

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

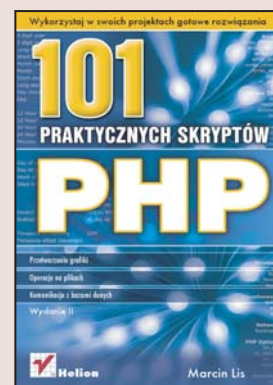
ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

PHP. 101 praktycznych skryptów. Wydanie II

Autor: Marcin Lis
ISBN: 83-246-0796-X
Format: B5, stron: 296



Wykorzystaj w swoich projektach gotowe rozwiązania

- Przetwarzanie grafiki
- Operacje na plikach
- Komunikacja z bazami danych

PHP to jeden z najpopularniejszych języków wykorzystywanych do tworzenia dynamicznych witryn WWW. Od początku swojej obecności na rynku zyskał ogromne uznanie programistów. Jest dostępny nieodpłatnie i ma ogromne możliwości. Jego najnowsza wersja otworzyła przed twórcami aplikacji nowe horyzonty, oferując im wszystkie korzyści wynikające z programowania obiektowego. PHP jest dostępny dla większości popularnych systemów operacyjnych z rodziny Unix/Linux, Windows, Mac OS oraz Risc OS. Współpracuje też z wieloma serwerami HTTP, dzięki czemu jest niezwykle uniwersalny.

W książce „PHP. 101 praktycznych skryptów. Wydanie II” znajdziesz gotowe skrypty, dzięki którym zdecydowanie przyspieszysz swoją pracę nad aplikacjami i serwisami internetowymi. To wydanie zostało tak zaktualizowane w stosunku do poprzedniego, że skrypty działają prawidłowo w najnowszej wersji języka PHP. Autor zapewnił także zgodność generowanych przez nie dokumentów HTML z obowiązującymi standardami tego języka. Każdy ze skryptów możesz po prostu wkleić do swojego kodu bądź zmodyfikować, aby dokładnie odpowiadał Twoim potrzebom. Korzystając z przykładów znajdujących się w tej książce, możesz dodać do swoich projektów funkcje obsługujące system plików, przetwarzające grafikę, zabezpieczające witryny i aplikacje przed nieautoryzowanym dostępem.

- Operacje sieciowe
- Praca z systemem plików
- Liczniki odwiedzin i księgi gości
- Przetwarzanie grafiki
- Autoryzacja użytkowników
- Komunikacja z bazami danych
- Przewodnik po najważniejszych elementach języka PHP

Skorzystaj ze sprawdzonych rozwiązań



Spis treści

Wstęp	9
Rozdział 1. Globalna sieć	11
Skrypt 1. Uzyskanie adresu IP	11
Skrypt 2. Odczytanie rekordów MX	13
Skrypt 3. Nawiązanie połączenia TCP	14
Skrypt 4. Wysyłanie poczty	17
Skrypt 5. Pobranie pliku z serwera ftp	19
Skrypt 6. Wysłanie pliku do serwera ftp	20
Skrypt 7. Wysłanie pliku do serwera (upload pliku)	21
Skrypt 8. Przekierowanie z użyciem znacznika <meta>	24
Skrypt 9. Przekierowanie z użyciem nagłówków HTTP	25
Skrypt 10. Przekierowanie ze względu na adres IP	25
Skrypt 11. Przekierowanie na losową witrynę	26
Skrypt 12. Zablokowanie wybranych adresów IP	27
Skrypt 13. Walidacja adresu e-mail	28
Skrypt 14. Wysłanie pliku do przeglądarki	28
Skrypt 15. Pobieranie plików z listy I	30
Skrypt 16. Pobieranie plików z listy II	32
Skrypt 17. Generowanie listy plików do pobrania	35
Rozdział 2. System plików	37
Skrypt 18. Wykonanie polecenia zewnętrznego	37
Skrypt 19. Wyświetlenie listy plików	38
Skrypt 20. Nawigacja po katalogach serwera	39
Skrypt 21. Usunięcie zawartości katalogu	42
Skrypt 22. Rozmiar katalogu	43
Skrypt 23. Kopiowanie zawartości katalogu	43
Skrypt 24. Wyświetlenie plików określonego typu jako odnośników	44
Rozdział 3. Liczniki, księgi gości itp.	47
Skrypt 25. Prosty licznik tekstowy	47
Skrypt 26. Licznik przechowujący datę początkową	48
Skrypt 27. Licznik graficzny	50
Skrypt 28. Licznik filtrujący adresy IP	51
Skrypt 29. Licznik filtrujący adresy IP II	53
Skrypt 30. Licznik uwzględniający tylko jedno odwołanie z danego IP	54

Skrypt 31. Głosowanie (ankieta)	55
Skrypt 32. Księga gości	61
Skrypt 33. Księga gości z nawigacją	65
Skrypt 34. Porada dnia	69
Skrypt 35. Porada dnia z hasłami w pliku	70
Skrypt 36. Inna strona dla znanego użytkownika	71
Rozdział 4. Grafika i obrazy	73
Skrypt 37. Najprostsza galeria	73
Skrypt 38. Zautomatyzowana galeria	75
Skrypt 39. Galeria z podpisami obrazów	79
Skrypt 40. Galeria z miniaturami obrazów	82
Skrypt 41. Zmiana rozdzielczości obrazu	86
Skrypt 42. Zmiana rozdzielczości obrazu z zachowaniem proporcji	87
Skrypt 43. Zmiana rozdzielczości z rozpoznaniem typu pliku	88
Skrypt 44. Informacja o wykorzystywanej bibliotece graficznej	91
Skrypt 45. Zmiana rozdzielczości obrazów z wybranego katalogu	92
Skrypt 46. Przeskalowanie serii obrazów	93
Skrypt 47. Obracanie obrazu	94
Skrypt 48. Obracanie serii obrazów	95
Skrypt 49. Konwersja obrazu do wybranego formatu	96
Skrypt 50. Konwersja obrazów z wybranego katalogu	99
Skrypt 51. Tekst jako obrazek	101
Skrypt 52. Nałożenie tekstu na obraz	103
Rozdział 5. Użytkownicy i hasła	105
Skrypt 53. Hasło dostępu do strony	105
Skrypt 54. Logowanie użytkowników	107
Skrypt 55. Logowanie z kodowaniem haseł	109
Skrypt 56. Zarządzanie hasłami	110
Skrypt 57. Hasło dostępu z użyciem sesji	114
Skrypt 58. Logowanie użytkowników z wykorzystaniem sesji	118
Skrypt 59. Inna strona dla każdego użytkownika	120
Skrypt 60. Generowanie losowego hasła	122
Skrypt 61. Automatyczne logowanie	123
Rozdział 6. Data i czas	127
Skrypt 62. Bieżąca data i czas	127
Skrypt 63. Ile dni do...?	128
Skrypt 64. Różnica między dwoma datami	129
Skrypt 65. Data ostatniej modyfikacji strony	132
Skrypt 66. Strona zależna od pory dnia	133
Skrypt 67. Strona zależna od dnia tygodnia	134
Skrypt 68. Wyświetlenie nazwy dnia tygodnia	134
Skrypt 69. Rysunek zależny od dnia tygodnia	135
Skrypt 70. Czas generowania strony	136
Skrypt 71. Kalendarz	137
Rozdział 7. Bazy danych	141
Skrypt 72. Licznik wykorzystujący bazę danych	141
Skrypt 73. Ankieta	143
Skrypt 74. Księga gości	146
Skrypt 75. Logowanie	150
Skrypt 76. Autoryzacja z wykorzystaniem sesji i kodowaniem danych	152
Skrypt 77. Zarządzanie hasłami	154

Skrypt 78. Zapamiętanie danych użytkownika	157
Skrypt 79. Automatyczne logowanie	163
Skrypt 80. Liczba osób obecnych na stronie	165
Skrypt 81. Liczba osób przeglądających stronę z różnych adresów IP	167
Skrypt 82. Statystyka strony (lista odwiedzin)	169
Skrypt 83. Liczba przeglądających stronę (z wykorzystaniem statystyki strony)	171
Skrypt 84. Pobieranie plików	172
Skrypt 85. Ranking plików	176
Skrypt 86. Automatyczne generowanie nazw plików	180
Skrypt 87. Lista odnośników	181
Skrypt 88. Zliczanie odwiedzeń z każdego adresu IP	183
Skrypt 89. Porada dnia	186
Skrypt 90. Statystyka przeglądarek	187
Rozdział 8. Rozmaitości	191
Skrypt 91. Lista odwiedzin	191
Skrypt 92. Rozpoznanie typu przeglądarki	192
Skrypt 93. Rozpoznanie typu systemu operacyjnego	194
Skrypt 94. Ocenzowanie tekstu	194
Skrypt 95. Ocenzowanie tekstu z wykorzystaniem zewnętrznego słownika	195
Skrypt 96. Losowy baner	196
Skrypt 97. Losowy baner z wybranego katalogu	197
Skrypt 98. Banery wyświetlane w określonej kolejności	198
Skrypt 99. Ochrona przed spamem	199
Skrypt 100. Adres e-mail w postaci obrazu	200
Skrypt 101. Zabezpieczenie witryny przed kopiowaniem	202
Dodatek A Krótki przewodnik po PHP	205
Krótką historia PHP	205
Instalacja	205
PHP i HTML	206
Znaczniki PHP	206
Pierwszy skrypt	207
Łączenie skryptów	208
Komentarze	209
Zmienne w PHP	210
Typy danych	210
Konwersje typów	217
Zmienne globalne (superglobalne)	221
Operatory	222
Operatory arytmetyczne	222
Operatory logiczne	222
Operatory bitowe	223
Operatory porównywania (relacyjne)	223
Operatory przypisania	224
Operatory inkrementacji/dekrementacji	225
Pozostałe operatory	226
Priorytety operatorów	227
Instrukcje	227
Instrukcje warunkowe	227
Pętle	229
Składnia alternatywna	231
Funkcje	232
Argumenty funkcji	232

Klasy i obiekty	234
Dziedziczenie	235
Konstruktory	236
Operator zakresu	237
Współpraca z przeglądarką	238
Metoda GET	239
Metoda POST	241
Współpraca z systemem	242
Odczyt i zapis plików	242
Data i czas	245
Bazy danych	253
Obsługa baz danych	253
Łączenie z bazą danych	253
Zapytania	254
Pobieranie danych	255
Pobieranie wyników zapytania	257

Dodatek B Wybrane funkcje dostępne w PHP 259

Funkcje systemu plików	259
Funkcja basename	259
Funkcja chgrp	260
Funkcja chmod	260
Funkcja chown	260
Funkcja clearstatcache	260
Funkcja copy	261
Funkcja dirname	261
Funkcja disk_free_space	261
Funkcja disk_total_space	261
Funkcja fclose	262
Funkcja feof	262
Funkcja fflush	262
Funkcja fgets	262
Funkcja fgetsv	263
Funkcja fgets	263
Funkcja fgets	263
Funkcja file_exists	263
Funkcja file_get_contents	264
Funkcja file_put_contents	264
Funkcja file	264
Funkcja fileatime	264
Funkcja filectime	265
Funkcja filegroup	265
Funkcja fileinode	265
Funkcja filemtime	265
Funkcja fileowner	265
Funkcja fileperms	266
Funkcja filesize	266
Funkcja filetype	266
Funkcja flock	266
Funkcja fopen	267
Funkcja fpassthru	268
Funkcja fputs	268
Funkcja fread	268
Funkcja fscanf	268

Funkcja fseek	269
Funkcja fstat	269
Funkcja ftell	269
Funkcja ftruncate	269
Funkcja fwrite	270
Funkcja glob	270
Funkcja is_dir	271
Funkcja is_executable	271
Funkcja is_file	271
Funkcja is_link	271
Funkcja is_readable	271
Funkcja is_uploaded_file	272
Funkcja is_writable	272
Funkcja is_writeable	272
Funkcja link	272
Funkcja linkinfo	272
Funkcja lstat	273
Funkcja mkdir	273
Funkcja move_uploaded_file	273
Funkcja parse_ini_file	273
Funkcja pathinfo	274
Funkcja pclose	274
Funkcja popen	274
Funkcja readfile	274
Funkcja readlink	275
Funkcja realpath	275
Funkcja rename	275
Funkcja rewind	275
Funkcja rmdir	275
Funkcja set_file_buffer	276
Funkcja stat	276
Funkcja symlink	276
Funkcja tempnam	276
Funkcja tmpfile	277
Funkcja touch	277
Funkcja unlink	277
Funkcje sieciowe	277
Funkcja checkdnsrr	277
Funkcja closelog	278
Funkcja debugger_off	278
Funkcja debugger_on	278
Funkcja define_syslog_variables	278
Funkcja dns_check_record	278
Funkcja dns_get_mx	279
Funkcja dns_get_record	279
Funkcja fsockopen	279
Funkcja gethostbyaddr	280
Funkcja gethostbyname	280
Funkcja gethostbyname_l	280
Funkcja getmxrr	280
Funkcja getprotobyname	280
Funkcja getprotobynumber	281
Funkcja getservbyname	281
Funkcja getservbyport	281

Funkcja ip2long	281
Funkcja long2ip	281
Funkcja openlog	282
Funkcja pfsockopen	282
Funkcja socket_get_status	282
Funkcja socket_set_blocking	283
Funkcja socket_set_timeout	283
Funkcja syslog	283
Skorowidz	285

Rozdział 3.

Liczniki, księgi gości itp.

Skrypt 25. Prosty licznik tekstowy

Bardzo popularne na witrynach internetowych są różnego rodzaju liczniki odwiedzin. Do ich tworzenia doskonale nadają się skrypty PHP, a najprostszy licznik tekstowy można zawrzeć w dosłownie kilku wierszach kodu.

Dane naszego licznika, czyli liczbę odwiedzin, będziemy przechowywać w pliku *counter.txt* w postaci tekstowej. Dzięki temu łatwo będziemy mogli wprowadzić wartość początkową, korzystając z dowolnego edytora tekstu. Plik ten otworzymy przy pomocy funkcji `fopen` w trybie `r+`, czyli do zapisu i odczytu. Po otwarciu wczytamy jedną linię tekstu i przypiszemy ją do zmiennej `$count`. Wartość tej zmiennej zwiększymy o jeden i wyświetlimy ją na ekranie. Po wykonaniu tych czynności trzeba będzie przesunąć wskaźnik położenia w pliku na jego początek i ponownie zapisać w nim wartość zmiennej `$count`. Na koniec plik należy zamknąć przy pomocy funkcji `fclose`. Cała konstrukcja prezentuje się następująco:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Licznik tekstowy</title>
</head>
<body>
<div style="text-align:center">
<h2>Tutaj znajduje się treść strony.</h2>
<br /><br />
</div>
<div style="text-align:center">
Ta strona została odwiedzona
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
autor: Marcin Lis http://marcinlis.com*/
```



```

function getHits()
{
    if(($fp = @fopen("counter.txt", "r+")) === false)
        return false;
    $count = fgets($fp);
    $count = $count + 1;
    fseek($fp, 0);
    fputs($fp, $count);
    fclose($fp);
    return $count;
}
echo(getHits());
?>
    razy od 18 kwietnia 2008 r.
</div>
</body>
</html>

```

Uruchomienie tego skryptu spowoduje pojawianie się strony jak na rysunku 3.1. Jak widać, realizacja najprostszego licznika nie powinna sprawić kłopotu nawet początkującym programistom. Należy jedynie pamiętać, aby ustawić uprawnienia pliku *counter.txt*, tak aby PHP miał możliwość jego odczytu i zapisu.

Rysunek 3.1.
Działanie licznika
tekstowego



Skrypt 26. Licznik przechowujący datę początkową

Część liczników zawiera informację o dacie, od której rozpoczęło się zliczanie liczby odwiedzin. W skrypcie 25. była ona umieszczana w treści strony, jednak można ją również umieścić w pliku przechowującym liczbę odwiedzin. Dla uproszczenia dzień, miesiąc i rok można zapisać w kolejnych liniach, dzięki czemu nie będzie potrzebna żadna analiza danych po wczytaniu, ich edycja będzie wyjątkowo prosta, a postać daty prezentowanej na witrynie zależna jedynie od sposobu jej przetwarzania w kodzie PHP.

Zasada działania skryptu jest taka sama jak w przypadku skryptu 25. Nowe elementy to wczytanie trzech linii zawierających elementy składowe daty. Wykorzystane zostaną w tym celu trzy dodatkowe wywołania funkcji *fgets*, w postaci:

```
$dzien = trim(fgets($fp));
$miesiac = trim(fgets($fp));
$rok = trim(fgets($fp));
```

Zastosowana w wywołaniach funkcja `trim` służy do usunięcia znaków końca linii, które pozostają w ciągu wynikowym zwracanym przez `fgets`. Odczytane w ten sposób dane zostaną wyświetlone na stronie w formacie: *dzień-miesiąc-rok*.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Licznik tekstowy</title>
</head>
<body>
<div style="text-align:center">
<h2>Tutaj znajduje się treść strony.</h2>
<br /><br />
</div>
<div style="text-align:center">
Ta strona została odwiedzona
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
autor: Marcin Lis http://marcinlis.com*/

function printHits()
{
    $end_line = "\r\n";
    if(($fp = fopen("counter.txt", "r")) === false)
        return false;
    $count = fgets($fp);
    $dzien = trim(fgets($fp));
    $miesiac = trim(fgets($fp));
    $rok = trim(fgets($fp));
    $count = $count + 1;
    fclose($fp);
    if(!($fp = fopen("counter.txt", "wb")) === false){
        fputs($fp, $count.$end_line);
        fputs($fp, $dzien.$end_line);
        fputs($fp, $miesiac.$end_line);
        fputs($fp, $rok.$end_line);
        fclose($fp);
    }
    echo("$count razy od $dzien-$miesiac-$rok r.");
}
printHits();
?>
</div>
</body>
</html>
```

Skrypt 27. Licznik graficzny

Skrypty 25. i 26. pokazywały, w jaki sposób umieścić na stronie WWW liczniki tekstowe pokazujące liczbę odwiedzin. Atrakcyjnym rozwiązaniem może być również wykorzystanie licznika prezentującego dane w postaci graficznej. W tym celu należy przygotować dziesięć obrazów przedstawiających kolejne cyfry od 0 do 9. Do ich wykonania można wykorzystać dowolny edytor graficzny lub poszukać gotowych grafik w internecie. Obrazy należy zapisać w plikach, których nazwy odpowiadają poszczególnym cyfrom, czyli *0.jpg*, *1.jpg* itd.

Podstawowa część skryptu będzie taka sama jak w poprzednich dwóch przykładach. Dane dotyczące liczby odwiedzin oraz daty uruchomienia licznika zapisane są w pliku *counter.txt* w postaci tekstowej. Po wczytaniu tych danych trzeba jednak będzie dokonać ich analizy tak, aby można było wczytać odpowiednie pliki z grafiką.

Skorzystamy z możliwości poruszania się po łańcuchu znaków, tak jak po tablicy. Wystarczy zatem napisać pętlę, która będzie pobierała kolejne znaki z ciągu oznaczającego liczbę odwiedzin i wygeneruje odpowiednie znaczniki ``. Pętla taka będzie miała postać:

```
for($i = 0; $i < $strLength; $i++){
    $temp = $count[$i]. '.jpg';
    $imgStr .= "<img src=\"\$temp\" alt=\"{$count[$i]}\>";
}
```

Zmienna `$count` zawiera liczbę odwiedzin w postaci ciągu znaków, a `$strLength` to długość tego ciągu. Po wykonaniu całej pętli zmienna `$imgStr` będzie zawierała odpowiedni ciąg znaczników ``. Na przykład dla liczby 123 będzie to:

```

```

Oczywiście zakładamy, że obraz cyfry 1 znajduje się w pliku o nazwie *1.jpg*, cyfry 2 w pliku *2.jpg* itd. Ostatecznie kod skryptu przyjmie postać widoczną na poniższym listingu, a wczytanie go do przeglądarki spowoduje wyświetlenie widoku jak na rysunku 3.2.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Licznik graficzny</title>
</head>
<body>
<div style="text-align:center">
<h2>Tutaj znajduje się treść strony.</h2>
<br /><br />
</div>
<div style="text-align:center">
Ta strona została odwiedzona
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
autor: Marcin Lis http://marcinlis.com*/
```

Rysunek 3.2.
*Licznik odwiedzin
 w formie graficznej*



```
function getHits()
{
    if((($fp = @fopen("counter.txt", "r+")) === false)
        return false;
    $count = fgets($fp);
    $count = $count + 1;
    fseek($fp, 0);
    fputs($fp, $count);
    fclose($fp);

    $count = strval($count);
    $strLength = strlen($count);
    $imgStr = "";

    for($i = 0; $i < $strLength; $i++){
        $temp = $count[$i]. '.jpg';
        $imgStr .= "<img src=\"\$temp\" alt=\"{\$count[$i]}\">";
    }
    return $imgStr;
}
echo(getHits());
?>
    razy od 18 kwietnia 2008 r.
</div>

</body>
</html>
```

Skrypt 28. Licznik filtrujący adresy IP

Prosta modyfikacja skryptu 25. pozwoli nam na odrzucenie zliczania wywołań strony, przychodzących z wybranych adresów IP. W ten sposób możemy zablokować wywołania np. z adresu lokalnego 127.0.0.1. Adresy, które chcemy zablokować, umieścimy w tablicy \$ip_table. Sprawdzaniem, czy dane IP znajduje się w tej tablicy, zajmie się funkcja banIP. Zwróci ona wartość true, jeśli dane wywołanie ma zostać odrzucone, lub wartość false — w przeciwnym wypadku. Wystarczy zatem sprawdzać wartość zwracaną przez badIP i w zależności od tego zwiększać lub nie wartość licznika. Adres IP,

z którego nadeszło bieżące wywołanie strony, odczytujemy z tablicy \$_SERVER, odwołując się do klucza REMOTE_ADDR.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
</head>
<body>
<div style="text-align:center">
<h2>Tutaj znajduje się treść strony.</h2>
<br /><br />
</div>
<div style="text-align:center">
Ta strona została odwiedzona
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
   autor: Marcin Lis                               http://marcinlis.com*/

function badIP($ip)
{
    $ip_table = array(
        0 => "127.0.0.1",
        1 => "10.0.0.1"
    );
    foreach($ip_table as $bannedIP){
        if($bannedIP == $ip)
            return true;
    }
    return false;
}

function getHits()
{
    if(($fp = @fopen("counter.txt", "r+")) === false)
        return false;
    $count = fgets($fp);
    if(!badIP($_SERVER['REMOTE_ADDR'])){
        $count = $count + 1;
        fseek($fp, 0);
        fputs($fp, $count);
    }
    fclose($fp);
    return $count;
}
echo(getHits());
?>
    razy od 18 kwietnia 2008 r.
</div>

</body>
</html>

```

Skrypt 29. Licznik filtrujący adresy IP II

Czasami istnieje konieczność częstego uaktualniania listy adresów IP, które mają być pomijane przy zliczaniu odwiedzeń do naszej strony. W przypadku skryptu 28. jest to możliwe, jednak ingerujemy wtedy bezpośrednio w jego treść. Może to być niezbyt wygodne, np. w sytuacji, kiedy jedna osoba zajmuje się programowaniem skryptu, a inna uaktualnianiem bazy adresów IP. Dobrym rozwiązaniem takiego problemu może być przechowywanie listy adresów w oddzielnym pliku tekstowym.

Modyfikacji, a dokładniej napisania od nowa, wymaga znana nam już z wcześniejszego przykładu funkcja `badIP`. Należy w niej otworzyć plik z danymi, może on mieć nazwę np. `banned_ip.txt`, oraz sprawdzić, czy któryś ze znajdujących się w nim zapisów odpowiada adresowi IP, z którego nadeszło bieżące wywołanie. Plik ten będzie w postaci tekstowej, a każdy jego wiersz będzie zawierał dokładnie jeden adres. Do odczytu kolejnych wierszy wykorzystamy funkcję `fgets`.

Stosując skrypt tego typu, musimy jednak pamiętać, że przy każdym wywołaniu strony odwołuje się on do pliku `banned.txt`. Przy dużej liczbie odwiedzających będzie on więc mniej wydajny niż skrypt 28., gdzie adresy IP były wpisane bezpośrednio w kod. Pamiętajmy również, że plik z adresami musi być utworzony przez nas „ręcznie”. Skrypt nie podejmuje bowiem próby jego utworzenia.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
</head>
<body>
<div style="text-align:center">
<h2>Tutaj znajduje się treść strony.</h2>
<br /><br />
</div>
<div style="text-align:center">
Ta strona została odwiedzona
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
   autor: Marcin Lis                               http://marcinlis.com*/
```

```
function badIP($ip)
{
    $fd = fopen("banned_ip.txt", "r");
    while (!feof($fd)) {
        $line = trim(fgets($fd));
        if($line == $ip){
            fclose($fd);
            return true;
        }
    }
    fclose($fd);
    return false;
}
```

```

function getHits()
{
    if(($fp = @fopen("counter.txt", "r+")) === false)
        return false;
    $count = fgets($fp);
    if(!badIP($_SERVER['REMOTE_ADDR'])){
        $count = $count + 1;
        fseek($fp, 0);
        fputs($fp, $count);
    }
    fclose($fp);
    return $count;
}
echo(getHits());
?>
    razy od 18 kwietnia 2008 r.
</div>

</body>
</html>

```

Skrypt 30. Licznik uwzględniający tylko jedno odwołanie z danego IP

Odpowiednio modyfikując kod zaprezentowany w skrypcie 29., możemy pokusić się o stworzenie licznika, który będzie uwzględniał tylko jedno odwołanie z danego adresu IP. To znaczy, że jeśli z danego komputera nastąpiło chociaż jedno odwołanie do strony, to już nie będzie ono więcej uwzględniane przez licznik.

Ponownie należy zatem zmodyfikować funkcję `badIP`. Po pierwsze, dodamy instrukcje sprawdzające, czy plik `banned_ip.txt` istnieje. Jeśli nie, podejmiemy próby jego utworzenia. Po drugie, jeśli adres, z którego nastąpiło bieżące wywołanie, nie znajduje się w tym pliku, odpowiednio uaktualnimy dane. Zapisu dokonamy za pomocą funkcji `fputs`. Należy również pamiętać, aby przed zapisem przesunąć wskaźnik pliku na jego koniec, korzystając z kombinacji funkcji `fseek` i `filesize`.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
</head>
<body>
<div style="text-align:center">
<h2>Tutaj znajduje się treść strony.</h2>
<br /><br />
</div>
<div style="text-align:center">
Ta strona została odwiedzona
<?php

```

```
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"  
autor: Marcin Lis http://marcinlis.com*/  
  
function badIP($ip)  
{  
    $end_line = "\r\n";  
    if(file_exists("banned_ip.txt")){  
        $fd = fopen("banned_ip.txt", "r+");  
    }  
    else{  
        $fd = fopen("banned_ip.txt", "a+");  
    }  
    if(!$fd) return false;  
    while (!feof($fd)) {  
        $line = trim(fgets($fd));  
        if($line == $ip){  
            fclose($fd);  
            return true;  
        }  
    }  
    fseek($fd, filesize("banned_ip.txt"));  
    fputs($fd, $ip.$end_line);  
    fclose($fd);  
    return false;  
}  
  
function getHits()  
{  
    if(!($fp = @fopen("counter.txt", "r+")) === false)  
        return false;  
    $count = fgets($fp);  
    if(!badIP($_SERVER['REMOTE_ADDR'])){  
        $count = $count + 1;  
        fseek($fp, 0);  
        fputs($fp, $count);  
    }  
    fclose($fp);  
    return $count;  
}  
echo(getHits());  
?>  
razy od 18 kwietnia 2008 r.  
</div>  
  
</body>  
</html>
```

Skrypt 31. Głosowanie (ankieta)

Równie popularne jak liczniki są ankiety umożliwiające zbieranie opinii na różne tematy. Napisanie takiego skryptu wymaga nieco więcej pracy niż w przypadku zwykłego licznika, nie jest jednak szczególnie skomplikowane. Nasza ankieta będzie się

składała z dwóch części. Pierwsza z nich to kod HTML wyświetlający pytanie oraz formularz umożliwiający wybranie odpowiedzi, druga to skrypt PHP przetwarzający dane oraz wyświetlający wyniki. Kod formularza HTML będzie miał następującą postać:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Ankieta</title>
</head>
<body>
<div>
<p style="font-weight:bold">Jaki jest Twój ulubiony kolor?</p>
</div>
<div>
<form method="get" action="ankieta.php">
<table>
  <tr>
    <td>czerwony</td>
    <td>
      <input type="radio" name="vote" value="czerwony">
    </td>
  </tr>
  <tr>
    <td>zielony</td>
    <td>
      <input type="radio" name="vote" value="zielony">
    </td>
  </tr>
  <tr>
    <td>niebieski</td>
    <td>
      <input type="radio" name="vote" value="niebieski">
    </td>
  </tr>
  <tr>
    <td>fioletowy</td>
    <td>
      <input type="radio" name="vote" value="fioletowy">
    </td>
  </tr>
  <tr>
    <td>czarny</td>
    <td>
      <input type="radio" name="vote" value="czarny">
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <input type="submit" value="Głosuj">
      <input type="hidden" name="action" value="vote">
    </td>
  </tr>
</table>
</form>
</div>
```

```

<div>
<a href="ankieta.php?action=show"> Zobacz wyniki </a>
</div>
</body>
</html>

```

Po wczytaniu go do przeglądarki ujrzymy widok zaprezentowany na rysunku 3.3. Jak widać, ankieta będzie pozwalała na oddanie głosu na ulubiony kolor. Oczywiście jest to tylko przykład, który może być dowolnie modyfikowany. Formularz został utworzony za pomocą standardowych znaczników HTML `<form>` i `<input>`, a do jego formatowania została użyta zwykła tabela. Dane przekazywane będą do skryptu *ankieta.php* za pomocą metody GET.

Rysunek 3.3.

Formularz
umożliwiający
głosowanie



Za formularzem został umieszczony odnośnik, który będzie umożliwiał obejrzenie wyników głosowania bez oddawanie głosu. Odnośnik ten ma postać:

```
<a href="http://localhost/ankieta.php?action=show">
```

a zatem jeśli do skryptu *ankieta.php* zostanie przekazany parametr `action` o wartości `show`, będzie to oznaczało, że ankieta ma zostać jedynie wyświetlona na ekranie, jeśli natomiast parametr `action` będzie miał wartość `vote` (zapewnia to ukryte pole formularza: `<input type="hidden" name="action" value="vote">`), będzie to oznaczało, że ma zostać zapisany nowy głos.

Treść skryptu przetwarzającego dane (zapisanego w pliku *ankieta.php*) będzie następująca:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Wyniki głosowania</title>
</head>
<body>
<div>
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
autor: Marcin Lis http://marcinlis.com*/

```

```

function vote()
{
    if(!isset($_GET["vote"]) || $_GET["vote"] == ""){
        echo("Nie został zaznaczony żaden z kolorów.");
        return false;
    }
    else{
        $color = $_GET["vote"];
    }

    if(file_exists("colors.txt")){
        if(($fp = fopen("colors.txt", "r+")) === false){
            echo("Błąd serwera. Głos nie został uwzględniony.");
            return false;
        }
    }
    else{
        if(($fp = fopen("colors.txt", "w+")) === false){
            echo("Błąd serwera. Głos nie został uwzględniony.");
            return false;
        }
    }

    $czerwony = intval(fgets($fp));
    $zielony = intval(fgets($fp));
    $niebieski = intval(fgets($fp));
    $fioletowy = intval(fgets($fp));
    $czarny = intval(fgets($fp));

    switch($color){
        case "czerwony":$czerwony++;break;
        case "zielony":$zielony++;break;
        case "niebieski":$niebieski++;break;
        case "fioletowy":$fioletowy++;break;
        case "czarny":$czarny++;break;
    }
    fseek($fp, 0);
    fputs($fp, $czerwony."\r\n");
    fputs($fp, $zielony."\r\n");
    fputs($fp, $niebieski."\r\n");
    fputs($fp, $fioletowy."\r\n");
    fputs($fp, $czarny."\r\n");
    fclose($fp);
    return true;
}

function show()
{
    if(file_exists("colors.txt")){
        if(($fp = fopen("colors.txt", "r")) === false){
            echo("Błąd serwera. Wyniki ankiety nie są dostępne.");
            return;
        }
    }
    else{
        if(($fp = fopen("colors.txt", "w+")) === false){
            echo("Błąd serwera. Wyniki ankiety nie są dostępne.");
        }
    }
}

```

```

        return;
    }
}

$czerwony = intval(fgets($fp));
$zielony = intval(fgets($fp));
$niebieski = intval(fgets($fp));
$fioletowy = intval(fgets($fp));
$czarny = intval(fgets($fp));

$votes_no = $czerwony + $zielony + $niebieski + $fioletowy + $czarny;
$votes_no == 0?$votes_no = 1:0;

$czerwony_proc = sprintf("%.2f", $czerwony * 100 / $votes_no);
$zielony_proc = sprintf("%.2f", $zielony * 100 / $votes_no);
$niebieski_proc = sprintf("%.2f", $niebieski * 100 / $votes_no);
$fioletowy_proc = sprintf("%.2f", $fioletowy * 100 / $votes_no);
$czarny_proc = sprintf("%.2f", $czarny * 100 / $votes_no);
echo("<table border='1'>");

echo("<tr><td>Nazwa koloru</td><td>Liczba głosów</td>");
echo("<td>Procent głosów</td></tr>");

echo("<tr><td>czerwony</td><td>$czerwony</td>");
echo("<td>$czerwony_proc</td></tr>");

echo("<tr><td>zielony</td><td>$zielony</td>");
echo("<td>$zielony_proc</td></tr>");

echo("<tr><td>niebieski</td><td>$niebieski</td>");
echo("<td>$niebieski_proc</td></tr>");

echo("<tr><td>fioletowy</td><td>$fioletowy</td>");
echo("<td>$fioletowy_proc</td></tr>");

echo("<tr><td>czarny</td><td>$czarny</td>");
echo("<td>$czarny_proc</td></tr>");

echo("</table>");
}

if(isset($_GET["action"])){
    if($_GET["action"] == "show"){
        show();
    }
    else if($_GET["action"] == "vote"){
        if(vote()) show();
    }
    else{
        echo("Otrzymano nieprawidłowe dane.");
    }
}
else{
    echo("Otrzymano nieprawidłowe dane.");
}
?>

```

```

</div>
</body>
</html>

```

Jego wykonywanie rozpoczyna się od sprawdzenia, czy w tablicy \$_GET znajduje się indeks o nazwie action. Jeśli nie, oznacza to, że do skryptu nie zostały przekazane odpowiednie dane, do przeglądarki jest więc wysyłany odpowiedni komunikat i skrypt kończy działanie. Jeśli jednak indeks action jest obecny, sprawdzana jest jego wartość. Jeśli jest to ciąg show, wywoływana jest funkcja show wyświetlająca dane, jeśli natomiast jest to ciąg vote, to wywoływana jest funkcja vote zapisująca nowy głos oraz jeżeli wywołanie vote zakończyło się sukcesem, również funkcja show. W przypadku, kiedy wartością klucza action nie jest ani vote, ani show, do przeglądarki wysyłany jest komunikat o nieprawidłowym wywołaniu skryptu (nieprawidłowych danych).

Zadaniem funkcji show jest wyświetlenie tabeli zawierającej wyniki głosowania. Przykładowy wygląd takiej tabeli został zaprezentowany na rysunku 3.4. Działanie funkcji rozpoczyna się od sprawdzenia, czy istnieje plik colors.txt zawierający dane. Jeśli plik istnieje, następuje próba otwarcia go w trybie do odczytu, jeśli natomiast nie istnieje, następuje próba utworzenia go przez wywołanie funkcji fopen w trybie w+. Jeżeli którakolwiek z tych operacji zakończy się niepowodzeniem, funkcja wyświetla komunikat o błędzie i skrypt kończy działanie.

Rysunek 3.4.
Tabela zawierająca
wyniki ankiety



Nazwa koloru	Liczba głosów	Procent głosów
czerwony	1562	32.91
zielony	894	18.84
niebieski	1411	29.73
fioletowy	259	5.46
czarny	620	13.06

Po otwarciu pliku za pomocą funkcji fread odczytywane są jego kolejne wiersze. Pierwszy wiersz zawiera informację o liczbie głosów oddanych na kolor czerwony, drugi o liczbie głosów oddanych na kolor zielony itd. Odczytane wartości poddawane są konwersji do typu int (odpowiadają za to wywołania funkcji intval) i przypisywane zmiennym: \$czerwony, \$zielony, \$niebieski, \$fioletowy i \$czarny. Całkowita liczba oddanych głosów jest zapisywana w zmiennej \$votes_no, wartość ta jest następnie wykorzystywana do obliczenia, jaki procent głosów uzyskał każdy z kolorów. Do obliczenia procentowego udziału głosów wykorzystywany jest wzór:

$$\text{procent} = \text{liczba głosów na dany kolor} * 100 / \text{całkowita liczba głosów}$$

Wyniki są formatowane z dokładnością do dwóch miejsc po przecinku za pomocą funkcji sprintf. Uzyskane w ten sposób dane umieszczane są w tabeli HTML generowanej za pomocą standardowych znaczników: <table>, <tr> i <td>, i wysyłane do przeglądarki za pomocą instrukcji echo.

Zadaniem funkcji `vote` jest umieszczenie aktualnie otrzymanego głosu w pliku `colors.txt`. Jej działanie rozpoczyna się od sprawdzenia, czy w tablicy `$_GET` znajduje się klucz o nazwie `vote` oraz czy nie jest on pustym ciągiem znaków. Jeżeli jeden z wymienionych warunków jest prawdziwy, oznacza to, że do skryptu nie zostały przekazane poprawne dane, a zatem funkcja wyświetla komunikat o błędzie i kończy działanie. Jeśli jednak klucz o nazwie `vote` istnieje, jego wartość jest przypisywana zmiennej `color`. Po wykonaniu tych czynności sprawdzane jest, czy istnieje plik `colors.txt`. Jeśli istnieje, podejmowana jest próba otwarcia go w trybie `r+`, jeśli natomiast nie istnieje, podejmowana jest próba otwarcia go w trybie `w+` (co jest w tym przypadku równoznaczne z próbą utworzenia pliku na dysku). Jeżeli próby te zakończą się niepowodzeniem, funkcja wyświetla komunikat o błędzie i kończy działanie.

Po otwarciu pliku następuje odczyt jego zawartości, analogicznie jak miało to miejsce w przypadku funkcji `show`. Liczba głosów oddanych na poszczególne kolory przypisywana jest zmiennym: `$czerwony`, `$zielony`, `$niebieski`, `$fioletowy` i `$czarny`. Następnie za pomocą instrukcji wyboru `switch` sprawdzane jest, na jaki kolor został oddany bieżący głos (czyli jaki kolor wskazuje zmienna `$color`), i opowiadająca mu zmienna jest zwiększana o jeden. Na zakończenie wskaźnik pozycji w pliku jest przesuwany na pozycję zerową (na początek pliku), wszystkie dane są ponownie zapisywane i plik jest zamykany za pomocą funkcji `fclose`.

Skrypt 32. Księga gości

Skrypt generujący księgę gości będzie się składał z trzech części: szablonu głównego, który należy zapisać pod nazwą `guestbook.php`, formularza służącego do dodawania wpisów, który należy zapisać pod nazwą `add.inc`, oraz skryptu zarządzającego wpisami, który należy zapisać pod nazwą `guestbook.inc`. Wygląd przykładowej księgi gości utworzonej przez taki zestaw skryptów obrazuje rysunek 3.5.

Szablon główny (plik `guestbook.php`) będzie miał postać:

```

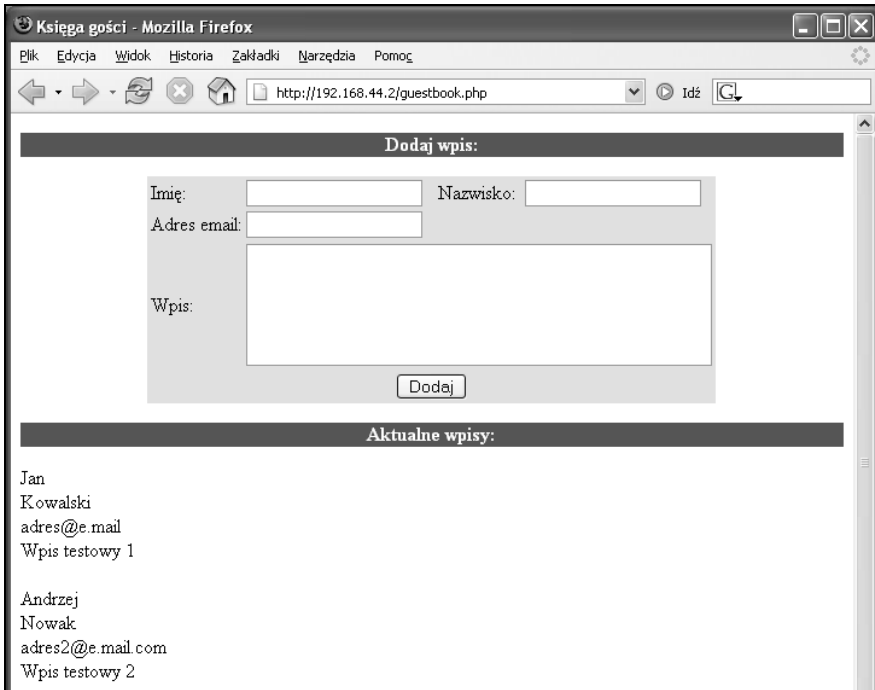
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Księga gości</title>
</head>
<body>

<p style="text-align:center;background-color:red;
        color:white;font-weight:bold">
Dodaj wpis:
</p>

<?php include("add.inc"); ?>

<p style="text-align:center;background-color:red;
        color:white;font-weight:bold">

```



Rysunek 3.5. Przykładowa księga gości

```
Aktualne wpisy:
</p>

<?php include("guestbook.inc"); ?>

</body>
</html>
```

Zawiera on jedynie kod HTML tworzący strukturę strony. Znajdują się w nim dyrektywy `include`, które wczytują pozostałe części skryptu, czyli pliki `add.inc` i `guestbook.inc`. W pliku `add.inc` znajduje się formularz pozwalający na dodawanie wpisów. Formularz został utworzony za pomocą standardowych elementów języka HTML, czyli znaczników `<form>` oraz `<input>`; jest on pozycjonowany za pomocą typowej tabeli i ma postać:

```
<form action="guestbook.php"
      method="post"
  >
  <table border="0"
        align="center"
        style="background-color:yellow;"
  >
  <tr>
    <td>Imię:</td>
    <td>
      <input type="text" name="imie" style="width:150">
    </td>
    <td>Nazwisko:</td>
```

```

        <td>
            <input type="text" name="nazwisko" style="width:152">
        </td>
    </tr><tr>
        <td>Adres email:</td>
        <td colspan="3">
            <input type="text" name="email" style="width:380">
        </td>
    </tr><tr>
        <td>Wpis:</td>
        <td colspan="3">
            <textarea name="contents" rows="5" cols="45" style="width:380"></textarea>
        </td>
    </tr><tr>
        <td colspan="4" align="center">
            <input type="submit" value="Dodaj">
        </td>
    </tr>
</table>
</form>

```

Częścią najważniejszą jest jednak sam skrypt zarządzający wpisami znajdujący się w pliku *guestbook.inc*:

```

<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
   autor: Marcin Lis                               http://marcinlis.com*/

function removeTags($str)
{
    return htmlentities($str);
}

function readGuestBook()
{
    $contents = "";
    if(($fp = fopen("guestbook.txt", "r")) === false)
        return false;
    while(!feof($fp)){
        $contents .= fgets($fp) . "<br />";
    }
    fclose($fp);
    return $contents;
}

function addToGuestBook($imie, $nazwisko, $email, $contents)
{
    if(($fp = @fopen("guestbook.txt", "r")) === false)
        return false;

    $imie = removeTags($imie);
    $nazwisko = removeTags($nazwisko);
    $email = removeTags($email);
    $contents = removeTags($contents);

    $tempC = fread($fp, filesize("guestbook.txt"));
    fclose($fp);

```



```

    $fp = fopen("guestbook.txt", "w");
    fputs($fp, $imie."\r\n");
    fputs($fp, $nazwisko."\r\n");
    fputs($fp, $email."\r\n");
    fputs($fp, $contents."\r\n");
    fputs($fp, "\r\n");
    fputs($fp, $tempC);
    fclose($fp);
}

if(isset($_POST["imie"])){
    $imie = $_POST["imie"];
}
else{
    $imie = "";
}

if(isset($_POST["nazwisko"])){
    $nazwisko = $_POST["nazwisko"];
}
else{
    $nazwisko = "";
}

if(isset($_POST["email"])){
    $email = $_POST["email"];
}
else{
    $email = "";
}

if(isset($_POST["contents"])){
    $contents = $_POST["contents"];
}
else{
    $contents = "";
}

if($imie == "" && $nazwisko == ""
    && $email == "" && $contents == ""){
    echo(readGuestBook());
}
else{
    addToGuestBook($imie, $nazwisko, $email, $contents);
    echo(readGuestBook());
}
?>

```

Wykonywanie kodu rozpoczyna się od odczytania zawartość tablicy `$_POST` i przypisania wartości poszczególnych pól do zmiennych: `$imie`, `$nazwisko`, `$email` i `$contents`. Jeśli po tym przypisaniu co najmniej jedna z wymienionych zmiennych nie zawiera pustego ciągu znaków, oznacza to, że użytkownik kliknął klawisz *Dodaj* i należy dokonać wpisu do księgi gości.

Za dodanie nowego wpisu odpowiada funkcja o nazwie `addToGuestBook`. Wszystkie niezbędne dane dotyczące wpisu, czyli: imię, nazwisko, adres e-mail oraz treść wpisu, są

jej przekazywane w postaci argumentów. W tej funkcji otwierany jest plik *guestbook.txt* w trybie *r*, czyli tylko do odczytu, a następnie jego zawartość jest odczytywana i przypisywana tymczasowej zmiennej *\$tempC*.

Po wykonaniu wymienionych operacji plik jest zamykany za pomocą funkcji *fclose*, a następnie ponownie otwierany, tym razem w trybie *w* — tylko do zapisu. Otwarcie w tym trybie powoduje obcięcie długości pliku do zera, a tym samym skasowanie starej zawartości. Za pomocą serii instrukcji *fputs* zapisywane są zatem wszystkie nowe dane, a na końcu dopisywana jest zawartość tymczasowej zmiennej *\$tempC*. Dzięki temu ostatnio wprowadzone przez użytkownika dane zawsze będą znajdowały się na początku księgi gości.

Do wyświetlenia zawartości pliku *guestbook.txt* wykorzystywana jest funkcja *print-GuestBook*. Jej konstrukcja jest bardzo prosta — w pętli *while* za pomocą funkcji *fgets* są wczytywane i dopisywane do zmiennej *\$contents* kolejne wiersze tekstu. Dodatkowo do każdego odczytanego łańcucha znaków jest dodawany znacznik HTML `
`. Ostatecznie, po osiągnięciu końca pliku, zmienna *\$contents* będzie zawierała gotową do wyświetlenia na ekranie treść księgi gości.

Skrypt zawiera również funkcjonalność polegającą na uniemożliwieniu użytkownikowi wprowadzenia do treści wpisu znaczników HTML, które mogłyby zaburzyć układ księgi. Taka blokada może być wprowadzona na wiele różnych sposobów, można na przykład zdefiniować, które znaczniki HTML będą akceptowane, i odrzucać pozostałe, można też filtrować wprowadzone dane i usuwać każdy napotkany znacznik (np. przy użyciu funkcji *strip_tags*). Innym sposobem jest wprowadzenie sekwencji znaków specjalnych. Należy wtedy zamienić wszystkie wystąpienia `<` na `<` i wszystkie wystąpienia `>` na `>` lub, lepiej, użyć funkcji *htmlentities*. Zaletą tego sposobu jest prostota oraz to, że w księdze pojawi się dokładnie to, co wpisał użytkownik, włącznie ze wszystkimi znacznikami (rysunek 3.6 przy skrypcie 33.).

Wyboru konkretnej techniki pozbycia się niebezpieczeństwa związanego ze znacznikami HTML należy dokonać według własnych potrzeb i uznania. Za usuwanie znaczników odpowiada funkcja *removeTags* i to od jej wewnętrznej realizacji zależy sposób wykonania tego zadania. W powyższym skrypcie została wykorzystana ostatnia z omawianych wcześniej metod, czyli wprowadzenie do kodu sekwencji znaków specjalnych, za pomocą funkcji *htmlentities*.

Funkcja *removeTags* jest wywoływana w funkcji *addToGuestBook* i przetwarza zawartość zmiennych *\$imie*, *\$nazwisko*, *\$email* i *\$contents*.

Skrypt 33. Księga gości z nawigacją

Księga gości przedstawiona w skrypcie 32. sprawdzała się dobrze w sytuacji, kiedy wpisów nie było zbyt wiele. Gdyby jednak wpisów miało się pojawić bardzo dużo, to ze względu na to, że wszystkie byłyby wyświetlane na jednej stronie, strona ładowałaby się dosyć długo, a samo przeglądanie księgi mogłoby być nużące. Ten problem można

rozwiązać przez dodanie do księgi możliwości nawigacyjnych, wymaga to jednak znacznej modyfikacji skryptu z poprzedniego przykładu. Bez zmian pozostaną pliki pomocnicze: *add.inc* i *guestbook.php*, natomiast zmodyfikowany kod skryptu *guestbook.inc* jest widoczny na poniższym listingu:

```
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
   autor: Marcin Lis                               http://marcinlis.com*/

$file = 2;
$rek_size = 5;

function removeTags($str)
{
    return htmlentities($str);
}

function skipRekord($fp, $rek_size)
{
    for($i = 0; $i < $rek_size; $i++){
        fgets($fp);
        if(feof($fp)){
            return false;
        }
    }
    return true;
}

function readGuestBook($from, $file, $rek_size)
{
    $contents = "";
    if(($fp = fopen("guestbook.txt", "r")) === false)
        return false;
    for($i = 0; $i < $from; $i++){
        if(!skipRekord($fp, $rek_size))
            break;
    }
    for($i = $from * $rek_size; $i < ($from + $file) * $rek_size; $i++){
        if(feof($fp)){
            break;
        }
        $tempStr = fgets($fp);
        $contents .= $tempStr."<br />";
    }
    fclose($fp);
    return trim($contents);
}

function addToGuestBook($imie, $nazwisko, $email, $contents)
{
    if(($fp = @fopen("guestbook.txt", "r")) === false)
        return false;

    $imie = removeTags($imie);
    $nazwisko = removeTags($nazwisko);
    $email = removeTags($email);
    $contents = removeTags($contents);
```

```
    $tempC = fread($fp, filesize("guestbook.txt"));
    fclose($fp);
    $fp = fopen("guestbook.txt", "w");
    fputs($fp, $imie."\r\n");
    fputs($fp, $nazwisko."\r\n");
    fputs($fp, $email."\r\n");
    fputs($fp, $contents."\r\n");
    fputs($fp, "\r\n");
    fputs($fp, $tempC);
    fclose($fp);
}

if(isset($_GET["from"])){
    $from = $_GET["from"];
}
else{
    $from = 0;
}

if($from <= 0){
    $from = 0;
    $prev = 0;
    $next = $ile;
}
else{
    $prev = $from - $ile;
    $next = $from + $ile;
}

if(isset($_POST["imie"])){
    $imie = $_POST["imie"];
}
else{
    $imie = "";
}

if(isset($_POST["nazwisko"])){
    $nazwisko = $_POST["nazwisko"];
}
else{
    $nazwisko = "";
}

if(isset($_POST["email"])){
    $email = $_POST["email"];
}
else{
    $email = "";
}

if(isset($_POST["contents"])){
    $contents = $_POST["contents"];
}
else{
    $contents = "";
}
}
```

```

if($imie == "" && $nazwisko == ""
    && $email == "" && $contents == ""){
    $tempStr = readGuestBook($from, $ile, $rek_size);
    if($tempStr == "" || $tempStr == "\r\n"){
        echo("Koniec wpisów". "<br />");
        $next -= $ile;
    }
    else{
        echo($tempStr);
    }
}
else{
    addToGuestBook($imie, $nazwisko, $email, $contents);
    echo(readGuestBook($from, $ile, $rek_size));
}

$code = <<<TEMP
<a href="guestbook.php?from=$prev">Poprzednie</a>
<a href="guestbook.php?from=$next">Następne</a>
TEMP;
echo("$code");
?>

```

Na początku zostały zdefiniowane dwie nowe zmienne: `$ile`, wskazująca liczbę wpisów wyświetlanych na pojedynczej stronie, oraz `$rek_size`, określająca, ile linii zawiera pojedynczy wpis. Bardzo przydatna jest również dodatkowa funkcja o nazwie `skipRekord`, która przesuwaa wskaźnik aktualnej pozycji w pliku na następny wpis. Jej działanie polega na odczytaniu liczby linii wskazywanych przez zmienną `$rek_size`.

Odczytem danych z książki gości zajmuje się funkcja `readGuestBook`. W stosunku do przedstawionej w skrypcie 32. wymagała ona jednak wielu zmian. Przede wszystkim w aktualnej wersji otrzymuje trzy argumenty: `$from`, `$ile` oraz `$rek_size`. Znaczenie dwóch ostatnich zostało wyjaśnione w poprzednim akapicie, argument `$from` oznacza natomiast, od którego rekordu ma się zacząć czytanie pliku, i jest wykorzystywany w dodatkowej pętli pomijającej zadaną liczbę rekordów.

Kolejna pętla, zajmująca się właściwym odczytem danych, musi w obliczeniach uwzględnić przesunięcie o `$from` rekordów. Niezbędne jest zatem wykonanie kilku dodatkowych działań arytmetycznych, tak aby prawidłowo wyliczyć warunek jej zakończenia. Stąd postać pętli:

```

for($i = $from * $rek_size; $i < ($from + $ile) * $rek_size; $i++){
    if(feof($fp)){
        break;
    }
    $contents .= fgets($fp). "<br />";
}

```

Do nawigacji pomiędzy kolejnymi wpisami są wykorzystywane typowe odnośniki HTML definiowane znacznikami `<a>`. Są one wyświetlane na dole książki gości, po ostatnim wpisie na stronie (rysunek 3.6). Odnośniki mają postać:

```

<a href="guestbook.php?from=$prev">Poprzednie</a>
<a href="guestbook.php?from=$next">Następne</a>

```

Dodaj wpis:

Imię: Nazwisko:

Adres email:

Wpis:

Aktualne wpisy:

Jan
Kowalski
adres@e.mail
To mój adres

Andrzej
Nowak
adres2@e.mail.com
Wpis testowy 2

[Poprzednie](#) [Następne](#)

Rysunek 3.6. Znaczniki HTML zostały przedstawione w postaci tekstowej

Każdy odnośnik ponownie wywołuje skrypt, przypisując polu `from` wartość zawartą w zmiennej `$prev` lub `$next`. Zmienne te muszą być oczywiście odpowiednio zainicjowane. Jeżeli `$from` będzie mniejsze lub równe 0, to `$prev` otrzymuje wartość 0, a `$next` wartość `$ile`. W przeciwnym przypadku (czyli kiedy `$from` jest większe od 0), `$prev` otrzymuje wartość `$from - $ile`, a `$next` wartość `$from + $ile`. Niezbędne jest również zablokowanie możliwości powiększania wartości zmiennej `$next` w nieskończoność. W tym celu jest sprawdzane, czy funkcja `readGuestBook` zwróciła pusty ciąg znaków lub też ciąg oznaczających koniec linii tekstu. W obu przypadkach na ekranie jest wyświetlany tekst „Koniec wpisów”, a zmienna `$next` jest pomniejszana o wartość zapisaną w zmiennej `$ile`.

Skrypt 34. Porada dnia

Prostym, jednak często spotykanym, skryptem jest tak zwana porada dnia, czyli wyświetlenie pewnej sentencji wybranej losowo z przygotowanego wcześniej zbioru. Może to być myśl filozoficzna, przysłowie lub dowcip, zależy to jedynie od pomysłowości autora witryny oraz jej charakteru. Porada dnia może zostać wygenerowana za pomocą poniższego skryptu:

```
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
   autor: Marcin Lis                               http://marcinlis.com*/

$porady = array(
    0 => "Porada numer 1",
```

```
1 => "Porada numer 2",
2 => "Porada numer 3",
3 => "Porada numer 4",
4 => "Porada numer 5",
);

function getText()
{
    srand((float) microtime() * 10000000);
    $arr = $GLOBALS["porady"];
    return($arr[array_rand($arr)]);
}

echo(getText());
?>
```

Lista porad powinna zostać zapisana w tablicy o nazwie `porady`. Za wylosowanie jednej z nich odpowiada funkcja `getText`. Inicjuje ona najpierw generator liczb pseudolosowych¹, następnie przypisuje do zmiennej `$arr` globalną tablicę `$porady`. Losowanie odbywa się za pomocą funkcji `array_rand`, która zwraca losowy indeks tablicy przekazanej tej funkcji w postaci argumentu. Wylosowany tekst jest zwracany za pomocą instrukcji `return`.

Skrypt 35. Porada dnia z hasłami w pliku

Skrypt 34. pokazywał, w jaki sposób wyświetlić na stronie poradę dnia. Wszystkie hasła przechowywane były jednak bezpośrednio w kodzie skryptu w globalnej tablicy `$porady`. Jeśli porad jest dużo i często się zmieniają lub są uaktualnianie, lepszym rozwiązaniem może być umieszczenie ich w zewnętrznym pliku, tak jak pokazuje to poniższy kod:

```
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
   autor: Marcin Lis                               http://marcinlis.com*/

function readArray()
{
    $texts = array();
    if(!$fd = @fopen("porady.txt", "r")) return false;
    $counter = 0;
    while (!feof ($fd)){
        $str = trim(fgets($fd));
        if($str != "")
            $texts[$counter++] = $str;
    }
}
```

¹ Począwszy od PHP 4.0.2, inicjalizacja generatora liczb pseudolosowych nie jest konieczna (ta czynność jest wykonywana automatycznie), jednak w celu zachowania maksymalnej kompatybilności została pozostawiona w skrypcie.

```
    }
    return $texts;
}

function getText()
{
    if(($arr = readArray()) != false){
        srand((float) microtime() * 10000000);
        return($arr[array_rand($arr)]);
    }
    else return "Brak porady dnia na dzisiaj...";
}
echo(getText());
?>
```

Wszystkie porady powinny znaleźć się w pliku tekstowym o nazwie *porady.txt*; każda w oddzielnym wierszu. Dane z tego pliku odczytywane są w funkcji `readArray`. Każdy wiersz tekstu po odczytaniu jest umieszczany w oddzielnej komórce tablicy `$texts`. Tablica ta jest następnie zwracana za pomocą instrukcji `return`. W przypadku gdyby odczytanie danych z pliku nie powiodło się, funkcja zwróci wartość `false`.

Za wylosowanie jednego z tekstów odpowiada funkcja `getText`. Pobiera ona tablicę tekstów za pomocą wywołania funkcji `readArray`. Jeżeli wywołanie to powiedzie się, to znaczy zwrócona zostanie wartość różna od `false`, inicjowany jest generator liczb pseudolosowych (`srand((float) microtime() * 10000000);`) oraz wykonywane jest samo losowanie (odpowiada za to funkcja `array_rand`). Wylosowana wartość jest zwracana za pomocą instrukcji `return`.

Skrypt 36. Inna strona dla znanego użytkownika

Wykorzystując pliki cookies, możemy spowodować, aby użytkownikom, którzy pierwszy raz wchodzi na naszą stronę, wyświetlała się inna witryna niż tym, którzy już nas kiedyś odwiedzili. Użyjemy do tego celu cookie o nazwie `znany`. Jeśli będzie ono obecne na komputerze użytkownika, wczytywać będziemy plik *index1.html*, w przeciwnym razie — *index2.html*. W tym drugim przypadku będziemy również ustawiać cookie, wykorzystując w tym celu funkcję `setCookie`. Cały skrypt będzie miał następującą postać:

```
<?php
/*Skrypt pochodzi z książki "PHP. 101 praktycznych skryptów"
   autor: Marcin Lis                               http://marcinlis.com*/
if(isset($_COOKIE["znany"])){
    include('index1.html');
}
else{
    setCookie("znany", "tak", time() + 3600 * 24 * 30);
    include('index2.html');
}
?>
```