jest nieuzasadnione).

Wskazówki

- Program PHPDoc jest aplikacją typu open source, wspomagającą pracę w PHP. Opracowano go na podstawie popularnego programu JavaDoc (służącego do dokumentowania programów w języku Java). Program ten jest przeznaczony do wspomagania procesu dokumentowania kodu. Więcej informacji na temat tego programu można znaleźć pod adresem *www.PHPDoc.de* (rysunek 1.3).
- Jeżeli zamierzamy stworzyć kod, który będzie wykorzystywany wraz z pakietem PEAR (zobacz rozdział 12., „Rozszerzanie PHP"), wówczas powinniśmy przestrzegać reguł formatowania obowiązujących dla tego pakietu (rysunek 1.4).

Tablice

Tablice są specjalnego rodzaju zmiennymi, które mogą działać jak tabele w arkuszu kalkulacyjnym. Ze względu na możliwości, jakie oferują, oraz elastyczność, tablice są powszechnie wykorzystywane w zaawansowanym programowaniu w PHP. W tym podrozdziale opiszemy kilka najbardziej powszechnych funkcji obsługi tablic.

Nowością w PHP4 jest konstrukcja `foreach`, którą opracowano w celu umożliwienia łatwiejszego dostępu do wszystkich indeksów i wartości tablicy. Zakładając, że mamy tablicę o nazwie `$array`, możemy uzyskać dostęp do każdego jej elementu za pomocą następującego kodu:

```
foreach ($array as $key => $value) {
     // kod
}
```

Osobiście do obsługi tablic wykorzystuję konstrukcję `foreach` i będę to robił także w tej książce. Jeżeli korzystamy z wcześniejszej wersji PHP, będziemy musieli przepisać powyższy kod na następującą postać:

```
reset($array);
while (list($key, $value) = each($array)) {
    // kod
}
```

Podstawową różnicą pomiędzy konstrukcją `foreach` oraz `while` jest fakt, że pierwsza z tych pętli automatycznie ponownie ustawia tablicę na pozycję początkową, natomiast druga tego nie robi.

Inną funkcją obsługi tablic, którą często wykorzystuję, chociaż nie jest to nowość wersji PHP4, jest `array_walk()`. Funkcja ta pozwala na podstawienie każdego elementu tablicy do funkcji zdefiniowanej przez użytkownika.

```
function print_albums($value, $key) {
    echo $value. " <br/>\n
}
$array=array("Pablo Honey","The Bends", "OK
➥Computer", "Kid A", "Amnesiac");
array_walk($array,'print_albums');
```

Zgodnie ze składnią funkcja `array_walk()` pobiera tablicę jako pierwszy parametr oraz nazwę funkcji bez żadnych nawiasów ani argumentów jako drugi parametr. W czasie wykonywania iteracji funkcji `array_walk()` funkcja będzie otrzymywać kolejne indeksy tablicy oraz wartości. Powyższy kod spowoduje wygenerowanie następującego wyniku:

```
Pablo Honey <br />
The Bends <br />
OK Computer <br />
Kid A <br />
Amnesiac <br />
```

Ze skryptów znajdujących się w dalszej części książki dowiemy się, w jaki sposób przekazać więcej argumentów funkcji wywoływanej za pomocą `array_walk()`.

Zwróćmy uwagę, że funkcja `array_walk()` działa jedynie z funkcjami definiowanymi przez użytkownika i nie będzie działać z wbudowanymi funkcjami PHP. Należy też pamiętać o konieczności wykonania funkcji `reset` w PHP4, jeżeli chcemy skorzystać z tablicy ponownie.

Dla zademonstrowania niektórych zaawansowanych technik obsługi tablic zapiszemy pierwszą część aplikacji obsługującej totalizator sportowy. Jej przeznaczeniem jest obsługa profesjonalnego futbolu amerykańskiego, ale można ją z łatwością przystosować do dowolnej innej dziedziny. Pierwsze dwa skrypty dynamicznie wygenerują formularz, gdzie użytkownicy będą mogli wprowadzać swoje typy. Ponieważ nie wszyscy mamy dostęp do bazy danych, aplikacja ta do zapisu informacji będzie wykorzystywać jednorodny plik tekstowy.

Skrypt 1.4. *Dane dla totalizatora będą zapisywane w plikach tekstowych rozdzielanych tabulatorami jak plik przedstawiony niżej:*

Kod

```
1    Goście      Gospodarze
2    New Orleans   Buffalo
3    New England   Cincinnati
4    Seattle      Cleveland
5    Tampa Bay      Dallas
6    Detroit      Green Bay
7    Oakland      Kansas City
8    Carolina      Minnesota
9    Indianapolis   NY Jets
10    Pittsburgh      Jacksonville
11    Chicago      Baltimore
12    St. Louis      Philadelphia
13    Washington      San Diego
14    Atlanta      San Francisco
15    Miami      Tennessee
16    NY Giants      Denver
```

Aby skorzystać z tablic:

1. Utworzymy plik tekstowy w edytorze tekstów.
2. Zapiszemy wiersz nagłówka (skrypt 1.4):

   ```
   Gospodarze   Goście
   ```

   W pliku tekstowym najpierw będzie wymieniona drużyna gości, a następnie drużyna gospodarzy. Każdy mecz będzie zajmował oddzielny wiersz. Wiersz nagłówkowy służy do wyznaczenia układu tablicy oraz działa jako obejście niektórych specyficznych właściwości tablic. Kiedy ten plik będzie wczytany do tablicy, rozpocznie się ona — jak zwykle w przypadku tablic — od indeksu 0. Ponieważ wolę, aby indeksy tablic były zgodne z numerami gier, a chciałbym numerować wiersze od 1. do 15. (lub 14., 16. itp.), dlatego opuszczam jeden wiersz. Tak więc pierwsza gra jest wymieniona w drugim wierszu, zatem będzie umieszczona pod indeksem 1.
3. Wpiszemy kolejne gry:

   ```
   New Orleans   Buffalo
   New England   Cincinnati
   Seattle       Cleveland
   Tampa Bay     Dallas
   Detroit       Green Bay
   Oakland       Kansas City
   Carolina      Minnesota
   Indianapolis  NY Jets
   Pittsburgh    Jacksonville
   Chicago       Baltimore
   St. Louis     Philadelphia
   Washington    San Diego
   Atlanta       San Francisco
   Miami         Tennessee
   NY Giants     Denver
   ```

   Gry są wyszczególnione w formacie Drużyna gości [TAB] Drużyna gospodarzy [RETURN] (bez spacji). Znaku tabulacji użyjemy w celu rozróżnienia dwóch elementów każdego wiersza.
4. Zapiszemy plik jako *week1.txt*.

5. Podobne pliki stworzymy dla pozostałych tygodni w sezonie, nazywając je na przykład *week2.txt*, aż do *week17.txt*. Jeżeli śledzimy ten przykład bez zmian, możemy pobrać wszystkie te pliki z witryny pod adresem *www.DMCinsights.com/phpadv*. W plikach tych nie należy wprowadzać żadnych komentarzy, ponieważ kolidowałoby to ze sposobem ich wykorzystania (i tak będą odczytywane tylko przez PHP).

6. Stworzymy katalog *2001* na naszym serwerze WWW. Możemy użyć dowolnej innej nazwy, która będzie określała sezon jako całość.

7. Uprawnienia do katalogu ustawimy jako 777 (odczyt, zapis, wyszukiwanie dla wszystkich). Ze względów bezpieczeństwa, jeżeli to możliwe, powinniśmy umieścić ten katalog poniżej katalogu głównego serwera. Jeżeli już to zrobimy, powinniśmy upewnić się, że stosujemy właściwe odwołania do tego katalogu we wszystkich przedstawionych poniżej skryptach (tzn. używając odwołania *../2001* zamiast *2001/*).

8. Umieścimy wszystkie pliki dla poszczególnych tygodni w katalogu *2001*.

Teraz, kiedy utworzyliśmy dane zawierające informacje o wszystkich grach sezonu, czas zapisać skrypt, który przekształci te dane na format HTML.

9. W edytorze tekstowym stworzymy nowy dokument HTML (skrypt 1.5):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
➥Transitional//PL"
      "http://www.w3.org/TR/2000/REC-xhtml1-
➥20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Totalizator piłkarski</title>
</head>
```

Skrypt 1.5. *Ten skrypt dynamicznie generuje formularz na podstawie pliku w formacie zwykłego tekstu*

Kod

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//PL"
"http://www.w3.org/TR/2000/REC-xhtml1-20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
        <title>Totalizator piłkarski</title>
</head>
<body>
<?php # make_picks_form.php
$week=1;
$file = '2001/week' . $week . '.txt'; // Wskazanie pliku na podstawie numeru tygodnia.

if ($games == @file ($file)) { // Wczytanie pliku do tablicy.

        // Po przeczytaniu pliku, wyświetlenie formularza.
        echo '<form action="store_picks.php" method="post">
<table border="1" width="80%" cellspacing="4" cellpadding="4" align="center">
        <tr align="center" valign="top">
                <td colspan="3" width="100%" align="center" valign="top" nowrap="nowrap">
                Wprowadź nazwę użytkownika:
                <input type="text" name="username" size="20" maxlength="20"></td>
        </tr>
                ';

foreach ($games as $key => $array) { // Zastosowanie tablicy do wyświetlenia każdej gry.

    if ($key != 0) { // Wiersza Goście   Gospodarze nie wyświetlamy.

      $second_array = explode ("\t", $array); // Przekształcenie każdego wiersza tablicy
      w oddzielną tablicę.
      $away = $second_array[0]; // Pierwszy element odpowiada zespołowi gości.
      $home = substr ($second_array[1], 0, (strlen($second_array[1]) - 1)); //
      Drugi element odpowiada zespołowi gospodarzy, ale musimy usunąć znak return.

     echo "        <tr align=\"center\" valign=\"top\">
 <td align=\"left\" valign=\"top\" nowrap=\"nowrap\"><input type=\"radio\"
 name=\"picks[$key]\" value=\"$away\">$away</td>
 <td align=\"center\" valign=\"top\">przeciw</td>
 <td align=\"right\" valign=\"top\" nowrap=\"nowrap\"><input
 type=\"radio\" name=\"picks[$key]\" value=\"$home\">$home</td>
 </tr>\n";
    }
 }

 // Uzupełnienie formularza i tabeli.
 echo "        <tr align=\"center\" valign=\"top\">
   <td colspan=\"3\" width=\"100%\" align=\"center\" valign=\"top\"><input
   type=\"submit\" name=\"submit\" value=\"Wyślij!\">
 </tr>
</table>
<input type=\"hidden\" name=\"week\" value=\"$week\">
</form>\n";

} else { // Wyświetlenie komunikatu o błędzie, jeżeli otwarcie pliku nie powiodło się.
    echo "Nie można otworzyć pliku $file! Należy upewnić się, czy zmienna week ma ustawioną
    wartość.<br></br>";
}
?>
</body>
</html>
```

Instrukcje echo, print oraz cudzysłowy

W PHP istnieje kilka sposobów przesyłania tekstu do przeglądarki. Najbardziej popularne sposoby polegają na wykorzystaniu instrukcji `print` oraz `echo`. Istnieją tylko pewne subtelne różnice w sposobie działania tych dwóch instrukcji. W jakich sytuacjach należy stosować którą, jest raczej kwestią indywidualnego wyboru niż czegokolwiek innego.

Jedna z różnic polega na tym, że instrukcja echo jest minimalnie, prawie niezauważalnie szybsza niż instrukcja print. Wynika to stąd, że po pomyślnym wykonaniu instrukcja print zwraca wartość (1 oznacza, że instrukcja zakończyła się pomyślnie), natomiast instrukcja echo tego nie robi.

Właściwością funkcji echo, której nie znajdziemy w funkcji print, jest możliwość łatwego umieszczania zmiennych w kodzie HTML. Zazwyczaj (w naszych skryptach) fragment formularza zapisanego w formacie HTML ma następującą postać:

```
<input type="text" name="username" value="<?php echo $username; ?>"
```

W takim przypadku możemy zastosować dwa skróty. Po pierwsze, zawsze możemy pominąć ostatni średnik w skrypcie PHP, chociaż korzystanie z tego nie jest dobrą praktyką. Poza tym, stosując instrukcję echo, możemy skrócić kod, wpisując znak równości po początkowym znaczniku PHP (ale tylko w przypadku wykorzystywania krótkiej postaci znacznika). Tak więc zapis:

```
<?php echo $username; ?>
```

przyjmie postać:

```
<?= $username ?>
```

Ważniejsze od tego, z której funkcji skorzystamy (technicznie to nie są funkcje, a konstrukcje języka), jest odpowiedź na pytanie, czy skorzystamy z apostrofów, czy też z cudzysłowów.

W obu instrukcjach echo oraz print można korzystać zarówno z apostrofów, jak też z cudzysłowów. Różnica polega na sposobie interpretowania zmiennych oraz na tym, że niektóre znaki należy poprzedzić znakiem odwrotnego ukośnika. Użycie apostrofu spowoduje wyświetlenie zawartości w postaci takiej, jak jest i wymaga poprzedzenia znakiem odwrotnego ukośnika jedynie samego znaku apostrofu. Tak więc instrukcja:

```
echo '<a href="index.php">Home</a>';
```

zadziała bez problemu. Odpowiednik tej instrukcji z zastosowaniem cudzysłowów przyjąłby następującą postać:

```
echo "<a href=\"index.php\">Home</a>";
```

Aby w źródle HTML znalazł się znak cudzysłowu (co jest konieczne dla zachowania poprawności składni), należy go poprzedzić znakiem odwrotnego ukośnika. Z tego względu wysyłając kod HTML do przeglądarki, stosuję instrukcję echo z apostrofem.

Jednym z problemów podczas wykorzystywania apostrofów jest sposób interpretacji zmiennych i sekwencji specjalnych, które w tym przypadku są traktowane dosłownie, co oznacza, że instrukcje postaci:

```
$zmienna='wartosc';
echo '$zmienna \n';
```

spowodują wyświetlenie ciągu $zmienna \n, a nie wartości zmiennej, po której następuje znak końca wiersza, co nastąpi w wyniku zastosowania poniższego kodu:

```
$zmienna ='wartosc';
echo "$zmienna \n";
```

Instrukcje echo, print oraz cudzysłowy *(ciąg dalszy)*

Ten sam problem występuje w czasie przypisywania wartości do zmiennej lub przy odwołaniach do indeksów tablic.

```
$zmienna ='wartosc';
echo $tablica['$zmienna']; // Odnosi się do elementu pod indeksem "$zmienna".
echo $tablica["$zmienna"]; // Odnosi się do elementu pod indeksem "wartosc".
```

Biorąc pod uwagę sposób wyświetlania zmiennych, powinienem także wyjaśnić sposób interpretowania w PHP tablic wielowymiarowych. Zazwyczaj element tablicy wielowymiarowej (tablicy złożonej z tablic) możemy zlokalizować, stosując konstrukcję:

```
$nazwa_tablicy['nazwa_tabl_wewn']['indeks_tabl_wewn']
```

Ale próba wyświetlenia tego elementu za pomocą instrukcji

```
echo "Wartość wynosi $nazwa_tablicy['nazwa_tabl_wewn']['indeks_tabl_wewn']";
```

nie zadziała. W takim przypadku mamy dwie możliwości. Po pierwsze, możemy użyć operatora konkatenacji, aby opuścić cudzysłów. Niestety, w niektórych systemach (np. Mac OS X) taka konstrukcja nie zadziała:

```
echo "Wartość wynosi". $nazwa_tablicy['nazwa_tabl_wewn']['indeks_tabl_wewn']";
```

Możemy też wykorzystać nawiasy klamrowe wewnątrz cudzysłowów:

```
echo "Wartość wynosi {$nazwa_tablicy['nazwa_tabl_wewn']['indeks_tabl_wewn']}";
```

Dotyczy to zarówno instrukcji `echo`, jak też instrukcji `print`.

Doskonały przewodnik dotyczący ciągów znaków, cudzysłowów, apostrofów itp. zagadnień można znaleźć pod adresem *www.zend.com/zend/tut/using-strings.php*.

10. Rozpoczniemy wpisywanie treści dokumentu HTML oraz umieścimy inicjujący znacznik PHP:

```
<body>
<?php
```

11. Wskazujemy plik, który będzie wykorzystywany:

```
$file='2001/week' . $week. '.txt';
```

Zakłada się, że do skryptu zostanie przekazana zmienna `$week` o wartości od 1 do 17 (w moim przypadku). Tak więc wygenerowanie formularza dla każdego tygodnia wymaga jedynie przekazania do tego skryptu numeru tego tygodnia.

12. Utworzymy instrukcję warunkową, w której podejmiemy próbę odczytania określonego pliku do tablicy:

```
if($games=@file ($file)) {
```

Jeżeli skrypt prawidłowo odczyta plik, wówczas zostanie utworzona tablica `$games`, zawierająca informacje o wszystkich grach danego tygodnia. Dzięki zastosowaniu znaku `@` spowodujemy, że nie będą wyświetlane komunikaty o błędach w przypadku, gdy odczytanie pliku nie powiedzie się (porównajmy rysunek 1.5, gdzie nie zastosowano znaku `@`, z rysunkiem 1.6, gdzie zastosowano ten znak).

13. Rozpoczniemy formularz HTML.

```
echo 'form action="store_picks.php" method=
➥"post">
<table border="1" width="80%" cellspacing=
➥"4" cellpadding="4" align="center">
<tr align="center" valign="top">
<td colspan="3" width="100%" align="center"
➥valign="top" nowrap="nowrap">Wprowadź
➥nazwę użytkownika:
 <input type="text" name="username"
➥size="20" maxlength="20"></td>
 </tr>
 ';
```

Rysunek 1.5. *Ponieważ nie wprowadziliśmy wartości zmiennej $week (zobacz wiersz adresu), to skrypt nie mógł odczytać pliku. Na rysunku 1.6 pokazano uzyskany wynik, kiedy zastosowano znak @ w celu wyłączenia komunikatu o błędzie generowanego przez PHP*

Rysunek 1.6. *Zastosowanie znaku @ umożliwia wyłączenie wyświetlania mylących i brzydkich technicznych komunikatów o błędach*

Formularz HTML jest dosyć prosty, służy do wprowadzania jedynie nazwy użytkownika oraz typu dla każdej gry. Później można opracować system uwierzytelniania, który będzie działać, zanim użytkownik wykona ten krok (system ten mógłby następnie automatycznie wprowadzać lub zapisywać nazwy użytkowników).

- Uwaga: Zakończenie instrukcji echo kończącym znakiem apostrofu w osobnym wierszu może wydawać się dziwne, ale w rezultacie kod HTML staje się bardziej czytelny. Dzięki niemu mamy pewność, że następna instrukcja HTML rozpocznie się w osobnym wierszu, a nie od kończącego `</tr>`.

14. Utworzymy pętlę, przeglądającą tablicę, wpisując każdą grę do formularza:

```
foreach ($games as $key => $array) {
if ($key != 0) {
$second_array = explode ("\t", $array);
$away = $second_array[0];
$home = substr ($second_array[1], 0,
➥(strlen($second_array[1]) - 1));
echo "        <tr align=\"center\"
➥valign=\"top\">
<td align=\"left\" valign=\"top\"
➥nowrap=\"nowrap\"><input type=\"radio\"
➥name=\"picks[$key]\"
➥value=\"$away\">$away</td>
<td align=\"center\"
➥valign=\"top\">przeciw</td>
<td align=\"right\" valign=\"top\"
➥nowrap=\"nowrap\"><input type=\"radio\"
➥name=\"picks[$key]\"
➥value=\"$home\">$home</td>

</tr>\n";
}
}
```

Pierwszy wiersz w tym kodzie stanowi nowa konstrukcja `foreach`, której użycie stanowi łatwą i w pewnym sensie szybszą metodę obsługi tablic.

Drugi wiersz zapewnia, że obsługiwany element nie jest pierwszym elementem w pliku. Jak wynika z przykładu pokazanego wyżej (skrypt 1.4), elementem tablicy pod indeksem `0` jest wiersz o wartości `Goście [TAB] Gospodarze`, którego nie będziemy wyświetlać.

Jeżeli nie jest to pierwszy element tablicy, wówczas zostanie on podzielony za pomocą funkcji `explode()` na poszczególne części (drużynę gości oraz drużynę gospodarzy). Każda drużyna jest formalnie przyporządkowana osobnej zmiennej (za każdym razem obcinany jest znak RETURN z nazwy drużyny gospodarzy).

Ostatecznie wyświetlany jest wiersz tabeli złożony z uzyskanych informacji.

15. Uzupełnimy kod formularza HTML oraz tabeli.

```
echo "<tr align=\"center\" valign=\"top\">
<td colspan=\"3\" width=\"100%\"
➥align=\"center\" valign=\"top\"><input
➥type=\"submit\" name=\"submit\"
➥value=\"Wyślij!\">
</tr>
</table>
<input type=\"hidden\" name=\"week\"
➥value=\"$week\">
</form>\n";
```

Należy pamiętać, aby zastosować typ `HIDDEN` w celu przechowania numeru tygodnia, aby był on dostępny w skrypcie *store_picks.php*, który obsługuje formularz.

16. Zakończymy główną instrukcję warunkową i zamkniemy znaczniki PHP i HTML.

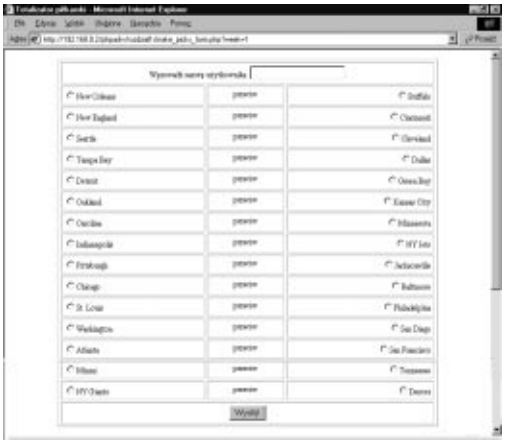
```
} else {
echo "Nie można otworzyć pliku $file!
➥Należy upewnić się, czy zmienna week ma
➥ustawioną →wartość.<br></br>";
}
?>
</body>
</html
```

Komunikat o błędzie, wygenerowany przez powyższy kod, jest przeznaczony bardziej dla programisty niż dla użytkownika (prawdopodobnie nie będziemy chcieli wyświetlać nazwy pliku w komunikacie o błędzie, który może być przeglądany przez wszystkich). Komunikat ten można przetworzyć na bardziej przyjazny dla użytkownika w zależności od sposobu działania naszej aplikacji.

17. Zapisujemy plik jako *make_picks_form.php* i kopiujemy ten plik na nasz serwer, do tego samego katalogu, gdzie utworzyliśmy katalog *2001*. Ładujemy go w przeglądarce WWW, dodając do adresu URL ciąg `?week=1`, aby skrypt mógł działać poprawnie (rysunki 1.7 oraz 1.8). Na razie nie korzystamy z przycisku *Wyślij*.

Wskazówki

- W rozdziale 5., „Projektowanie aplikacji WWW”, zostanie omówione bardziej szczegółowo użycie symbolu `@`.
- W rozdziale 4., „Bezpieczeństwo”, opracujemy system rejestracji użytkowników oraz uwierzytelniania, którego będziemy mogli używać wraz z tymi skryptami oraz z dowolnymi innymi.



Rysunek 1.7. *Oto formularz utworzony przez skrypt make_picks_form, umożliwiający wytypowanie zwycięzcy we wszystkich 15 grach*

Rysunek 1.8. *Rozważne wykorzystywanie instrukcji echo oraz spacji w skryptach PHP powoduje, że kod źródłowy HTML staje się bardziej czytelny*

Stałe

Stałe to specjalny rodzaj zmiennych, któremu — moim skromnym zdaniem — nie poświęca się dostatecznie dużo uwagi. Stałe, jak wskazuje nazwa, zawierają wartość, która nie zmienia się, ani też nie może się zmieniać w czasie działania skryptu. Ponadto dodatkową zaletą stałych jest fakt, że w danym zakresie są one globalne, co oznacza, że są automatycznie dostępne wewnątrz funkcji.

W nazwach stałych wielkość liter ma znaczenie, podobnie jak w nazwach wszystkich zmiennych w PHP. Zasadą jest zapisywanie nazw stałych tylko wielkimi literami. W czasie nadawania nazw zmiennym należy stosować te same zasady, jak w przypadku zmiennych — litera (z wyjątkiem początkowego znaku dolara) lub znak podkreślenia, po którym następują litery, cyfry lub znaki podkreślenia.

Stałą tworzy się za pomocą funkcji `define()` w następujący sposób:

```
define("NAZWA_STAŁEJ:,"wartość");
```

Stałą można zdefiniować tylko jako liczbę (całkowitą lub zmiennoprzecinkową), albo jako ciąg znaków. Nie można zdefiniować stałej jako tablicy lub jako obiektu, a po utworzeniu stałej jej wartości nie można aktualizować, tak więc nie można przekształcić stałej łańcuchowej na liczbę całkowitą. Stałą można uważać za niezmienną zmienną.

W PHP istnieje wiele wbudowanych stałych, włącznie z `PHP_OS` (system operacyjny), `PHP_VERSION` (np. *4.0.3pl1*), `TRUE` oraz `FALSE`. Dwie zmienne wbudowane — `__FILE__` oraz `__LINE__` oznaczające bieżącą nazwę pliku oraz numer wiersza — zmieniają się w czasie działania skryptu. Stałe te przydają się do uruchamiania diagnostycznego, o czym przekonamy się w rozdziale 5., „Projektowanie aplikacji WWW”.

Lubię wykorzystywać stałe tam, gdzie występują zmienne, które nie powinny się zmieniać, lub w przypadku, jeżeli zmienna ma być dostępna w całym skrypcie. Jako demonstrację zdefiniowania i wykorzystania stałych zapiszemy skrypt *store_picks.php* dla naszego totalizatora online. Skrypt ten będzie obsługiwał dane pochodzące ze strony *make_picks_form.php* utworzonej poprzednio.

Aby wykorzystać stałe:

1. W edytorze tekstowym utworzymy nowy dokument HTML (skrypt 1.6).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
➥1.0 Transitional//PL"
"http://www.w3.org/TR/2000/REC-xhtml1-
➥20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
     <title>Totalizator piłkarski</title>
</head>
```

2. Rozpoczniemy wpisywanie treści dokumentu HTML oraz umieścimy inicjujący znacznik PHP:

```
<body>
<? php
```

3. Zdefiniujemy stałą `WEEK`.

```
define("WEEK", $week);
```

Ponieważ na każdej ze stron wykorzystywana jest wartość zmiennej `$week`, chcemy zapewnić, żeby zmienna ta nie zmieniała wartości w czasie działania skryptu. Z tego powodu zmienną tę przekształcimy na stałą.

4. Utworzymy funkcję, zapisującą typy użytkowników.

```
function write_data ($value, $key, $fp) {
   fputs ($fp, "$value\n");
}
```

Funkcja `write_data()` będzie wykorzystywana w połączeniu z funkcją `array_walk()` zgodnie z tym, co napisano we wcześniejszej części tego rozdziału. Funkcja `write_data()` otrzymuje element tablicy, jej indeks oraz wskaźnik do pliku. Zawartość zapisaną do pliku będzie stanowić nazwa zespołu wybrana przez użytkownika zakończona znakiem RETURN. W celu wygenerowania znaku RETURN należy wykorzystać sekwencję `\n`, ponieważ alternatywna sekwencja `\r` powoduje problemy z odczytem pliku w niektórych systemach (pierwsza sekwencja generuje znak RETURN, natomiast druga — znak CR).

Skrypt 1.6. *Skrypt* store_picks.php *definiuje zmienną* `week` *jako stałą, aby zapewnić niezmienność jej wartości w czasie działania skryptu*

Kod

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN"
2   "http://www.w3.org/TR/2000/REC-xhtml1-
    20000126/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml">
4   <head>
5      <title>Totalizator piłkarski</title>
6   </head>
7   <body>
8   <?php
9   // Ten skrypt obsługuje dane ze skryptu
    make_picks_form.php.
10
11  // Ustawienie numeru tygodnia jako stałej.
12  define ("WEEK", $week);
13
14  // Funkcja write_data zapisuje typy
    w pliku.
15  function write_data ($value, $key, $fp) {
16     fputs ($fp, "$value\n"); // Dodanie
       znaku return, aby umieścić każdy
       element w oddzielnym wierszu.
17  }
18
19  $file = '2001/picks_' . WEEK . '_' .
    $username . '_.txt'; // Wskazanie pliku
    w formacie 'picks_1_Larry_.txt.
20
21  if ($fp = @fopen ($file, "w")) {
    // Otwarcie pliku do zapisu.
22
23     fputs ($fp, "Wiersz - atrapa\n");
       // Utworzenie wiersza - atrapy.
24     array_walk ($picks, 'write_data',
       $fp); // Wysłanie całej tablicy
       do funkcji write_data.
25     fclose ($fp); // Zamknięcie pliku.
26     echo 'Twoje typy zostały zapisane.';
27
28  } else { // Wyświetlenie komunikatu
    o błędzie, jeżeli otwarcie pliku nie
    powiodło się.
29        echo "Nie można otworzyć pliku
          $file! Należy upewnić się, czy
          zmienna week ma ustawioną wartość.
          <br></br>";
30  }
31  ?>
32  </body>
33  </html>
```

5. Wskazujemy plik, który będzie wykorzystywany

```
$file = '2001/picks_' . WEEK . '_' .
➥$username . '_.txt';
```

Nazwa pliku składa się z przedrostka `picks_`, po którym następuje numer tygodnia, znak podkreślenia, nazwa użytkownika i ponownie znak podkreślenia. Plik ma rozszerzenie *.txt*. Nazwy stałej nie można umieścić w apostrofach ani w cudzysłowach, ponieważ w takim przypadku PHP interpretowałby ją dosłownie jako `WEEK`, a nie jako wartość, którą stała reprezentuje. Jeżeli katalog *2001* umieścimy poza głównym katalogiem dokumentów WWW, wówczas w celu właściwego odwołania się do tego katalogu powinniśmy zastosować odpowiednią składnię.

6. Próbujemy otworzyć plik do zapisu.

```
if ($fp = @fopen ($file, "w")) {
```

Jeżeli ta instrukcja warunkowa przyjmie wartość `false`, wówczas należy sprawdzić, czy poprawnie ustawiliśmy uprawnienia do katalogu *2001* — nieprawidłowe ich ustawienie jest najbardziej prawdopodobną przyczyną błędu, ponieważ sam plik do tej pory nie istniał, zatem powinna istnieć możliwość zapisu do tego pliku. Inną prawdopodobną przyczyną może być użycie złego parametru, na przykład `a` zamiast `w`.

7. Zapisujemy dane do pliku.

```
fputs ($fp, "Wiersz - atrapa\n");
array_walk ($picks, 'write_data', $fp);
```

Pierwszym wierszem, który zapisujemy do pliku, jest wiersz — atrapa, a po nim, wykorzystując połączenie funkcji `array_walk()` ze zdefiniowaną uprzednio funkcją `write_data()`, zapisujemy pozostałe dane. Teoretycznie moglibyśmy zdefiniować dodatkową funkcję, która przed zapisem danych sprawdzałaby, czy wytypowano wszystkich zwycięzców.

8. Zamykamy plik, a po pomyślnym jego zamknięciu wysyłamy komunikat.

```
fclose ($fp); // Zamknięcie pliku.
echo 'Twoje typy zostały zapisane.';
```

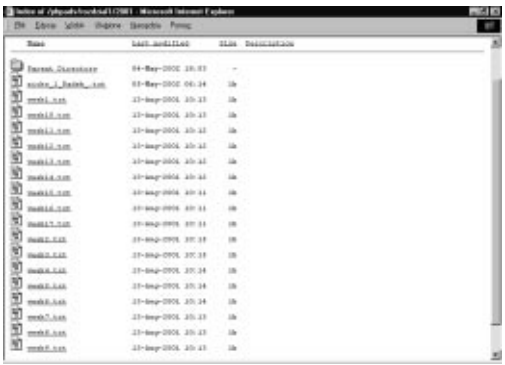
9. Kończymy instrukcję warunkową, PHP oraz HTML.

```
} else {
echo "Nie można otworzyć pliku $file!
➥Należy upewnić się, czy zmienna week
➥ma ustawioną wartość.<br></br>";
}
?>
</body>
</html>
```

10. Zapisujemy plik jako *store_picks.php* i kopiujemy go na serwer, do tego samego katalogu, gdzie zapisaliśmy skrypt *make_ picks_form.php* (skrypt 1.5). Uruchamiamy ten drugi skrypt w przeglądarce WWW (rysunek 1.7) tak, aby skrypt *store_picks.php* został wykonany (rysunek 1.9).

11. Sprawdzamy w katalogu 2001, czy utworzono nowy plik (rysunek 1.10). Możemy pobrać nowo utworzony plik i sprawdzić jego zawartość (skrypt 1.7).

Rysunek 1.9. *Jeżeli skrypt działa prawidłowo, wówczas użytkownik uzyska ten lakoniczny komunikat*



Rysunek 1.10. *Katalog 2001, który jest bazą danych plików tekstowych, zawiera wszystkie dane, włącznie z nowo utworzonym plikiem picks_1_NAZWA_txt*

Skrypt 1.7. *Po wykonaniu skryptu make_picks_form.php oraz store_picks.php zostanie utworzony poniższy plik, zawierający moje typy dla gier w pierwszym tygodniu*

Kod

```
1    Wiersz - atrapa
2    Buffalo
3    New England
4    Seattle
5    Tampa Bay
6    Green Bay
7    Kansas City
8    Minnesota
9    Indianapolis
10   Pittsburgh
11   Chicago
12   Philadelphia
13   San Diego
14   San Francisco
15   Miami
16   Denver
```

Wskazówki

- Aby sprawdzić, czy zdefiniowano stałą przed jej użyciem, można skorzystać z funkcji `defined()`:

  ```
  if (defined(NAZWA_STALEJ)) {...
  ```

  Należy uważać, aby nie pomylić funkcji `defined()` z funkcją `define()`, która służy do ustawiania wartości stałej.

- Aby obejrzeć listę wszystkich zdefiniowanych stałych, należy skorzystać z funkcji `get_defined_constants()`.

- Chociaż posiadanie katalogu z prawem zapisu w obrębie katalogu WWW (uprawnienia ustawione na 777), tak jak w powyższym przykładzie katalog *2001*, nie jest bezpieczne, to czasami jest to konieczność. Istnieją obejścia oraz elementy, na które należy zwracać uwagę w takich przypadkach. Zostaną one omówione w rozdziale 4., „Bezpieczeństwo”. Jeżeli istnieje możliwość umieszczenia katalogu z prawem zapisu poza katalogiem WWW, tak będzie lepiej, chociaż niektóre firmy świadczące usługi hostingu nie dadzą nam takiego wyboru.

Funkcje rekurencyjne i zmienne statyczne

Programując, z pewnością będziemy mieli okazję zdefiniowania i zastosowania własnej funkcji — jest to doskonałe narzędzie, które pomaga właściwie zorganizować nasz kod oraz zaoszczędzić czas. Elementem, z którego być może jeszcze nie korzystaliśmy, jest rekurencja wewnątrz funkcji.

Rekurencja to właściwość wywoływania funkcji w obrębie jej samej.

```
function nazwa_funkcji () {
// jakiś kod
   nazwa_funkcji();
}
```

W wyniku zastosowania rekurencji uzyskamy efekt, gdzie będziemy mogli wykorzystać funkcję zarówno zgodnie z jej przeznaczeniem, jak też jako pętlę.

Stosując tę technikę, należy uwzględnić jedną, ogromnie ważną uwagę. Należy upewnić się, że funkcja posiada warunek wyjścia. Na przykład funkcja o kodzie pokazanym poniżej będzie działać w nieskończoność:

```
function add_one($n) {
    $n++;
      add_one($n);
}
add_one(1);
```

Brak warunku określającego zakończenie obliczeń funkcji tworzy olbrzymi problem programistyczny — pętlę nieskończoną. Porównajmy funkcję pokazaną wyżej do następującej:

```
function count_to_100 ($n) {
  if($n <=100) {
    echo $n. "<br></br>";
  }
}
count_to_100(1);
```

Funkcja ta będzie wywoływać samą siebie dopóty, dopóki zmienna `$n` nie osiągnie wartości większej niż 100. W tym momencie działanie funkcji zakończy się.

Wykorzystując rekurencję lub dowolny skrypt, gdzie ta sama funkcja wykonuje się wiele razy, warto rozważyć zastosowanie instrukcji `static`. Użycie tej instrukcji powoduje, że wartości zmienne są pamiętane pomiędzy poszczególnymi wywołaniami funkcji bez konieczności użycia zmiennych globalnych. Funkcję `count_to_100()`, dla zapewnienia identycznego wyniku jej działania, można przepisać następująco:

```
function count_to_100() {
    static $n;
      if ($n<=100) {
         echo $n. "<br></br>";
      $n++;
      count_to_100();
    }
}
```

Jako kolejny krok na drodze do stworzenia aplikacji totalizatora piłkarskiego online zapiszemy skrypt, który wykorzystuje rekurencję oraz instrukcję `static`. Skrypt ten będzie umożliwiał administratorowi wprowadzenie zwycięzców każdego meczu, a następnie porównanie zwycięzcy z typami poszczególnych graczy.

Aby wykorzystać rekurencję oraz instrukcję static:

1. W edytorze tekstowym otwieramy skrypt *make_picks_form.php* (skrypt 1.5). Napiszemy skrypt nieco inny od tego, tak aby można było dokonać wyboru zwycięzców.
2. Modyfikujemy wiersze 15. – 21. oryginalnego dokumentu tak, aby formularz był obsługiwany przez inny skrypt oraz bez możliwości wprowadzania danych przez użytkowników (skrypt 1.8).

```
echo '<form action="process_winners.php"
➥method="post">
<table border="1" width="80%" cellspacing=
➥"4" cellpadding="4" align="center">
';
```

Chociaż formularz ten będzie bardzo podobny do formularza w skrypcie *make_picks_form.php*, to jednak będzie on obsługiwany inaczej — przez skrypt *process_winner.php*, który zapiszemy jako następny.

Skrypt 1.8. *Skrypt pick_winners.php to tylko nieznaczna modyfikacja jego poprzednika — skryptu make_picks_form.php*

Kod

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//PL"
2   "http://www.w3.org/TR/2000/REC-xhtml1-20000126/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml">
4   <head>
5           <title>Totalizator piłkarski</title>
6   </head>
7   <body>
8   <?php # make_picks_form.php
9
10  $file = '2001/week' . $week . '.txt'; // Wskazanie pliku na podstawie numeru tygodnia.
11
12  if ($games = file ($file)) { // Wczytanie pliku do tablicy.
13
14          // Po przeczytaniu pliku, wyświetlenie formularza.
15          echo '<form action="process_winners.php" method="post">
16  <table border="1" width="80%" cellspacing="4" cellpadding="4" align="center">
17              ';
18
19          foreach ($games as $key => $array) { // Zastosowanie tablicy do wyświetlenia każdej gry.
20
21            if ($key != 0) { // Wiersza Goście   Gospodarze nie wyświetlamy.
22
23              $second_array = explode ("\t", $array); // Przekształcenie każdego wiersza
                tablicy w oddzielną tablicę.
24              $away = $second_array[0]; // Pierwszy element odpowiada zespołowi gości.
25              $home = substr ($second_array[1], 0, (strlen($second_array[1]) - 1));
                // Drugi element odpowiada zespołowi gospodarzy, ale musimy usunąć znak return.
26
27              echo "        <tr align=\"center\" valign=\"top\">
28          <td align=\"left\" valign=\"top\" nowrap=\"nowrap\"><input type=\"radio\"
            name=\"winners[$key]\" value=\"$away\">$away</td>
29          <td align=\"center\" valign=\"top\">przeciw</td>
30          <td align=\"right\" valign=\"top\" nowrap=\"nowrap\"><input type=\"radio\"
            name=\"winners[$key]\" value=\"$home\">$home</td>
31      </tr>\n";
```

3. Modyfikujemy wiersze 31. – 35. oryginalnego skryptu tak, aby wygenerowana tablica miała nazwę `$winners`.

```
echo " <tr align=\"center\" valign=\"top\">
<td align=\"left\" valign=\"top\" nowrap=\
➥"nowrap\"><input type=\"radio\" name=\
➥"winners[$key]\" value=\"$away\">$away</td>
<td align=\"center\" valign=\"top\"
➥>przeciw</td>
<td align=\"right\" valign=\"top\" nowrap=
➥\"nowrap\"><input type=\"radio\" name=
➥\"winners[$key]\" value=\"$home\"
➥>$home</td>
</tr>\n";
```

Ten krok nie jest absolutnie konieczny poza tym, że skrypt *process_winners.php*, obsługujący ten formularz, powinien otrzymać bardziej odpowiednie nazwy zmiennych (`$winners`, a nie `$picks`).

4. Zapisujemy skrypt jako *pick_winners.php*.

Zapiszemy teraz skrypt, obsługujący dane z formularza *pick_winners.php*.

5. W edytorze tekstowym utworzymy nowy dokument HTML (skrypt 1.9).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
➥1.0 Transitional//PL"
"http://www.w3.org/TR/2000/REC-xhtml1-
➥20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
     <title>Totalizator piłkarski</title>
</head>
```

Skrypt 1.8. *ciąg dalszy*

Kod

```
32          }
33     }
34
35       // Uzupełnienie formularza i tabeli.
36       echo "        <tr align=\"center\"
         valign=\"top\">
37           <td colspan=\"3\" width=\"100%\"
             align=\"center\" valign=\"top\">
             <input
              type=\"submit\" name=\"submit\"
              value=\"Wyślij!\">
38       </tr>
39   </table>
40   <input type=\"hidden\" name=\"week\"
     value=\"$week\">
41   </form>\n";
42
43   } else { // Wyświetlenie komunikatu
     o błędzie, jeżeli otwarcie pliku
     nie powiodło się.
44   echo "Nie można otworzyć pliku $file!
     Należy upewnić się, czy zmienna week ma
     ustawioną wartość.<br></br>";
45   }
46   ?>
47   </body>
48   </html>
```

Skrypt 1.9. *W tym dość długim skrypcie wykorzystano zmienne statyczne oraz rekurencję w celu zapisu do plików, odczytywania danych z plików oraz wykonywania obliczeń*

Kod

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//PL"
"http://www.w3.org/TR/2000/REC-xhtml1-20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Totalizator piłkarski</title>
</head>
<body>
<?php

// Ustawienie numeru tygodnia jako stałej.
define ("WEEK", $week);

/* ---------------------------- */
/* Zapis zwycięzców w pliku.    */
/* ---------------------------- */

function write_data ($value, $key, $fp) { // Funkcja write_data zapisuje typy w pliku.

   static $first_line;

   if (!$first_line) { // Zapisanie pierwszego wiersza-atrapy, ale tylko raz.
      fputs ($fp, "Wiersz atrapa\n");
      $first_line = TRUE;
   }

   fputs ($fp, "$value\n"); // Dodanie znaku return, aby każdy zapis znalazł się
   w osobnym wierszu.
}

$file = '2001/picks_' . WEEK . '_Winners_.txt'; // Wskazanie nazwy pliku w formacie
'picks_1_Winners_.txt'.

if ($fp = @fopen ($file, "w")) { // Otwarcie pliku do zapisu.

   array_walk ($winners, 'write_data', $fp); // Wysłanie całej tablicy do funkcji
   write_data.
   fclose ($fp); // Zamknięcie pliku.

   echo 'Zapisano zwycięzców!<br></br>';

} else { // Wyświetlenie błędu, w przypadku, gdyby otwarcie pliku było niemożliwe.
   echo "Nie można otworzyć pliku $file! Proszę upewnić się, że zmienna week ma wartość.
   <br></br>";
}


/* ---------------------------- */
/* Pobranie typów graczy.       */
/* ---------------------------- */
```

Skrypt 1.9. *ciąg dalszy*

Kod

```
46
47    function read_picks ($dp, $directory) { // Czynność tę wykonuje funkcja read_picks.
48
49       global $picks;
50
51     if ($file_name = readdir ($dp)) { // Sprawdzenie każdego pliku w katalogu.
52
53       if (substr($file_name, 0, 6) == "picks_") { // Kontynuacja, w przypadku odnalezienia
         pliku z typami.
54
55          $parse_file_name = explode ("_", $file_name); // Przekształcenie nazwy pliku
            na odpowiedni format.
56
57           if (($parse_file_name[1] == WEEK) and ($parse_file_name[2] != "Winners")) {
            // Jeżeli to nie jest plik z danymi o zwycięzcach, a numer tygodnia jest właściwy...
58
59              $username = $parse_file_name[2];
60              $the_file = $directory . $file_name;
61              $users_picks = file ($the_file);
62              $picks[$username] = $users_picks;
63           }
64          }
65
66       read_picks ($dp, $directory); // Kolejna iteracja funkcji.
67     }
68   }
69
70   $picks = array(); // Zainicjowanie głównej zmiennej.
71
72   $directory = '2001/';
73   $dp = opendir ($directory);
74   read_picks ($dp, $directory);
75   closedir ($dp);
76
77
78   /* ---------------------------------- */
79   /* Obliczenie tygodniowych wyników.  */
80   /* ---------------------------------- */
81
82   while (list ($key, $value) = each ($picks)) { // Pętla dla wyników w głównej tablicy.
83
84       $wins = 0; // Ustawienie na wartość 0 dla każdej tablicy.
85       $losses = 0; // Ustawienie na wartość 0 dla każdej tablicy.
86
87       while (list ($key2, $value2) = each ($value)) { // Pętla dla każdej tablicy użytkownika.
88
89          $value2 = substr ($value2, 0, (strlen($value2) - 1)); // Usunięcie znaku return[1].
90
91       if (($value2 == $winners[$key2]) and ($key2 != 0)) { // Jeżeli typ użytkownika jest
         właściwy, przyznajemy mu punkt, ale nie liczymy wiersza nr 1.
```

[1] Jeżeli uruchamiamy skrypt w systemie Windows, wówczas należy zapisać ten wiersz jako `$value2 = substr ($value2, 0, (strlen($value2) - 1))`, ponieważ znak RETURN składa się w systemie Windows z dwóch znaków (CR+LF) — *przyp. tłum.*

Skrypt 1.9. *ciąg dalszy*

Kod

```
92          $wins++;
93       } elseif ($key2 != 0 ) {
94          $losses++;
95       }
96    }
97
98    $results[$key] = array ('wins' => $wins, 'losses' => $losses); // Zapisanie wyników
      każdego z użytkowników w tablicy.
99    }
100
101    ksort ($results); // Uporządkowanie tablicy alfabetycznie.
102
103
104    /* ------------------------------ */
105    /* Zapisanie tygodniowych wyników. */
106    /* ------------------------------ */
107
108    function write_results ($value, $key, $fp) { // Zapisuje wyniki do pliku.
109
110       static $first_line2;
111
112       if (!$first_line2) { // Zapisanie pierwszego wiersza-atrapy, ale tylko raz.
113          fputs ($fp, "Gracz\tpoprawne\tbłędne\n"); // Dodanie znaku return tak, aby każdy
             zapis znalazł się w osobnym wierszu.
114          $first_line2 = TRUE;
115       }
116
117       $data = implode ("\t", $value);
118       fputs ($fp, "$key\t$data\n"); // Dodanie znaku return tak, aby każdy zapis
             znalazł się w osobnym wierszu.
119    }
120
121    $file2 = '2001/results_' . WEEK . '_.txt'; // Wskazanie nazwy pliku w
                                                  formacie 'results_1_.txt'.
122
123    if ($fp2 = @fopen ($file2, "w")) { // Otwarcie pliku do zapisu.
124
125       array_walk ($results, 'write_results', $fp2); // Wysłanie zawartości całej tablicy
          do funkcji write_results.
126       fclose ($fp2); // Zamknięcie pliku.
127       echo 'Wyniki zapisano.<br></br>';
128
129    } else { // Wyświetlenie komunikatu o błędzie, jeżeli otwarcie pliku nie powiodło się.
130
131       echo "Nie można otworzyć pliku z tygodniowymi wynikami, $file2.<br></br>";
132    }
133    ?>
134    </body>
135    </html>
```

6. Rozpoczniemy wpisywanie treści dokumentu HTML oraz umieścimy inicjujący znacznik PHP:

```
<body>
<? php
```

7. Zdefiniujemy stałą WEEK

```
define("WEEK", $week);
```

Ponieważ nie chcemy, aby ta wartość ulegała zmianie oraz ze względu na to, że musi ona być dostępna dla wielu funkcji, ustawiamy ją jako stałą.

8. Stworzymy funkcję zapisującą listę zwycięzców.

```
function write_data ($value, $key, $fp) {
static $first_line;
if (!$first_line) {
fputs ($fp, "Wiersz atrapa\n");
$first_line = TRUE;
}
fputs ($fp, "$value\n");
}
```

Funkcja `write_data()` zasadniczo zapisuje wiersz tekstu do pliku. Ponieważ chcemy, aby pierwszy wiersz pliku był wierszem — atrapą, to stworzyliśmy instrukcję warunkową, która zapisze ten wiersz, ale tylko wtedy, gdy pierwszy wiersz nie został jeszcze zapisany (nie chcielibyśmy, aby wiersz — atrapa był zapisywany za każdym razem, kiedy wywołujemy tę funkcję). Ponieważ zmienna `$first_line` ustawiona na wartość `TRUE` po zapisaniu pierwszego wiersza jest zadeklarowana jako statyczna za pomocą instrukcji `static`, zatem jej wartość będzie pamiętana w każdej iteracji funkcji. Z tego powodu, kiedy funkcja zostanie wywołana po raz drugi, zmienna `$first_line` będzie pamiętana jako `TRUE` i instrukcja warunkowa nie wykona się.

9. Wybieramy plik do przetwarzania.

```
$file = '2001/picks_' . WEEK .
➥'_Winners_.txt';
```

Lista zwycięskich zespołów dla każdego tygodnia będzie zapisywana w pliku *pick_1_Winners_.txt* (jeżeli jest to tydzień nr 1). Pliki będą umieszczone w katalogu *2001*.

10. Zapiszemy instrukcję warunkową, która otwiera plik i zapisuje w nim tablicę zwycięzców.

```
if ($fp = @fopen ($file, "w")) {
array_walk ($winners, 'write_data', $fp);
fclose ($fp); // Zamknięcie pliku.
echo 'Zapisano zwycięzców!<br></br>';
} else {
echo "Nie można otworzyć pliku $file!
➥Proszę upewnić się, że zmienna week
ma wartość.<br></br>";
}
```

Kod ten jest bardzo podobny do zapisanego w skrypcie *store_picks.php*.

11. Utworzymy funkcję, która pobiera wszystkie istniejące typy użytkowników.

```
function read_picks ($dp, $directory) {
global $picks;
if ($file_name = readdir ($dp)) {
➥// Sprawdzenie każdego pliku w katalogu.
if (substr($file_name, 0, 6) == "picks_") {
➥$parse_file_name = explode ("_",
➥$file_name);
if (($parse_file_name[1] == WEEK) and
➥($parse_file_name[2] != "Winners"))
$username = $parse_file_name[2];
$the_file = $directory . $file_name;
$users_picks = file ($the_file);
$picks[$username] = $users_picks;
}
}
read_picks ($dp, $directory);
}
```

Funkcja ta wykonuje się w pętli (dzięki przedostatniemu wierszowi powodującemu rekurencję) dla wszystkich plików w katalogu, aż do odczytania wszystkich typów. Wtedy to warunek instrukcji warunkowej przyjmie wartość `false` i funkcja nie wywoła samej siebie. W przypadku, kiedy plik spełnia kryteria pliku zawierającego typy użytkownika (tzn. `substr($file_name,0,6)="picks_"`), wówczas dane zostaną odczytane i zapisane do globalnej tablicy `$picks` z wykorzystaniem nazwy użytkownika jako indeksu.

12. Utworzymy główną tablicę, rozpoznamy i otworzymy katalog, a następnie wywołamy funkcję `read_picks()`.

```
$picks = array();
$directory = '2001/';
$dp = opendir ($directory);
read_picks ($dp, $directory);
closedir ($dp);
```

13. Zapiszemy pętlę, w której sprawdzimy trafność typów poszczególnych graczy.

```
while (list ($key, $value) = each
➥($picks)) {
$wins = 0;
$losses = 0;
while (list ($key2, $value2) = each
➥($value)) {
$value2 = substr ($value2, 0,
➥(strlen($value2) - 1));
if (($value2 == $winners[$key2]) and
➥($key2 != 0)) {
$wins++;
} elseif ($key2 != 0 ) {
$losses++;
}
}
$results[$key] = array ('wins' => $wins,
➥'losses' => $losses);
}
```

Mamy tu dwie pętle, pierwsza działa dla wielowymiarowej tablicy `$picks`, składającej się z wyników wytypowanych przez użytkowników w danym tygodniu. Kolejna pętla działa w tablicy typów każdego z graczy i sprawdza, czy gracz prawidłowo wytypował zwycięzcę. Jeżeli tak się stało, zmienna `$wins` jest zwiększana o 1, jeżeli nie — zwiększana jest zmienna `$losses`. Ostatecznie zapisy są wprowadzane do wielowymiarowej tablicy `$results`, zawierającej liczbę trafnych i błędnych typów poszczególnych graczy.

14. Uporządkujemy alfabetycznie wyniki według nazwy użytkownika.

```
ksort($results);
```

Tablica `$results` w naszym przykładzie składa się teraz z czterech elementów pod indeksami *Brian*, *Juan*, *Larry* oraz *Michael* (moi czterej użytkownicy testowi). Każdy element wskazuje na inną tablicę, której indeksy to `wins` oraz `losses`. Funkcja `ksort()` posortuje pierwszy zestaw indeksów — nazwy użytkowników w porządku alfabetycznym. Sortowanie nie będzie miało wpływu na pozostałe dane.

15. Utworzymy funkcję, która zapisze wyniki dla określonego tygodnia.

```
function write_results ($value, $key, $fp) {
static $first_line2;
if (!$first_line2) {
fputs ($fp, "Gracz\tpoprawne\tbłędne\n");.
$first_line2 = TRUE;
}
$data = implode ("\t", $value);
fputs ($fp, "$key\t$data\n");.
}
```

Funkcja ta przypomina funkcję `write_data()`, z tą różnicą, że wykorzystuje inny nagłówek. Ponownie wykorzystujemy zmienną statyczną, aby nagłówek był zapisywany tylko raz.

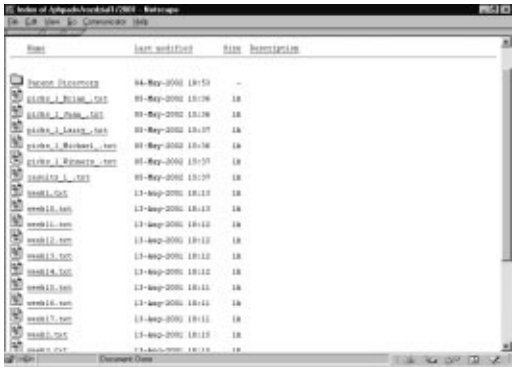
16. Wskazujemy plik wyników i tworzymy instrukcję warunkową, która spowoduje wysłanie danych do tego pliku.

```
$file2 = '2001/results_' . WEEK . '_.txt';
if ($fp2 = @fopen ($file2, "w")) {
array_walk ($results, 'write_results',
➥$fp2);
fclose ($fp2);
echo 'Wyniki zapisano.<br></br>';
} else {
echo "Nie można otworzyć pliku z
➥tygodniowymi wynikami, $file2.<br></br>";
}
```

Jest to kolejna kopia kodu wywołującego funkcję `write_data()`, który wykorzystaliśmy we wcześniejszej części skryptu. Kiedy funkcja `array_walk()` wyśle tablicę `$array` do funkcji `write_results()`, wówczas wszystkie poprawnie i błędnie wytypowane wyniki zostaną zapisane w pliku o nazwie *results_1_.txt* (dla tygodnia numer 1).

Rysunek 1.11. *Strona pick_winners.php wygląda podobnie do strony make_picks_form.php (rysunek 1.7) z tą różnicą, że nie ma tu możliwości wprowadzenia nazwy użytkownika*

Rysunek 1.12. *Jeżeli skrypt process_winners.php wykona się poprawnie, zobaczymy te dwa komunikaty. Wynik działania skryptu możemy sprawdzić, zaglądając do katalogu 2001 (rysunek 1.13)*



Rysunek 1.13. *Katalog zawiera teraz plik pick_ Winners_txt, który zawiera zwycięskie drużyny, a także plik results_1_.txt, w którym znajdują się informacje o trafności typów poszczególnych graczy*

Skrypt 1.10. *Skrypt process_winners.php generuje taki oto plik z wynikami dla każdego tygodnia, dla którego uruchomimy skrypt pick_winners.php. Plik z wynikami zawiera nazwy wszystkich użytkowników oraz liczby trafnych i błędnych typów dla każdego z nich*

Kod

```
1  Gracz        poprawne   błędne
2  Brian        8          7
3  Juan         11          4
4  Larry        10          5
5  Michael      11          4
```

17. Kończymy kod PHP oraz HTML

```
?>
</body>
</html>
```

18. Zapisujemy plik jako *process_winners.php*, kopiujemy go do tego samego katalogu, co skrypt *pick_winners.php* (skrypt 1.8), a następnie uruchamiamy ten drugi skrypt w przeglądarce WWW (rysunki 1.11 oraz 1.12).

19. Możemy też obejrzeć zawartość katalogu *2001* (rysunek 1.13), a także zawartość pliku *results_1_.txt* (skrypt 1.10).

Funkcje a odwołania

Domyślnie funkcje otrzymują argumenty według zasady *wywołanie przez wartość*. Oznacza to, że funkcja otrzymuje wartość zmiennej, a nie samą zmienną. Aby uaktualnić wartość zmiennej wewnątrz funkcji, należy zastosować instrukcję `global` albo przekazać zmienną przez odwołanie.

Oto przykład zwyczajnego działania funkcji:

```
function change_value($variable) {
   $variable++;
   echo $variable;
}
$variable = 1;
change_value($variable);
```

Instrukcja echo wyświetli wartość `2`, ale wartość zmiennej `$variable` w dalszym ciągu wynosi `1` (pamiętajmy, że zmienna `$variable` wewnątrz funkcji, oprócz wspólnej nazwy, nie jest tym samym, czym poza nią). Jeżeli byśmy jednak przekazali zmienną `$variable` przez odwołanie, wówczas wartość zmiennej `$variable` poza funkcją także się zmieni.

Aby przekazać zmienną przez odwołanie, a nie przez wartość, nazwę zmiennej należy poprzedzić znakiem `&`. Możemy to zrobić w wierszu, w którym wywołujemy funkcję — w takim przypadku dotyczyć to będzie określonej zmiennej — lub w definicji funkcji, wówczas będzie to dotyczyć każdej zmiennej przekazywanej do funkcji.

```
function change_value($variable) {
   $variable++;
   echo $variable;
}
$variable = 1;
change_value(&$variable);
```

Teraz funkcja wyświetla wartość `2`. Ten sam wynik można osiągnąć za pomocą następującej funkcji:

```
function change_value(&$variable) {
   $variable++;
   echo $variable;
}
$variable = 1;
change_value($variable);
```

Różnica w tym drugim przykładzie polega na tym, że każda zmienna wysłana do funkcji `change_value()` automatycznie przekazywana jest przez adres.

Zaletą wykorzystywania przekazywania zmiennych przez adres jest fakt, że w takim przypadku zaoszczędza nam to kłopotów związanych z zastosowaniem zmiennych globalnych oraz zwracaniem wartości przez funkcje. Przekonamy się o tym, kiedy utworzymy ostatni skrypt w naszej aplikacji totalizatora online.

Aliasy

Nowością w PHP4 jest możliwość przekazywania parametrów przez adres niezależnie od funkcji. Składnia instrukcji służących do wykonywania tego typu czynności jest następująca:

```
$variable1="Chicago Cubs";
$variable2=$variable1; // zmienna $variable2 jest kopią zmiennej $variable1
$variable3=& $variable1; // zmienna $variable3 jest odwołaniem do zmiennej $variable1
$variable2="National League Leaders"; // wartość zmiennej $variable2 to "National League
➥Leaders" wartość zmiennej $variable1 to w dalszym ciągu "Chicago Cubs"
$variable3.= ", National League Leaders";// obie zmienne $variable3 oraz $variable1 mają
➥wartość "Chicago Cubs, National League Leaders"
```

Wykorzystując tego typu odwołania, tworzymy alias. W powyższym przykładzie zmienna `$variable3` jest aliasem zmiennej `$variable1` (wiersz, za pomocą którego utworzyliśmy ten alias, moglibyśmy także zapisać jako `$variable3=&$variable1;`). Dowolne zmiany, dotyczące jednej ze zmiennych, będą automatycznie uwzględnione dla drugiej.

Zastosowanie odwołań do zmiennych zamiast kopii może wpłynąć na poprawę wydajności aplikacji, szczególnie gdy mamy do czynienia z tablicami dużych rozmiarów oraz obiektami. Podczas gdy każda kopia zmiennej zajmuje tyle samo miejsca w pamięci co oryginał, to alias zajmuje go znacznie mniej.

Dobrą analogią do tej sytuacji jest skrót do dokumentu w naszym komputerze. Powiedzmy, że mamy dokument Venus.txt w folderze *Astronomy* i stworzyliśmy alias lub skrót do tego dokumentu (na przykład na pulpicie) o nazwie *Evening_Star*. Zarówno skrót, jak i sam dokument odwołują się do tej samej wartości (zawartości otwartego dokumentu) i dowolne zmiany tej zawartości będą uwzględnione niezależnie od tego, czy otworzymy dokument za pośrednictwem skrótu *Evening_Star*, czy też za pomocą oryginalnego pliku *Venus.txt*. Różnica jest jednak taka, że plik *Venus.txt* ma 40 KB, podczas gdy skrót jedynie 2 KB.

Aby wykorzystać odwołania

1. W edytorze tekstowym utworzymy nowy dokument HTML (skrypt 1.11):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
➥1.0 Transitional//PL"
"http://www.w3.org/TR/2000/REC-xhtml1-
➥20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<title>Totalizator piłkarski</title>
</head>
<body>
```

2. Rozpoczniemy wpisywanie treści dokumentu HTML oraz umieścimy inicjujący znacznik PHP:

```
<body>
<?php
```

3. Zapiszemy funkcję, która będzie sumować trafne i błędne typy.

```
function do_the_math ($value, $key, $total) {
if ($key != 0) {
$line = explode ("\t", $value);
$username = $line[0];
$total["$username"]["wins"] += $line[1];
$total["$username"]["losses"] += $line[2];
}
}
```

Funkcja ta otrzymuje wartość z tablicy, jej indeks oraz tablicę `$total`. Wartość, która jest przekazywana w formacie `NAZWA_UŻYTKOWNIKA [TAB] TRAFNE [TAB] BŁĘDNE [RETURN]`, zostanie zdekomponowana pod warunkiem, że nie jest to pierwszy element tablicy (`$key!=0`) — wiersz — atrapa. Następnie sumowane są trafne i błędne typy danego użytkownika.

4. Utworzymy funkcję, która odczytuje z plików wyniki ze wszystkich tygodni:

```
function read_results ($dp, $directory,
➥&$total) {
if ($file_name = readdir ($dp)) {
if (substr($file_name, 0, 8) ==
➥"results_") {
$the_file = $directory . $file_name;
array_walk (file ($the_file),
➥'do_the_math', &$total);
}
read_results ($dp, $directory, $total);
}
}
```

Funkcja `read_results()` otrzymuje trzy argumenty — wskaźnik na katalog (utworzony w momencie otwierania katalogu), nazwę katalogu oraz tablicę `$total`. Ta ostatnia zmienna jest zawsze przekazywana przez odwołanie, a nie przez wartość, tak więc wszystkie dokonane zmiany dla tej zmiennej wewnątrz funkcji będą dotyczyły zmiennej poza funkcją.

Funkcja `read_results()` wykonuje się w pętli dla każdego pliku w katalogu (zauważmy, że w funkcji zastosowano rekurencję — funkcja wywołuje samą siebie). W przypadku rozpoznania pliku z wynikami, który charakteryzuje się nazwą postaci *results_WEEK_.txt*, plik ten będzie odczytany do tablicy, a tablica zostanie przekazana przez odwołanie do funkcji `do_the_math()`. Dzięki temu dowolne zmiany w tablicy `$total` poza funkcją będą odzwierciedlone w obrębie funkcji `read_results()`.

Skrypt 1.11. *Ostatni skrypt w tej aplikacji WWW odczytuje wszystkie pliki z wynikami. Wykorzystano w nim odwołania w celu obliczenia podsumowań dla sezonu — do aktualnej daty*

Kod

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//PL"
"http://www.w3.org/TR/2000/REC-xhtml1-20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <title>Totalizator piłkarski</title>
</head>
<body>
<?php

/* ------------------------------- */
/* Obliczenie wyników sezonu.      */
/* ------------------------------- */

function do_the_math ($value, $key, $total) { // Funkcja sumuje trafne i błędne typy
użytkowników.

   if ($key != 0) { // Ignorujemy pierwszy wiersz - atrapę.

      $line = explode ("\t", $value); // Wydzielamy nazwę użytkownika oraz trafne
      i błędne typy.
      $username = $line[0];
      $total["$username"]["wins"] += $line[1];
      $total["$username"]["losses"] += $line[2];
   }
}

function read_results ($dp, $directory, &$total) {

   if ($file_name = readdir ($dp)) { // Przeszukiwanie katalogu.
      if (substr($file_name, 0, 8) == "results_") { // Jeżeli jest to plik z wynikami ...
         $the_file = $directory . $file_name;
         array_walk (file ($the_file), 'do_the_math', &$total); // Wykonanie obliczeń
         dla danych.
      }
      read_results ($dp, $directory, $total); // Powtarzanie operacji dla kolejnych plików
      w katalogu.
   }
}

$total = array(); // Zainicjowanie głównej zmiennej.

$directory = '2001/';
$dp = opendir ($directory);
read_results ($dp, $directory, $total); // Uruchomienie procesu.
closedir ($dp);
echo "<pre>\n";
print_r ($total); // Szybkie wyświetlenie wyników.
echo "</pre>\n";
?>
</body>
</html>
```

5. Tworzymy główną zmienną, otwieramy katalog i rozpoczynamy proces obliczeń.

```
$total = array();
$directory = '2001/';
$dp = opendir ($directory);
read_results ($dp, $directory, $total);
closedir ($dp);
```

6. Wyświetlamy obliczenia w przeglądarce WWW.

```
echo "<pre>\n";
print_r ($total);
echo "</pre>\n";
```

Funkcja `print_r()` pobiera zmienną jako argument i wyświetla jej zawartość oraz strukturę. Dzięki temu funkcja ta jest bardzo użytecznym sposobem szybkiej analizy skomplikowanych tablic i obiektów. Moglibyśmy też pobrać tę tablicę i w celu wyświetlenia wyników utworzyć formalną tabelę. Funkcja `print_r()` jest nowością w PHP4. Użytkownicy wcześniejszej wersji powinni zamiast niej skorzystać z funkcji `var_dump()`.

7. Zamykamy kod PHP oraz dokument HTML.

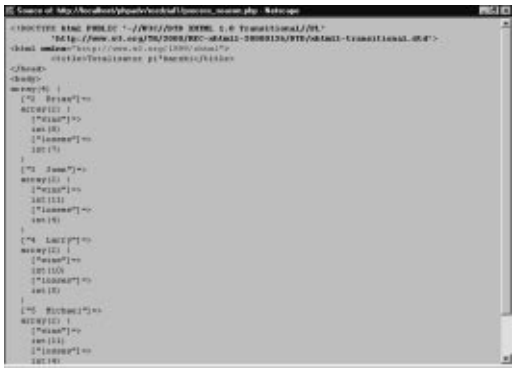
```
?>
</body>
</html>
```

8. Zapisujemy ten plik pod nazwą *process_season.php*, kopiujemy go na serwer i uruchamiamy w przeglądarce WWW (rysunek 1.14).

Rysunek 1.14. *Jak widać, skrypt process_season.php po prostu wyświetla wyniki. Po uzyskaniu tej informacji można ją bardzo łatwo przesłać pocztą elektroniczną*

Rysunek 1.15. *Funkcja var_dump() wyświetla te same informacje, co funkcja print_r() (rysunek 1.14), ale zawiera nieco więcej danych*

Rysunek 1.16. *Próba wykonania funkcji print_r lub funkcji var_dump() bez znaczników <pre></pre> powoduje wyświetlenie brzydkiego, nieczytelnego wyniku*



Rysunek 1.17. *Przeglądanie źródła wyniku pokazanego na rysunku 1.16 powoduje poprawę jakości wyświetlania tablicy, która została wyświetlona przez funkcję var_dump(). W zasadzie odpowiada to rysunkowi 1.15, gdzie użyto znaczników <pre></pre>*

Wskazówki

- Aby porównać funkcję `print_r()` z funkcją `var_dump()`, należy zmienić ostatni wiersz w skrypcie *process_season.php* (skrypt 1.11) na `var_dump($total)`, a następnie uruchomić ten skrypt w przeglądarce (rysunek 1.15).
- Zastosowanie znaczników HTML `<pre></pre>` powoduje wyświetlanie tablicy w przeglądarce w formacie źródła. Na rysunku 1.16 pokazano, jak wyglądałby wynik bez tych znaczników, chociaż źródło (rysunek 1.17) jest czytelne.
- Więcej informacji na temat odwołań można znaleźć pod adresem *www.php.net/manual/en/language.references.php*.