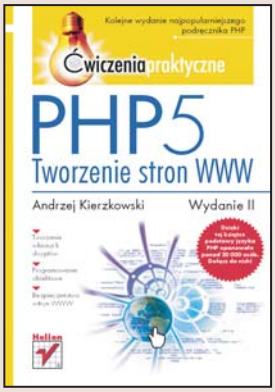
PHP5. Tworzenie stron WWW. Ćwiczenia praktyczne. Wydanie II



Autor: Andrzej Kierzkowski
ISBN: 83-246-0674-2
Format: A5, stron: 260

Kolejne wydanie najpopularniejszego podręcznika PHP

- Tworzenie własnych skryptów
- Programowanie obiektowe
- Bezpieczeństwo witryn WWW

Statyczne witryny WWW to początek kariery każdego webmastera. Jednak język HTML, choć ciągle rozwijany, ma ograniczone możliwości, co powoduje, że aby stworzyć niektóre elementy strony, będziesz musiał sięgnąć po inne technologie. Umieszczenie na stronie WWW interaktywnego menu, forum dyskusyjnego lub księgi gości wymaga zastosowania narzędzi innego rodzaju.

Chcesz wzbogacić swoją witrynę WWW o nowe, interaktywne elementy? Poznaj język PHP5!

Ten język programowania jest bardzo rozpowszechniony, efektywny, działa na wielu platformach, a co najważniejsze, jest dostępny bezpłatnie. Został zaprojektowany specjalnie do tworzenia aplikacji WWW. Jest też stosunkowo łatwy do opanowania i wygodny w użyciu.

Książka „PHP5. Tworzenie stron WWW. Ćwiczenia praktyczne. Wydanie II" to kolejne, zaktualizowane i uzupełnione, wydanie popularnego podręcznika przedstawiającego podstawy tworzenia aplikacji WWW. Czytając ją, nauczysz się obsługiwać za pomocą PHP5 formularze umieszczane na stronach WWW, wysyłać i odbierać pliki, zarządzać sesjami i cookies oraz korzystać z danych zgromadzonych w bazie MySQL. Rozdział poświęcony programowaniu obiektowemu został rozbudowany i wzbogacony o nowe przykłady i ćwiczenia. Książka dodatkowo zawiera rozdział o zagrożeniach wynikających ze stosowania PHP5 i sposobach zabezpieczania przed nimi witryn WWW.

- Instalowanie i konfiguracja PHP5
- Operatory
- Zmienne
- Pętle i konstrukcje warunkowe
- Przesyłanie danych z formularzy
- Obsługa plików i sesji
- Połączenia z bazami danych
- Tworzenie grafiki w PHP5
- Komponenty witryn WWW
- Programowanie obiektowe
- Zabezpieczanie aplikacji WWW

Dzięki tej książce podstawy języka PHP opanowało już prawie 20 000 osób. Dołącz do nich!

Wzbogać swój warsztat – poznaj język PHP5

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Dane — pobieranie, przekazywanie i przechowywanie

Programy PHP, które napisałeś do tej pory, nie były bardzo rozbudowane i nie szokowały swoimi możliwościami. Nie stanowiły też rozwiązania typowych problemów webmasterskich. Pewnie już zauważyłeś, że opierały się na komunikacji jednostronnej: program coś wyliczał i wyświetlał wyniki w przeglądarce. Nie można było mu jednak w żaden sposób przekazać danych ani też zapisać na serwerze wyników działania — w celu wykorzystania ich w przyszłości. Nie było też możliwości przekazywania wyników działania z jednej strony do drugiej. W tym rozdziale zajmiesz się tymi właśnie zagadnieniami: pobieraniem danych od użytkownika, pamiętaniem ich zarówno po stronie klienta, jak i serwera oraz przekazywaniem informacji pomiędzy stronami.

Pobieranie danych z formularzy

Podstawowym sposobem pobierania danych od użytkownika na stronach WWW są formularze. Zakładam, że mając styczność z HTML-em, zapoznałeś się już z tym tematem. Warto jednak przypomnieć kilka niezbędnych informacji.

Formularz na stronie WWW ma następującą postać:

```
<FORM ACTION="skrypt.php" METHOD=POST>
<!--Tu występują pola formularza -->
<INPUT TYPE=Submit VALUE="Wyślij">
</FORM>
```

Metodą w formularzu może być także GET (jest to zresztą metoda domyślna: jeżeli nie wpiszesz w definicji formularza żadnej, zostanie użyta właśnie ta). Metody różnią się między sobą sposobem przekazywania danych do serwera. Jeżeli użyjesz metody GET, dane z formularza zostaną doklejone do adresu skryptu odbierającego formularz po znaku zapytania. Adres będzie więc miał postać:

```
http://adres.serwera/skrypt.php?pole1=wart1&pole2=wart2&..&poleN=wartN
```

W przypadku formularzy z dużą liczbą danych taka metoda może być niewygodna. Ma jednak jedną niewątpliwą zaletę: można ją stosować nie tylko do obsługi formularzy, ale po prostu w łączu, które użytkownik będzie nawet mógł dodać do zakładek.

Metoda POST z kolei przekazuje dane z formularza w inny sposób (po nagłówkach zlecenia HTTP). Danych z formularza nie zobaczysz więc w adresie, ale można ich za to przesłać o wiele więcej.

W formularzu możesz używać: pól tekstowych, pól haseł, pól wyboru, przycisków opcji, obszarów tekstowych, menu, pól wyboru pliku do przesłania. Trudno w tym miejscu szczegółowo omawiać wszelkie elementy formularzy, dlatego zachęcam do zapoznania się z książką autorstwa Elizabeth Castro *Po prostu HTML 4. Wydanie III*, a w szczególności z jej 16. rozdziałem („Formularze").

Pobieranie danych z formularza przez skrypt jest trywialnym zadaniem. Dane wpisane w polach formularza są pamiętane w specjalnych tablicach ($_GET i $_POST), a indeksy, pod którymi są dostępne, noszą takie nazwy, jak pola formularza. Jeżeli formularz zostanie przesłany metodą GET, to wartości wpisane w polach znajdziesz w odpowiednich polach tablicy $_GET. Analogicznie jeżeli metodą przesłania formularza będzie POST, to dane będziesz mógł odczytać z tablicy $_POST.

ĆWICZENIE

4.1 Utworzenie prostego formularza

Utwórz program, który będzie wyświetlał formularz z jednym polem tekstowym, a po wysłaniu tego formularza wyświetli wpisaną wartość.

W zależności od tego, czy zmienna `$tekst` (`tekst` będzie nazwą pola w formularzu) przyjmuje jakąś wartość, czy też nie, wyświetlisz albo formularz, albo informację o tym, co zostało w nim wpisane.

4-01.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Formularz</TITLE>
  </HEAD>
  <BODY>
    <? // Drukuje formularz i jednocześnie odbiera i wyświetla wpisane
          w nim dane.

       if ($_GET['tekst']) {   // jest wpisana jakaś wartość w formularzu
         $tekst = $_GET['tekst'];
         print "Wpisana wartość to <B>$tekst</B><BR>";
         print '<A HREF="4-01.php">Powrót do formularza</A>";
       } else {    // nie ma wpisanych danych, wyświetlasz formularz
         print '<FORM ACTION="4-01.php" METHOD=GET>';
         print '<INPUT TYPE="text" NAME="tekst">';
         print '<INPUT TYPE="submit" VALUE="Wyślij">';
         print '</FORM>';
       }

    ?>
  </BODY>
</HTML>
```

Warto sprawdzić, jak działa skrypt, zarówno z wykorzystaniem metody `GET`, jak i `POST`. Pamiętaj, że poza zmianą metody w formularzu musisz też zmienić nazwę tablicy, z której są odczytywane dane, na `$_POST`. Zauważ różnicę w adresie po wysłaniu formularza za pomocą jednej i drugiej metody.

Wpisz w polu formularza tekst `<H1>tekst</H1>`. Pojawi się niespodziewany efekt — wyświetlony tekst zostanie wypisany dużą czcionką. Dzieje się tak, ponieważ wpisany tekst wklejasz bezpośrednio do kodu strony. Jeszcze ciekawszy efekt uzyskasz, wpisując tekst `<script>alert(123);</script>`.

Jak widzisz, łatwo stracić kontrolę nad tym, co wyświetla strona, jeżeli wyświetlane na niej dane pochodzą bezpośrednio z formularza. Sprawa nie jest poważna, jeżeli chodzi tylko o stronę wysyłaną pojedynczemu klientowi po wpisaniu przez niego danych. Można sobie jednak wyobrazić księgę gości z wpisami gromadzonymi w bazie danych i wyświetlanymi na żądanie. Jeżeli do bazy będziesz wpisywać nieobrobione dane z formularza, efekty działania odwiedzających (czasem zupełnie niepożądane) będą oglądali wszyscy. Z tego powodu powinieneś stosować funkcję „oczyszczającą" wpisywane dane, szczególnie takie, które mogą się pojawić na stronach WWW.

ĆWICZENIE

4.2 Bezpieczne wyświetlanie danych

Popraw program z ćwiczenia 4.1 tak, by przetwarzał wpisane dane, aby były bezpieczne w wyświetlaniu. Sprawdź jego działanie na przykładach z poprzedniego ćwiczenia.

W przyszłości zapewne przygotujesz własną funkcję „oczyszczającą" wprowadzane dane. W zależności od ich charakteru, funkcja powinna wykonywać różne operacje. Na początek wykorzystaj wbudowaną funkcję PHP, która znaczniki HTML-a przetworzy na „bezpieczne" do wyświetlenia. W szczególności zamieni znaki większości i mniejszości na `&gt;` i `&lt;`, co spowoduje, że wpisane znaczniki HTML nie zostaną przy ich dołączeniu do strony zinterpretowane, lecz wyświetlone.

`htmlspecialchars`	Zamienia znaczniki HTML na kody „bezpieczne" do wyświetlenia na stronach WWW.

4-02.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Formularz</TITLE>
  </HEAD>
  <BODY>
    <? // Drukuje formularz i jednocześnie odbiera i wyświetla wpisane
       // w nim dane. Dane są przetworzone funkcją htmlspecialchars.

       if ($_GET['tekst']) {  // jest wpisana jakaś wartość w formularzu
         $tekst = htmlspecialchars ($_GET['tekst']);
         print "Wpisana wartość to <B>$tekst</B><BR>';
         print '<A HREF="4-02.php">Powrót do formularza</A>";
       } else {   // nie ma wpisanych danych, wyświetlasz formularz
         print '<FORM ACTION="4-02.php" METHOD=GET>';
         print '<INPUT TYPE="text" NAME="tekst">';
         print '<INPUT TYPE="submit" VALUE="Wyślij">';
         print '</FORM>';
       }

    ?>
  </BODY>
</HTML>
```

Program różni się jednym wierszem przetwarzającym zmienną `$tekst`:

```
$tekst = htmlspecialchars ($tekst);
```

W przypadku standardowych danych trudno dostrzec różnicę. Jeżeli jednak użyje się któregoś ze stwarzających problemy wpisów z poprzedniego ćwiczenia, można zobaczyć efekt.

Dyskusyjną sprawą jest to, w jaki sposób przetwarzać wpisane przez użytkownika dane. W przykładzie z poprzedniego ćwiczenia wpisane przez użytkownika znaczniki zostaną po prostu wyświetlone. W przyszłości być może zechcesz je po prostu wyeliminować.

ĆWICZENIE

4.3 Obliczanie rozwiązań równania kwadratowego

Utwórz program, który będzie wyświetlał formularz z możliwością wpisania trzech danych: a, b i c, po czym wyświetli rozwiązanie równania kwadratowego: $ax^2 + bx + c = 0$.

Przypomnij sobie ćwiczenie 3.8. Rozwiązanie będzie podobne, ale pozwoli na interaktywne wpisywanie danych przez odwiedzającego stronę.

4-03.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Równanie kwadratowe</TITLE>
  </HEAD>
  <BODY>
    <? // Pobiera od użytkownika współczynniki równania, a następnie
       // oblicza wartości zerowe równania kwadratowego.
       $a = str_replace (",", ".", $_GET['a']); settype ($a, "double");
       $b = str_replace (",", ".", $_GET['b']); settype ($b, "double");
       $c = str_replace (",", ".", $_GET['c']); settype ($c, "double");
       if ($a || $b || $c) { // wartości w formularzu są ok
         print ("a = $a, b = $b, c = $c<BR>");
         if ($a) {
           $delta = $b*$b-4*$a*$c;
           if ($delta < 0) {
             print ('Równanie nie ma pierwiastków rzeczywistych');
           } elseif ($delta == 0) {
             $x1 = -$b/(2 * $a);
             print ("Równanie ma jeden pierwiastek rzeczywisty: $x1");
           } else {
             $x1 = (-$b-sqrt($delta)) / (2*$a);
             $x2 = (-$b+sqrt($delta)) / (2*$a);
             print ("Równanie ma dwa pierwiastki rzeczywiste: $x1 i $x2");
           }
         } elseif ($b) {
           $x1 = -$c/$b;
           print ("Równanie ma jeden pierwiastek rzeczywisty: $x1");
         } else {
           print ('Równanie nie ma pierwiastków rzeczywistych');
         }
         print '<BR><A HREF="4-03.php">Powrót do formularza</A>';
       } else { // nie ma wpisanych danych, wyświetlasz formularz
         print '<FORM ACTION="4-03.php" METHOD=GET>';
         print 'a: <INPUT TYPE="text" NAME="a"><BR>';
         print 'b: <INPUT TYPE="text" NAME="b"><BR>';
         print 'c: <INPUT TYPE="text" NAME="c"><BR>';
         print '<INPUT TYPE="submit" VALUE="Wyślij">';
         print '</FORM>';
       }

    ?>
  </BODY>
</HTML>
```

W programie wykorzystano funkcję `settype`, która zamienia (o ile się da) zmienną, będącą jej pierwszym parametrem, na typ określony drugim parametrem. Niektóre typy, które można zastosować, to `integer`, `double`, `string` oraz `array`. Zamiana na typ `double` (czyli na liczbę rzeczywistą) nie jest — niestety — doskonała. Funkcja bierze pod uwagę taki początek zmiennej, który odpowiada liczbie rzeczywistej, a jeżeli po nim występuje cokolwiek innego, zostaje opuszczone.

Przed wywołaniem funkcji `settype` została wykorzystana funkcja `str_replace`, która w łańcuchu stanowiącym jej trzeci parametr zamienia wszystkie wystąpienia pierwszego parametru na drugi. Uczyniono to po to, by umożliwić użytkownikowi oddzielenie części całkowitej i ułamkowej każdego ze współczynników zarówno znakiem kropki, jak i przecinka (dlatego przecinek zamieniasz na kropkę, która jest standardowym separatorem liczb ułamkowych).

ĆWICZENIE

4.4 Pobieranie nazwy i hasła użytkownika

Utwórz program, który po podaniu odpowiedniej nazwy użytkownika i jego hasła wyświetli tajną informację. Jeżeli zaś wpisane dane będą nieprawidłowe — informację o błędzie.

Dane o użytkownikach i hasłach zapamiętaj jawnie w tablicy asocjacyjnej. Oczywiście użytkownik nie będzie mógł ich podejrzeć. W przyszłości możesz zapamiętywać zaszyfrowane informacje w plikach tekstowych lub w bazie danych, jednak na początku takie proste rozwiązanie wystarczy.

4-04.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Informacja na hasło</TITLE>
  </HEAD>
  <BODY>
    <? // Pobiera od użytkownika nazwę i hasło. Jeżeli dane są prawidłowe,
       // wyświetla tajną informację, a jeżeli nie - informację o błędzie.
       $hasla = array ('admin' => 'hasloadmina15', 'tester' => 'tester15',
                       'marek' => '1wgd7w3', 'henryk' => '56sghx');
```

```
        if (($_POST['uzytkownik']) && ($_POST['haslo'])) { // wartości
                                                            w formularzu
                                                            są wpisane
          if ($hasla[$_POST['uzytkownik']] == $_POST['haslo']) {
            print ("Tajna informacja to: <B>2*2=4</B>");
          } else {
            print ("Wpisano niepoprawne dane o użytkowniku i haśle.<BR>");
            print ('<A HREF="4-04.php">Wróć</A> i spróbuj ponownie.');
          }
        } else { // nie ma wpisanych danych, wyświetlasz formularz
          print '<FORM ACTION="4-04.php" METHOD=POST>';
          print '<TABLE><TR><TD>użytkownik: </TD><TD><INPUT TYPE="text" ';
          print "NAME=\"uzytkownik\" VALUE=\"".$_POST['uzytkownik'].
          "\"></TD></TR>";
          print '<TR><TD>haslo: </TD><TD><INPUT TYPE="password" ';
          print 'NAME="haslo"></TD></TR></TABLE>';
          print '<INPUT TYPE="submit" VALUE="Wyślij">';
          print '</FORM>';
        }

    ?>
  </BODY>
</HTML>
```

W formularzu wykorzystano metodę `POST`, a nie `GET`. Zrobiłeś tak, aby nie ujawniać hasła użytkownika w adresie. Taka informacja mogłaby przypadkowo zostać ujawniona nieupoważnionym osobom (na przykład poprzez zapamiętanie w historii przeglądarki lub przypadkowe dodanie strony do zakładek).

Do wpisania hasła użytkownika w formularzu użyto pola typu `password`. Dzięki temu wpisywane hasło nie jest wyświetlane na ekranie (każdy znak jest zastępowany gwiazdką).

Zrobiłeś jeszcze jedno: w formularzu dla pola `uzytkownik` ustaliłeś wartość początkową — jako zmienną `$_POST['uzytkownik']`. Stało się tak dla wygody użytkownika. Załóżmy, że podczas wypełniania formularza nie zostaną wpisane kompletne dane (użytkownik wpisze jedynie nazwę, a zapomni o haśle). Wówczas po wysłaniu formularz zostanie ponownie wyświetlony, z wypełnionym już polem z nazwą.

W podobny sposób można wykorzystywać pola innych typów — na przykład menu czy opcji wyboru.

ĆWICZENIE

4.5 Wyświetlanie menu w formularzu I

Napisz program, który wyświetli menu, a następnie pokaże wybraną przez użytkownika opcję.

Wybraną przez użytkownika opcję określ za pomocą instrukcji `switch`. Na podstawie wartości zmiennej `$opcja` w `$cowybrano` zapamiętaj ją. Ponieważ została wykorzystana metoda `POST`, w adresie jej nie zauważysz.

4-05.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Menu w formularzu</TITLE>
  </HEAD>
  <BODY>
    <? // Wyświetla formularz z menu, a następnie wyświetla wybraną przez
       // użytkownika opcję.

       if ($_POST['opcja']) {    // wartości w formularzu są wpisane
         switch ($_POST['opcja']) {
           case 'pilka': $cowybrano = 'piłka nożna'; break;
           case 'kosz': $cowybrano = 'koszykówka'; break;
           case 'siat': $cowybrano = 'siatkówka'; break;
           case 'nar': $cowybrano = 'narciarstwo'; break;
           case 'hokej': $cowybrano = 'hokej'; break;
           case 'boks': $cowybrano = 'boks'; break;
           case 'inny': $cowybrano = 'inny sport'; break;
           default: $cowybrano = 'niezidentyfikowana opcja'; break;
         }
         print ("Użytkownik wybrał opcję: <B>$cowybrano</B>.<BR>");
         print '<BR><A HREF="4-05.php">Powrót do formularza</A>';
       } else {    // nie ma wpisanych danych, wyświetlasz formularz
         print '<FORM ACTION="4-05.php" METHOD=POST>';
         print '<SELECT NAME="opcja">';
         print '<OPTION SELECTED VALUE="">-&gt; wybierz, jak sport
         lubisz:';
         print '<OPTION VALUE="pilka">piłka nożna';
         print '<OPTION VALUE="kosz">koszykówka';
         print '<OPTION VALUE="siat">siatkówka';
```

```
        print '<OPTION VALUE="nar">narciarstwo';
        print '<OPTION VALUE="hokej">hokej';
        print '<OPTION VALUE="boks">boks';
        print '<OPTION VALUE="inny">inny sport';
        print '</SELECT>';
        print '<INPUT TYPE="submit" VALUE="Wyślij">';
        print '</FORM>';
      }

  ?>
 </BODY>
</HTML>
```

Zauważ, że wartość każdej z opcji nie musi nazywać się tak samo, jak tekst wypisywany użytkownikowi. Jest to dość wygodne (czasem opcje mają długie nazwy). W celu wyświetlenia pełnego opisu wybranej opcji można posłużyć się instrukcją `switch`. Dużo jednak wygodniej do pamiętania nazw opcji użyć tablicy asocjacyjnej, o czym przekonasz się w następnym ćwiczeniu.

ĆWICZENIE

4.6 Wyświetlanie menu w formularzu II

Popraw program z poprzedniego ćwiczenia tak, by nazwy opcji i wartości zostały zapisane w tablicy asocjacyjnej, a zmiana treści formularza była możliwa poprzez poprawienie jej zawartości.

Dane o opcjach i ich opisach zapamiętaj w tablicy asocjacyjnej. Nazwy opcji będą kluczami, a odpowiednie wartości — wartościami w tablicy dla tych kluczy.

4-06.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Menu w formularzu</TITLE>
  </HEAD>
  <BODY>
    <? // Wyświetla formularz z menu, a następnie wyświetla wybraną przez
       // użytkownika opcję.

       $opcje = array ('pilka' => 'piłka nożna', 'kosz' => 'koszykówka',
                       'siat' => 'siatkówka', 'nar' => 'narciarstwo',
                       'hokej' => 'hokej', 'boks' => 'boks',
```

```
                    'inny' => 'inny sport');

      if ($_POST['opcja']) {    // wartości w formularzu są wpisane
        $znal = 0;
        foreach ($opcje as $klucz => $wartosc) {
          if ($klucz == $_POST['opcja']) { $cowybrano = $wartosc;
          $znal = 1; }
        }
        if (!$znal) { $cowybrano = 'niezidentyfikowana opcja'; }
        print ("Użytkownik wybrał opcję: <B>$cowybrano</B>.<BR>");
        print '<BR><A HREF="4-06.php">Powrót do formularza</A>';
      } else {    // nie ma wpisanych danych, wyświetlasz formularz
        print '<FORM ACTION="4-06.php" METHOD=POST>';
        print '<SELECT NAME="opcja">';
        print '<OPTION SELECTED VALUE="">-&gt; wybierz, jak sport
        lubisz:';
        foreach ($opcje as $klucz => $wartosc) {
          print ("<OPTION VALUE=\"$klucz\">".$wartosc);
        }
        print '</SELECT>';
        print '<INPUT TYPE="submit" VALUE="Wyślij">';
        print '</FORM>';
      }

    ?>
  </BODY>
</HTML>
```

Trzeba przyznać, że taki sposób rozwiązania problemu jest bardzo wygodny, gdyż wszelkie zmiany w treści formularza nanosisz tylko raz (w tablicy `$opcje`) i mają one skutek w całym programie, zarówno przy wyświetlaniu formularza, jak i przy jego interpretacji.

Powyższe ćwiczenie jest wstępem do utworzenia ankiety na stronie WWW. Całościowe rozwiązanie tego zadania znajdzie się w rozdziale 6.

W bardzo podobny sposób można oprogramować przyciski opcji. Spróbuj samodzielnie zmienić powyższy program tak, by wyświetlał je zamiast menu. Rozwiązanie znajduje się w pliku *4-06a.php*.

W nieco inny sposób należy zaprogramować pola wyboru, w których użytkownik może zaznaczyć więcej niż jedną opcję.

ĆWICZENIE

4.7 Pole wyboru w formularzu

Napisz program z powyższych przykładów w taki sposób, by użytkownik mógł wybrać więcej niż jedną opcję. Użyj pola wyboru.

Każde pole wyboru projektowanego formularza nazwij tak, jak nazywa się klucz z tablicy asocjacyjnej.

4-07.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Pole wyboru w formularzu</TITLE>
  </HEAD>
  <BODY>
    <? // Wyświetla formularz z opcjami wyboru, a następnie wyświetla
       // wybrane przez użytkownika opcje.

       $opcje = array ('pilka' => 'piłka nożna', 'kosz' => 'koszykówka',
                       'siat' => 'siatkówka', 'nar' => 'narciarstwo',
                       'hokej' => 'hokej', 'boks' => 'boks',
                       'inny' => 'inny_sport');

       $znal = 0;
       foreach ($opcje as $klucz => $wartosc) {
          $jest = $_GET[$klucz];
          if ($jest) { $cowybrano = $cowybrano.$wartosc." "; $znal = 1; }
       }
       if ($znal) {  // wartości w formularzu są wpisane
         print ("Użytkownik wybrał opcje: <B>$cowybrano</B>.<BR>");
         print '<BR><A HREF="4-07.php">Powrót do formularza</A>';
       } else {      // nie ma wpisanych danych, wyświetlasz formularz
         print '<FORM ACTION="4-07.php" METHOD=GET>';
         print 'Wybierz, jaki sport lubisz:<P>';

         foreach ($opcje as $klucz => $wartosc) {
           print ("<INPUT TYPE=\"checkbox\" NAME=\"$klucz\" VALUE=\"1\">");
           print ($wartosc."<BR>");
         }
         print '<P><INPUT TYPE="submit" VALUE="Wyślij">';
         print '</FORM>';
       }

    ?>
  </BODY>
</HTML>
```

Celowo użyto metody `GET`, aby w adresie można było zobaczyć, w jaki sposób są przekazywane informacje o wybranych przy użyciu pól wyboru opcjach.

W każdym przebiegu pętli `foreach` jako `$klucz` zostanie podstawiona odpowiednia nazwa klucza, dzięki czemu można odwołać się do zawartości tablicy `$_GET` i pobrać zmienną formularza o odpowiedniej nazwie.

Ze względu na to, że użytkownik może wybrać więcej niż jedną opcję menu, nie wyświetlasz tylko jednej z nich, lecz zbierasz w pętli w zmiennej `$cowybrano` wszystkie wybrane opcje.

Wyjątkową rolę w formularzach spełniają pola ukryte. Jeżeli do tej pory nie miałeś styczności z aplikacjami działającymi po stronie klienta (*client-side*), możesz wątpić w ich sens. Nic bardziej mylnego — są niezmiernie pomocne podczas przekazywania informacji pomiędzy kolejnymi formularzami przedstawianymi użytkownikowi do wypełnienia. Na pewno docenisz je po wykonaniu poniższego ćwiczenia.

ĆWICZENIE

4.8 Użycie kilku formularzy

Napisz program, który w pierwszym formularzu zbierze informacje o tym, jakim towarem jest zainteresowany użytkownik, a w drugim, jak się nazywa. Po wpisaniu wszystkich potrzebnych danych program powinien wyświetlić zebrane informacje.

Tym razem instrukcja warunkowa będzie bardziej skomplikowana — wszak musisz, w zależności od sytuacji, wyświetlić dwa formularze lub końcowy wynik. Pierwszy warunek będzie informował, że użytkownik nie wybrał jeszcze żadnego towaru, drugi — że nie podał swoich danych. Jeżeli oba będą spełnione, zostanie wyświetlona informacja o wyborze użytkownika.

4-08.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Co chciałbyś</TITLE>
  </HEAD>
  <BODY>
```

```
    <? // Pozwala na wybór towarów, a następnie na wpisanie danych
       // użytkownika. Po wpisaniu wyświetla informacje o wyborze
       // i użytkowniku. Informacje o wybranych towarach są przekazywane
       // przez ukryte pola formularza.

       if (!($_POST['zak1'] || $_POST['zak2'])) { // nie wybrano żadnego
                                                  towaru
         print '<FORM ACTION="4-08.php" METHOD=POST>';
         print '<INPUT TYPE="checkbox" NAME="zak1" VALUE="1">Towar 1<BR>';
         print '<INPUT TYPE="checkbox" NAME="zak2" VALUE="1">Towar 2<BR>';
         print '<INPUT TYPE="submit" VALUE="Wyślij"></FORM>';
       } elseif (!($_POST['imienazwisko'])) {
             // są wpisane dane o towarach, ale nie o osobie
         if ($_POST['zak1']) { print 'Wybrano towar 1<BR>'; }
         if ($_POST['zak2']) { print 'Wybrano towar 2<BR>'; }
         print '<FORM ACTION="4-08.php" METHOD=POST>';
         print "<INPUT TYPE=\"hidden\" NAME=\"zak1\" VALUE=\"".$_POST
         ['zak1']."\">";
         print "<INPUT TYPE=\"hidden\" NAME=\"zak2\" VALUE=\"".$_POST
         ['zak2']."\">";
         print 'Podaj imię i nazwisko:<BR>';
         print '<INPUT TYPE="text" NAME="imienazwisko">';
         print '<INPUT TYPE="submit" VALUE="Wyślij"></FORM>';
       } else { // masz już wszystkie dane
         if ($_POST['zak1']) { print 'Wybrano towar 1<BR>'; }
         if ($_POST['zak2']) { print 'Wybrano towar 2<BR>'; }
         print "<BR>Zamawiający: $imienazwisko<BR>";
       }

    ?>
  </BODY>
</HTML>
```

Warto zauważyć, że informacje o wybranych towarach są przekazywane pomiędzy pierwszym formularzem i stroną końcową poprzez ukryte pola, zaszyte w drugim formularzu.

Często się zdarza, że chcesz dane wpisane przez użytkownika w formularzu otrzymać pocztą elektroniczną. Jeżeli korzystasz z systemu Unix, sprawa jest łatwa — istnieje funkcja `mail`, która się tym zajmie. Wywołanie jej w postaci:

```
mail ('odbiorca@adres.pl', 'Wiadomość testowa', 'Tak działa funkcja mail',
      "From: wwwadmin@serwer.pl\r\n");
```

spowoduje posłanie na adres (będący parametrem) e-maila z tematem „Wiadomość testowa" i treścią: „Tak działa funkcja mail". Nadawcą listu będzie *wwwadmin@serwer.pl* (jako czwarty parametr można określić nagłówki listu oddzielone od siebie znakami nowego wiersza).

Jeżeli korzystasz z Krasnala, wysyłanie maili powinno od razu działać poprawnie. Natomiast jeżeli pracujesz na jakiejś innej dystrybucji, musisz uwierzyć, że ta funkcja działa poprawnie, mimo że prawdopodobnie w Twoim systemie spowoduje błąd. Umieść ją w programie tam, gdzie jest to konieczne, ale na razie w komentarzu, a jeszcze lepiej — w instrukcji warunkowej. Po przeniesieniu na linuksowy „produkcyjny" serwer wysyłanie poczty elektronicznej powinno działać bezproblemowo. Podobnie stanie się, jeżeli odpowiednio skonfigurujesz serwer Windows (albo wpisując odpowiednie IP serwera SMTP w pliku *php.ini* i będąc z nim połączonym, albo instalując serwer SMTP na własnym komputerze).

Cookies

Jeżeli już miałeś do czynienia z programowaniem po stronie serwera lub używałeś JavaScriptu w bardziej zaawansowany sposób, pewnie wiesz, czym są *cookies*. Jeżeli nie, zapewne informacja „posłałem użytkownikowi ciasteczko" wywołuje u Ciebie konsternację. Zwróć uwagę, że takie „ciasteczko" (ang. *cookie*) nie jest sposobem osłodzenia życia osobom oglądającym stronę (choć każdy twórca stron powinien ich hołubić), a jedynie prostą metodą zapamiętania pewnych danych o kliencie w jego przeglądarce.

Zasada jest prosta. O ile użytkownik wyrazi na to zgodę (decydują o tym ustawienia jego przeglądarki), masz prawo zapisać pewne dotyczące go informacje na jego dysku. Warto zapamiętać, że:

1. tylko Twój serwis będzie mógł pozyskać je z powrotem;
2. użytkownik może zabronić ich zapisu i nadzorować dowolnie ich treść.

Oba fakty (jeśli wziąć pod uwagę prawo każdego do prywatności) wydają się bardzo sensowne. Pomimo tych ograniczeń *cookies* mogą być bardzo przydatne webmasterowi. Możesz je na przykład wykorzystywać, by umożliwić osobom, które wyrażą na to zgodę, zapamiętanie ich danych przy kolejnym wypełnianiu formularza. To duże ułatwienie dla odwiedzających, a wiele osób udostępnia w ten sposób swoje dane

zaufanym serwisom, aby nie musieć wpisywać informacji o sobie wielokrotnie. Wiele serwisów stosuje *cookies*, by śledzić działania użytkowników i tworzyć jego „profil" (w różnych celach — marketingowych czy handlowych). Takie działanie jest zresztą często uważane za wątpliwe pod względem etycznym.

Jeżeli ktoś już robił zakupy w księgarni Amazon lub chociażby korzystał z jej wyszukiwarki, nie powinien się zdziwić, gdy wchodząc ponownie na jej stronę, zauważy książki, które są związane z jego zainteresowaniami. Kiedyś nawet księgarnia ta sprawdzała, czy klient korzystał już z jej oferty, czy też nie, i jako nowemu dawała niższe ceny. Dwie osoby przeglądające informacje o danej książce mogły zobaczyć ją z całkiem różnymi cenami. Wywołało to wielkie oburzenie klientów, co zaowocowało zaniechaniem wspomnianej praktyki.

Można być pewnym, że za takim działaniem na pewno kryły się *cookies*, które księgarnia wysłała użytkownikowi i wykorzystywała, by ocenić, czy jest nowym, czy starym klientem, a także jakie są jego zainteresowania.

Warto przez pewien czas poobserwować, jak i jakie informacje wysyłają różne serwisy w postaci *cookies*. Istnieje możliwość ustawienia sobie opcji informowania o ich otrzymywaniu w przeglądarce:

- w Internet Explorerze w formularzu *Opcje internetowe* w zakładce *Zabezpieczenia* i w opcji *Poziom niestandardowy* dla ustawień *cookies* włącz dwa razy opcję *Monituj*,
- w Mozilli w formularzu *Preferences* w sekcji *Advanced* w grupie *Cookies* zaznacz opcję *Warn me before accepting a cookie*.

Od tej pory, jeżeli jakiś serwis będzie próbował wysłać *cookie*, będziesz o tym informowany.

Cookie zawiera kilka informacji, a wśród nich:

- nazwę i wartość (nie mogą zawierać spacji),
- termin ważności (jeżeli nie jest podany, uznaje się, że wygasa wraz z wyłączeniem przeglądarki),
- informację `domain`, określającą serwer, którego dotyczy *cookie* (standardowo jest to ten serwer, który „ciasteczko" wysłał, ale na przykład wpisanie `.domena.pl` pozwala na dostęp do niego z domen typu *domena.pl*, *www.domena.pl*, *online.domena.pl*, *serwis.domena.pl* itd.),

- informację `path`, która określa, dla jakiego podkatalogu serwisu *cookie* ma być widoczne (zazwyczaj chcesz, aby było widoczne w całym serwisie i ustawiasz `path=/`).

Musisz mieć świadomość, że nie wszyscy użytkownicy będą odbierać Twoje *cookies*. Narosło wokół nich wiele mitów, a prasa popularna podawała nie zawsze prawdziwe informacje o zagrożeniach, które ze sobą niosą (czasem nazywając je nawet *programami*, które serwis umieszcza na dysku użytkownika!). W związku z tym istnieje grono osób, które mają wyłączone ich przyjmowanie.

Nie znaczy to jednak, że nie powinno się korzystać z *cookies*. Warto wykorzystywać je w takich zastosowaniach, w których mogą ułatwić użytkownikowi wizytę na stronie, lub które pozwolą Ci na zbieranie informacji statystycznych oraz takich, które możesz uzyskać, ale nie są koniecznie potrzebne do działania serwisu. Ciągle powinieneś jednak dbać, by ten poprawnie działał u użytkownika, który ma wyłączone przyjmowanie *cookies*.

ĆWICZENIE

4.9 Zliczanie liczby wyświetleń strony I

Napisz program, który będzie zliczał, ile razy użytkownik wyświetlił daną stronę w czasie działania jego przeglądarki.

To bardzo typowy przykład zastosowania *cookie*. Będziesz w nim przechowywać informację o liczbie wyświetleń danej strony. *Cookie* będzie nosiło nazwę `ile`, a jego wartością będzie liczba wyświetleń strony. Dostęp do informacji jest trywialny i bardzo podobny do odczytu danych z formularza. Wartość „ciastka" o nazwie `ile` jest pamiętana w specjalnej tablicy `$_COOKIE` w polu `ile`, więc możesz ją odczytać poprzez `$_COOKIE['ile']`.

W celu przesłania użytkownikowi *cookie* posłuż się poleceniem `setcookie`.

`setcookie` Posyła użytkownikowi *cookie*. Funkcja posiada sześć parametrów: pierwszy określa nazwę, drugi wartość *cookie*, trzeci to termin ważności, czwarty to informacja `path`, piąty — `domain`, a ostatni parametr to znacznik, który określa, czy *cookie* może być wysyłane jedynie przy bezpiecznym połączeniu (SSL), i domyślnie przyjmuje wartość 0.

4-09.php

```
<? // Program zlicza liczbę wyświetleń przez użytkownika danej strony
   // w czasie działania przeglądarki. Informacja jest zbierana w cookie.
   setcookie ('ile', ++$_COOKIE['ile']);
?>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Ile razy?</TITLE>
  </HEAD>
  <BODY>
    <?
       print "Stronę odwiedziłeś ".$_COOKIE['ile']." razy w czasie
       bieżącego ";
       print "uruchomienia przeglądarki";
    ?>
  </BODY>
</HTML>
```

Pomimo że funkcja `setcookie` może mieć aż sześć parametrów, możesz ją wywoływać z mniejszą ich liczbą. W naszym programie podajesz jedynie pierwsze dwa — nazwę i wartość. Pozostałe przyjmą wartości domyślne.

Powyższy skrypt nie ma zbyt wiele wspólnego ze zwykłym licznikiem — ciągle należy pamiętać, że zlicza odwiedziny każdego użytkownika oddzielnie.

Skrypt przy każdym wywołaniu odczytuje wartość *cookie* o nazwie `ile` (poprzez zmienną `$_COOKIE['ile']`) i powiększa ją o 1. Obie te operacje są wykonywane jedną instrukcją przy użyciu przedrostkowego operatora inkrementacji. Jeżeli użytkownik nie miał do tej pory takiej informacji, zostanie ona ustalona na 1. Po uaktualnieniu wartości skrypt wyświetla informację o dotychczasowej liczbie odwiedzin. Zapamiętaj, że funkcja `setcookie` musi być wywoływana zawsze przed wysłaniem czegokolwiek do przeglądarki, nawet spacji czy pustego wiersza.

Wadą programu jest to, że zlicza odwiedziny jedynie w czasie działania przeglądarki. Po jej ponownym włączeniu rozpocznie obliczenia od 1. Aby to poprawić, powinieneś ustalić termin ważności *cookie*.

ĆWICZENIE

4.10 Zliczanie liczby wyświetleń strony II

Napisz program, który będzie zliczał liczbę wyświetleń strony przez użytkownika nawet po wyłączeniu przez niego przeglądarki.

Zadanie jest bardzo proste. W celu jego realizacji użyj trzeciego argumentu funkcji `setcookie` — daty ważności „ciastka".

4-10.php

```
<? // Program zlicza liczbę wyświetleń przez użytkownika danej strony.
   // Informacja jest zbierana w cookie, którego ważność wynosi 30 dni.

   setcookie ('ile2', ++$_COOKIE['ile2'], time()+2592000);
?>
<HTML>
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Ile razy?</TITLE>
  </HEAD>
  <BODY>
    <?
       print "Stronę odwiedziłeś ostatnio ".$_COOKIE['ile2']." razy.";
    ?>
  </BODY>
</HTML>
```

Cookie celowo zostało nazwane inaczej niż w poprzednim przykładzie, aby podczas testowania nie było niejasności. Pamiętaj, że programy z obu ćwiczeń (4.9 i 4.10) mogą zobaczyć *oba* „ciastka": `ile` i `ile2`. Nie ma tu znaczenia, który z nich któremu nadał wartość.

Chcesz określić termin ważności *cookie* na miesiąc. Używasz funkcji `time`, by ustalić bieżący znacznik czasu. Obliczasz liczbę sekund w 30 dniach (60 sekund *60 *24 *30), co daje wynik 2 592 000. Ustalasz więc termin ważności, dodając tę wartość do bieżącego znacznika czasu.

W programie fakt, że termin ważności został ustawiony na miesiąc, nie oznacza, że *cookie* wygaśnie po 30 dniach. Jeżeli strona w tym czasie będzie odwiedzona, zostanie odświeżona wartość *cookie* `ile2`, z nowym (znowu miesięcznym) terminem ważności. *Cookie* straci więc swoją ważność po 30 dniach od ostatnich odwiedzin użytkownika na stronie.

Powyższe przykłady nie są w praktyce szczególnie przydatne. Spróbuj teraz wykorzystać poznaną technologię do realizacji użytecznych zadań.

ĆWICZENIE

4.11 Przechowywanie informacji w cookies

Napisz program, który wyświetla formularz pobierający imię i nazwisko użytkownika. Jeżeli użytkownik wyrazi na to zgodę, program powinien zapamiętać wpisane dane w cookies w celu uproszczenia ponownego wypełnienia formularza.

W wyświetlanym formularzu uwzględnij pole wyboru, które pozwoli użytkownikowi na decyzję, czy chce, by jego dane zostały zapamiętane, czy też nie. Każdemu użytkownikowi posyłasz *cookies* — w zależności od jego wyboru z wpisanymi wartościami lub z wartościami pustymi, by usunąć ewentualne poprzednio zapisane informacje.

4-11.php

```
<? // Drukuje formularz i jednocześnie odbiera i wyświetla wpisane
   // w nim dane. Informacje są zapisywane w cookie.

   $imie = htmlspecialchars ($_POST['imie']);
   $nazwisko = htmlspecialchars ($_POST['nazwisko']);
   $imiecookie = htmlspecialchars ($_COOKIE['imiecookie']);
   $nazwiskocookie = htmlspecialchars ($_COOKIE['nazwiskocookie']);
   if (($nazwisko) && ($imie)) { // są wpisane wartości w formularzu
      if ($_POST['pamietac']) {
        setcookie ("imiecookie", "$imie", time()+25920000);
        setcookie ("nazwiskocookie", "$nazwisko", time()+25920000);
      } else {
        setcookie ("imiecookie", "");
        setcookie ("nazwiskocookie", "");
      }
   }
?>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Pamiętasz imię i nazwisko</TITLE>
  </HEAD>
  <BODY>
    <?
```

```
        if (($nazwisko) && ($imie)) { // są wpisane wartości w formularzu
          print "Wpisana wartość to <B>$imie $nazwisko</B>.<BR>";
          print '<A HREF="4-11.php">Powrót do formularza</A>';
        } else { // nie ma wpisanych danych, wyświetlasz formularz
          print '<FORM ACTION="4-11.php" METHOD=POST>';
          print '<TABLE><TR><TD>nazwisko: </TD><TD><INPUT TYPE="text" ';
          print "NAME=\"nazwisko\" VALUE=\"$nazwiskocookie\"></TD></TR>";
          print '<TR><TD>imię: </TD><TD><INPUT TYPE="text" ';
          print "NAME=\"imie\" VALUE=\"$imiecookie\"></TD></TR></TABLE>";
          print 'Chcę, żeby program pamiętał dane: <INPUT TYPE="checkbox"';
          print 'NAME="pamietac"><BR>';
          print '<BR><INPUT TYPE="submit" VALUE="Wyślij">';
          print '</FORM>';
        }

    ?>
  </BODY>
</HTML>
```

Ponieważ odebrane dane zostaną wyświetlone na stronach WWW, po odebraniu danych z formularza przetwarzasz je funkcją `htmlspecialchars`, aby uniknąć niepożądanych efektów, takich jak w ćwiczeniu 4.1. Możesz zapytać jednak, w jakim celu użyto tej funkcji do przetworzenia wartości *cookies*? Pamiętaj, że musisz do nich podchodzić tak samo nieufnie, jak do wartości pól formularzy — użytkownik, chcąc zaszkodzić, może manipulować wartościami *cookies* podobnie, jak wartościami wpisanymi w formularzu!

Jeżeli chcesz zapamiętać jakieś dane osobowe użytkownika swojego serwisu, dobrym zwyczajem jest zapytanie go najpierw, czy rzeczywiście tego chce. Ciekawym rozwiązaniem jest to zaprezentowane w tym ćwiczeniu — w postaci pola wyboru. Być może użytkownik będzie korzystał z cudzego komputera, przeglądając serwis (np. w kawiarni internetowej). Dzięki polu w formularzu będzie mógł łatwo zadecydować, czy chce, by jego dane zostały zapamiętane na komputerze, czy też nie.

ĆWICZENIE

4.12 Losowe wyświetlanie bannera

Napisz program, który wyświetla użytkownikowi banner reklamowy (wylosowany jeden z pięciu).

Postaraj się, by — o ile to możliwe — użytkownik przy kolejnej wizycie na stronie w pierwszej kolejności oglądał te bannery, których jeszcze nie widział.

Dla każdego bannera w *cookie* zapamiętaj informację, iż był już wyświetlony danemu użytkownikowi. Pięć bannerów zamieszczono w katalogu *bannery* — w przykładach na serwerze.

4-12.php

```
<? // System wyświetlania bannerów, który stara się, by użytkownikowi
   // nie wyświetlić powtórnie bannera, dopóki nie obejrzy przynajmniej
   // raz wszystkich.

   for ($i=1; $i<=5; $i++) { // sprawdzasz, które bannery już wyświetlano
      $jest = $_COOKIE['banner'.$i];
      if (!($jest)) { $tab[] = $i; $znaleziono = 1; }
   }
   // jeżeli nie wyświetlano, włączasz do tabeli wszystkie.
   if (!($znaleziono)) { for ($i=1; $i<=5; $i++) { $tab[$i-1] = $i; }}

   $nr= $tab[rand()%count($tab)];

   setcookie ("banner".$nr, "1", time()+86400);
?>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Losowy (prawie?) banner</TITLE>
  </HEAD>
  <BODY>
    <?
       print "<IMG SRC=\"bannery/$nr.gif\">";
    ?>
  </BODY>
</HTML>
```

W pierwszej pętli `for` sprawdzasz, które bannery nie były jeszcze wyświetlane. Jeżeli taki znajdziesz, wpisz jego numer do tabeli. Warto zauważyć, że kolejna dana jest dopisana do tablicy w niespotykany jeszcze sposób: `$tab[ ]` odwołuje się do pierwszego wolnego pola w tabeli — jest to bardzo wygodna i często stosowana konstrukcja.

Jeżeli nie znaleziono żadnego bannera, który nie był jeszcze wyświetlony, uznajesz, że użytkownik widział już wszystkie. W związku z tym można mu wyświetlić dowolny z nich — więc wszystkie dopisujesz do tablicy. Następnie losujesz numer z tabeli (zwróć uwagę na sposób,

w jaki określasz liczbę jej elementów, używając funkcji `count`). Na końcu posyłasz użytkownikowi informację o wylosowanym bannerze i wyświetlasz go.

W poprzednim ćwiczeniu (przy podawaniu danych osobowych) pozostawienie odbiorcy decyzji, czy chce, by te informacje były pamiętane, miało głęboki sens. W przypadku tego ćwiczenia oczywiście takiego sensu nie ma. Niezależnie od tego, z jakiego komputera korzysta użytkownik, pozostawienie informacji, który losowy banner już widział, nie ma żadnego znaczenia.

Warto pomyśleć, jak zmodyfikować program, by przesyłał tylko jedno *cookie* informacyjne, a nie inne dla każdego bannera. Jeden serwis może posłać danemu użytkownikowi tylko niewielką liczbę „ciasteczek", więc powinieneś je szanować. Spróbuj zmodyfikować program tak, by w jednym przesyłanym *cookie* informacje o wyświetlonych bannerach były oddzielone przecinkami.

Należy zauważyć, że w obu powyższych praktycznych przykładach skrypty będą działały poprawnie, nawet jeżeli użytkownik nie odbierze „ciasteczek". Pierwszy po prostu nie podpowie mu jego danych w formularzu, jednak poprawnie go wyświetli i odbierze wpisane dane. Drugi program w przypadku braku informacji będzie po prostu wyświetlał losowy banner. W żadnym wypadku programy nie stracą jednak swojej funkcjonalności.

Obsługa plików

Do tej pory dane dla tworzonych przez Ciebie programów były wpisywane z klawiatury, a wyniki wypisywane na ekranie. Przydałaby się jednak także metoda pamiętania informacji w jakiś trwalszy — i łatwy do przekazania programowi — sposób. Taki warunek spełniają pliki dyskowe. PHP potrafi zapisywać i odczytywać pliki w systemie operacyjnym serwera. Należy podkreślić jeszcze raz: jesteś w stanie (zgodnie z uprawnieniami, które zostaną tam nadane odpowiednim katalogom i plikom) tworzyć, zapisywać, odczytywać i usuwać pliki na serwerze. Na serwerze, a nie na komputerze użytkownika.

Zapis i odczyt plików może być niezwykle przydatny do zapamiętania niewielkich ilości informacji zarówno dla całego serwisu, jak i dla pojedynczego użytkownika. Tak naprawdę mechanizm sesji przedstawiony

w kolejnym punkcie opiera się właśnie na plikach tekstowych. Jeżeli będziesz chciał pamiętać więcej informacji, lepszą metodą okaże się skorzystanie z bazy danych (punkt 4.5), ale jeżeli jest ich niewiele, można użyć plików tekstowych.

Najprostsza operacja na pliku składa się z trzech etapów: otwarcia pliku, zapisu lub odczytu z niego danych i zamknięcia pliku.

ĆWICZENIE

4.13 Tworzenie pliku i zapis danych

Napisz program, który w podkatalogu dane utworzy plik z Twoim imieniem i nazwiskiem.

Na początku należy utworzyć katalog *dane*, w którym powstanie plik. Jeżeli korzystasz z systemu Windows, sprawa jest prosta: po prostu utwórz go jako podkatalog tego, w którym przechowujesz pliki PHP. Jeżeli natomiast korzystasz z serwera linuksowego, musisz jeszcze nadać odpowiednie prawa katalogowi, w celu umożliwienia wszystkim zapisu. Pozwoli na to program FTP, którego używasz (jeżeli nie, koniecznie go zmień!). Ustaw dla katalogu prawa na 666.

Wykorzystaj trzy funkcje operujące na plikach.

`fopen` Otwarcie pliku o nazwie określonej pierwszym parametrem, w trybie określonym drugim parametrem. Na przykład `r` oznacza otwarcie do odczytu, a `w` — do zapisu. Funkcja zwraca uchwyt pliku, poprzez który możesz się do niego odwoływać.

`fputs` Pozwala na zapis tekstu, który jest drugim parametrem, do pliku określonego uchwytem w pierwszym parametrze.

`fclose` Zamyka wskazany przez uchwyt plik.

4-13.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Zapisujesz plik</TITLE>
  </HEAD>
  <BODY>
    <? // Program tworzy plik tekstowy z imieniem i nazwiskiem.
```

```
        $plik = @fopen ("dane/imienazwisko.txt", "w");
        if (!($plik)) {
          print "BŁĄD: Nie da się utworzyć pliku.";
        } else {
          print "Plik z imieniem i nazwiskiem został utworzony.";
          fputs ($plik, "Andrzej Kierzkowski");
          fclose ($plik);
        }
    ?>
  </BODY>
</HTML>
```

Zwróć uwagę na znak `@` przed wywołaniem `fopen` — dzięki niemu informujesz PHP, że ewentualne błędy w wykonaniu nie mają być wysyłane do przeglądarki. Po próbie otwarcia pliku dobrym zwyczajem jest zweryfikowanie, czy operacja się powiodła. Czynisz to poprzez sprawdzenie, czy uchwyt (zmienna `$plik`) ma ustaloną wartość. Jeżeli pracujesz w systemie Windows, a program zakończył działanie błędem, najprawdopodobniej nie założyłeś katalogu *dane* lub założyłeś go w złym miejscu (powinien być podkatalogiem tego, w którym znajduje się program z bieżącego ćwiczenia). Jeżeli program zakończył działanie pomyślnie, poszukaj utworzonego pliku i sprawdź jego zawartość.

ĆWICZENIE

4.14 Odczyt danych z pliku

Napisz program, który odczyta utworzony w poprzednim ćwiczeniu plik i wyświetli jego zawartość.

Tym razem otworzysz plik w trybie „do odczytu”. Użyj funkcji, która odczytuje dane z pliku.

`fgets` Odczytuje z pliku wskazanego przez pierwszy argument liczbę znaków w bieżącym wierszu, określoną przez drugi, opcjonalny, argument.

4-14.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Odczytywanie pliku</TITLE>
  </HEAD>
```

```
<BODY>
  <? // Program odczytuje plik tekstowy z imieniem i nazwiskiem.

     $plik = @fopen ("dane/imienazwisko.txt", "r");
     if (!($plik)) {
       print "BŁĄD: Nie da się otworzyć pliku.";
     } else {
       $wiersz = fgets ($plik, 255);
       print "Dane z pliku: <B>$wiersz</B>";
       fclose ($plik);
     }
  ?>
</BODY>
</HTML>
```

Aby sprawdzić, czy plik istnieje, możesz wykorzystać w tym celu funkcję `file_exist`.

`file_exists` Zwraca prawdę, gdy istnieje plik, którego nazwę podaje się jako parametr, a fałsz — w przeciwnym razie.

Pomyśl, jak poprawić program, by najpierw sprawdzał, czy plik istnieje, a dopiero potem go otwierał.

Ciekawy jest fakt, że za pomocą funkcji z poprzedniego ćwiczenia możesz odczytywać pliki nie tylko z serwera, ale także inne dostępne w internecie za pośrednictwem protokołów HTTP i FTP.

ĆWICZENIE

4.15 Zmiana sposobu wyświetlania odczytanej strony WWW

Napisz program, który odczyta zawartość głównej strony serwisu http://helion.pl i wyświetli ją, zamieniając najpierw kolor tła z niebieskawego na zielony.

W celu wykonania tego ćwiczenia serwer, na którym uruchamiasz programy PHP, musi być podłączony do internetu, aby program mógł odczytać stronę Helionu.

4-15.php

```
<? // Program odczytuje stronę helion.pl i wyświetla, podmieniając kolory.

  $plik = fopen ("http://helion.pl", "r");
  if (!($plik)) {
    print "BŁĄD: Nie da się otworzyć strony http://helion.pl";
  } else {
    while (!(feof($plik))) {
      $wiersz = (fgets ($plik, 255));
      $wiersz = str_replace ('#E4E7ED', '#A2FFB5', $wiersz);
      $wiersz = str_replace (' SRC="', ' SRC="http://helion.pl', $wiersz);
      print "$wiersz";
    }
    fclose ($plik);
  }
?>
```

Nie wypisujesz nagłówków HTML, licząc na to, że strona ma swoje. Po odczytaniu przetwarzasz plik wiersz po wierszu, używając funkcji `feof`.

`feof` Zwraca prawdę, gdy osiągnięto koniec pliku, określonego pierwszym parametrem, a fałsz — w przeciwnym razie.

W każdym wierszu dokonujesz dwóch operacji: po pierwsze — zamieniasz kolor z niebieskawego (`#E4E7ED`) na zielony (`#A2FFB5`) a po drugie — poprawiasz adresy wszystkich ilustracji, by zawierały również adres serwera (w innym przypadku nie byłoby ich widać, ponieważ program szukałby ich na Twoim serwerze).

Efekt jest… ciekawy, choć niebieski kolor komponuje się znacznie lepiej. Zauważ, że układ strony się nieco „rozsypał”. Spróbuj odgadnąć, co jest tego przyczyną. Przyjrzyj się, jak są odczytywane pliki z definicjami styli.

W praktyce najczęściej pliki pamiętane na dysku będą wspólne dla wszystkich odwiedzających — znajdą się w nich informacje, którymi będą się oni mogli dzielić. Istnieje w związku z tym ryzyko, że dwie osoby jednocześnie będą pracować na jednym pliku, co może spowodować niemałe kłopoty.

Wyobraź sobie sytuację, gdy w pliku jest pamiętany aktualny stan licznika odwiedzin danej strony. Sam licznik działa w taki sposób, że odczytuje wartość z tego pliku, zwiększa ją o 1 i zapisuje z powrotem (i oczywiście wyświetla wartość odwiedzającemu). Nieszczęśliwy przypadek może spowodować, że dwie osoby niemalże jednocześnie uruchamiają program. Działania mogą przybrać opisany niżej, niekorzystny dla Ciebie przebieg.

1. Pierwszy użytkownik odczyta wartość z pliku.
2. Drugi użytkownik odczyta *tę samą* wartość z pliku.
3. Pierwszy użytkownik zwiększy wartość o 1 i zapisze.
4. Drugi użytkownik zwiększy wartość o 1 i zapisze.

Jaki będzie efekt? Stronę odwiedziły dwie osoby, a wskazanie licznika wzrośnie tylko o 1! Stanie się tak dlatego, że dwie osoby będą miały dostęp do pliku w jednym czasie, podczas gdy odczyt, zwiększenie i zapisanie wartości z powrotem powinno się odbyć w czasie, gdy inne osoby nie mogą nawet odczytać wartości licznika.

Jest to możliwe do zrealizowania poprzez zablokowanie pliku na czas trwania operacji.

`flock` Ustawia blokadę pliku, którego uchwyt jest pierwszym argumentem w trybie określonym przez drugi. Na przykład tryb 2 oznacza blokowanie na wyłączność, a tryb 3 — odblokowanie.

ĆWICZENIE

4.16 Prosty licznik tekstowy

Napisz program, realizujący prosty licznik tekstowy, którego wskazanie jest pamiętane w pliku tekstowym. Pamiętaj o blokowaniu pliku.

Plik ze wskazaniem licznika zapamiętaj w katalogu *dane*. Program sprawdzi najpierw, czy plik licznika już istnieje, a jeśli takiego pliku nie ma, zajmie się jego utworzeniem.

4-16.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Licznik</TITLE>
  </HEAD><BODY>
```

```
    <? // Program pokazuje działanie prostego licznika tekstowego.

       if (!(file_exists("dane/liczba.txt"))) { // nie ma pliku z licznikiem
         $plik = fopen ("dane/liczba.txt", "w+");
         fputs ($plik, "0");                    // więc go tworzysz
         fclose ($plik);
       }

       $plik = fopen ("dane/liczba.txt", "r+");
       if (!($plik)) {
         print "BŁĄD: Nie da się otworzyć pliku.";
       } else {
         flock ($plik, 2);
         $ile = fgets ($plik, 255);
         $ile++;
         print "Licznik wskazuje $ile.";
         fseek ($plik, 0);
         fputs ($plik, "$ile");
         flock ($plik, 3);
         fclose ($plik);
       }
    ?>
  </BODY>
</HTML>
```

Jeżeli plik nie istnieje, otwierasz go w trybie `w+` (tryb ten pozwala na otwarcie istniejącego pliku do zapisu, uprzednio usuwając jego zawartość, a jeśli nie istnieje — tworzy go). Jeżeli plik istnieje, otwierasz go w trybie `r+`, co pozwala na czytanie i zapis do niego.

Po odczytaniu danych wskaźnik przesunął się poza liczbę odwiedzin. Jeżeli wpisałbyś nową, zostałaby dopisana na końcu pliku, po pierwotnej. Aby wpisać ją znów we właściwym miejscu, należy przesunąć się z powrotem do początku pliku. W tym celu użyto funkcji `fseek`.

`fseek` Przesuwa wskaźnik położenia w pliku, którego uchwyt jest pierwszym argumentem, w miejsce wskazane przez drugi.

Warto zauważyć, że blokada pliku nastąpiła po jego otwarciu (musi tak być, inaczej zmienna plikowa nie byłaby określona), a odblokowanie — przed zamknięciem. W czasie istnienia blokady pliku dokonujesz na nim wszystkich operacji: odczytu, przesunięcia wskaźnika i zapisu.

W ćwiczeniach przedstawionych do tej pory pliki umieszczano w podkatalogu tego, w którym występują dokumenty PHP, co oznacza, że znajdują się one w drzewie katalogów serwera WWW. Nietrudno przewidzieć, co to oznacza. Zamiast *http://localhost/cwphp/4-14.php* wystarczy w przeglądarce wpisać adres *http://localhost/cwphp/dane/imienazwisko.txt*, a wówczas to, co być może miało być ukryte, stanie się całkiem jawne. Niejednokrotnie poważne serwisy postępują w taki sposób, ujawniając nawet dane dotyczące swoich klientów. Jeżeli koniecznie chcesz przechowywać pliki z informacjami w strukturze katalogów serwisu, koniecznie zabezpiecz je metodą, którą poznałeś w ćwiczeniu 2.4. Pamiętaj jednak, że nie jest to zabezpieczenie bardzo pewne, dlatego dużo lepiej jest umieszczać pliki *poza* strukturą serwisu. Możesz też utworzyć w katalogu plik *.htaccess* i umieścić w nim polecenie `Deny from all`, które uniemożliwia odczytanie za pomocą serwera plików leżących wewnątrz. Warto też zwrócić uwagę, że masz dostęp do całego dysku serwera, który jest ograniczony jedynie nadanymi prawami do katalogów. Aby odwoływać się do pliku, który znajduje się poza strukturą serwisu, najlepiej używaj ścieżki bezwzględnej (zaczynającej się od katalogu głównego struktury plików). Możesz więc próbować tworzyć pliki o nazwach */home/httpd/aki/mojplik.txt* (o ile nadałeś katalogowi *aki* odpowiednie prawa) czy *C:\usr\apache\plik.txt.*

Pomimo faktu, że pliki dyskowe zazwyczaj tworzy się po to, by wymieniać informacje pomiędzy poszczególnymi użytkownikami, jest możliwe łatwe utworzenie unikatowego pliku tymczasowego dla jednego użytkownika. W tym celu wykorzystuje się funkcję `tempnam`.

`tempnam` Tworzy plik o unikatowej nazwie w katalogu określonym pierwszym parametrem. Początek nazwy można określić drugim parametrem. Plik zwraca nazwę utworzonego pliku.

ĆWICZENIE

4.17 Przechowywanie informacji w pliku tymczasowym

Napisz program, który będzie zbierał informacje o użytkowniku, podobnie jak ten z ćwiczenia 4.8.

Dane niech będą jednak zbierane w pliku tymczasowym, którego nazwa będzie przekazywana w ukrytym polu.

Bezpieczną lokalizacją dla plików tymczasowych w systemie Windows jest *C:\WINDOWS\ TEMP*, a w Linuksie — katalog */tmp*. Jeżeli w wywołaniu funkcji `tempnam` podasz nazwę nieistniejącego katalogu, plik zostanie założony w domyślnym katalogu plików tymczasowych w systemie.

4-17.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Co chciałbyś -- 2</TITLE>
  </HEAD><BODY>
    <? // Pozwala na wybór towarów, a następnie na wpisanie danych
       // użytkownika. Po wpisaniu wyświetla informacje o wyborze
       // i użytkowniku. Informacje o wybranych towarach są
       // przechowywane w pliku tekstowym.

       if (!($_POST['nazwapliku'] || $_POST['zak1'] || $_POST['zak2'])) {
       // nie ma danych
         print '<FORM ACTION="4-17.php" METHOD=POST>';
         print '<INPUT TYPE="checkbox" NAME="zak1" VALUE="1">Towar 1<BR>';
         print '<INPUT TYPE="checkbox" NAME="zak2" VALUE="1">Towar 2<BR>';
         print '<INPUT TYPE="submit" VALUE="Wyślij">';
         print '</FORM>';
       } else {
         if (($_POST['zak1']) || ($_POST['zak2'])) { // wybrano towar
                                                     w poprzednim
                                                     formularzu
           $nazwapliku = tempnam ('C:\WINDOWS\TEMP', '');
           $plik = fopen ($nazwapliku, "w");
           fputs ($plik, $_POST['zak1']."\n");
           fputs ($plik, $_POST['zak2']."\n");
           fclose ($plik);
           print '<FORM ACTION="4-17.php" METHOD=POST>';
           print "<INPUT TYPE=\"hidden\" NAME=\"nazwapliku\" ";
           print "VALUE=\"$nazwapliku\">";
           print 'Podaj imię i nazwisko:<BR>';
           print '<INPUT TYPE="text" NAME="imienazwisko">';
           print '<INPUT TYPE="submit" VALUE="Wyślij">';
           print '</FORM>';
         } else { // masz już wszystkie dane
           $plik = fopen ($_POST['nazwapliku'], "r");
           $zak1 = fgets ($plik, 255);
           $zak2 = fgets ($plik, 255);
           fclose ($plik);
           if ($zak1 > 0) { print 'Wybrano towar 1<BR>'; }
           if ($zak2 > 0) { print 'Wybrano towar 2<BR>'; }
```

```
            print "<BR>Zamawiający: ".$ _POST['imienazwisko']."<BR>";
            unlink ($_POST['nazwapliku']);
          }
        }
    ?>
  </BODY>
</HTML>
```

Jak widzisz, w ukrytym polu formularza była przenoszona jedynie informacja o nazwie pliku. Wybrane przez użytkownika towary przechowano w pliku tymczasowym. Aby „posprzątać" po sobie i usunąć niepotrzebny już plik, użyłeś funkcji `unlink`.

`unlink` Usuwa plik o nazwie podanej jako parametr.

Istnieje możliwość wczytania całej zawartości pliku do tablicy za pomocą pojedynczej instrukcji. Wtedy każdy wiersz pliku zostanie zamieszczony w oddzielnym polu tablicy.

`file` Wczytuje całą zawartość pliku o nazwie podanej jako parametr.

Trzeba uważać, by nie przeprowadzać w ten sposób operacji na dużych plikach, gdyż może się to zakończyć katastrofą. Dla małych plików jest to jednak bardzo wygodna metoda.

ĆWICZENIE

4.18 Losowe przysłowie

Napisz program, który będzie wyświetlał jedno wylosowane przysłowie z wielu zapamiętanych w pliku tekstowym.

Zacznij od utworzenia pliku tekstowego z przysłowiami. Każde zamieść w osobnym wierszu.

4-18.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Odczytywanie pliku</TITLE>
  </HEAD><BODY>
    <? // Program odczytuje plik tekstowy i wyświetla losowe przysłowie.
```

```
        $przyslowia = file ("dane/przyslowia.txt", false);
        $przyslowie = chop($przyslowia[rand()%count($przyslowia)]);
        print "Przysłowie na dziś: <B>\"$przyslowie\"</B>";
    ?>
  </BODY>
</HTML>
```

Program wczytuje cały plik z przysłowiami do tablicy. Następnie przeprowadzasz losowanie przysłowia (uprzednio poznawszy rozmiar tablicy). Z wylosowanego przysłowia odcinasz znak końca wiersza (używając funkcji `chop`) i wyświetlasz je.

Pliki rozważnie stosowane przez webmastera mogą stać się w jego rękach potężnym narzędziem. Powinien on jednak ciągle brać pod uwagę, jak dużo danych może znajdować się w nich po pewnym czasie działania serwisu i już na etapie projektowania rozważyć możliwość pamiętania informacji w bazie danych (dowiesz się o nich w punkcie 4.5). Czas poświęcony na naukę tej technologii i dobre zaprojektowanie bazy danych szybko zwróci się dzięki ułatwieniu operowania danymi oraz niezawodności i stabilności, którą to zapewni.

Nie należy jednak popadać w przesadę. Dobrze jest rozważyć, czy newsy zamieszczane na stronie (szczególnie, jeżeli przewidujesz, że codziennie będzie dochodziło kilka nowych) umieścić w bazie danych. Jednak do implementacji licznika dla jednej strony (trzeba pamiętać tylko jedną liczbę) wykorzystanie plików jest zupełnie wystarczające.

Sesje

Jednym z podstawowych problemów, z którymi musi zmierzyć się webmaster w swojej praktyce, jest przenoszenie informacji dotyczących konkretnego użytkownika pomiędzy stronami serwisu. Dobrym przykładem jest budowa sklepu internetowego, w którym z każdym użytkownikiem jest związany koszyk z zamówionymi przez niego towarami. Niezależnie od tego, w którym miejscu serwisu znajduje się osoba, serwis powinien być w stanie wyświetlić na żądanie jego zawartość, umożliwić dalsze zakupy czy przejście do wirtualnej kasy.

Protokół HTTP, na którym opiera się działanie WWW, nie zapewnia takiej możliwości. Każde żądanie wyświetlenia strony stanowi spójną całość i nie ma tu możliwości pamiętania identyfikatora użytkownika (dopóki nie będziesz żądać logowania się na stronach, ale taka metoda raczej nie jest spotykana).

Przy użyciu poznanych w poprzednich punktach technik, a więc formularzy, przekazywania parametrów metodą `GET`, *cookies* oraz dzięki wykorzystaniu plików tekstowych możesz samodzielnie skonstruować metodę identyfikacji użytkowników na stronach serwisu.

Musisz oprzeć się na tym, że ta część użytkowników, która akceptuje „ciastka", będzie identyfikowana za ich pomocą. Wystarczy, że każdemu użytkownikowi na początku będziesz podsyłał w postaci *cookie* unikatowy identyfikator. Można w tym celu wykorzystać jeden z bardziej popularnych modułów serwera Apache — *mod_usertrack*. W tym celu musisz go dołączyć, usuwając komentarz z wiersza:

```
LoadModule usertrack_module modules/ApacheModuleUserTrack.dll
```

w pliku *httpd.conf* w katalogu z plikami konfiguracyjnymi Apache'a (jeżeli nie korzystasz z Krasnala, wiersz może mieć nieco inną postać). W tym samym pliku należy też dodać dyrektywę:

```
CookieTracking On
```

(wstaw ją na jego końcu). Od tej pory, po ponownym uruchomieniu, sam serwer będzie próbował posłać każdemu użytkownikowi unikatowe *cookie* o nazwie `Apache`, w którym będziesz mógł pamiętać jego dane.

Włącz w przeglądarce informowanie o wysyłanych *cookies* i sprawdź działanie tej metody.

Innym sposobem może być wykorzystanie funkcji PHP `uniqid`, za pomocą której można wygenerować na podstawie zegara systemowego losowy identyfikator i przesłać go w postaci *cookie*. Nie jest dobrym pomysłem korzystanie z funkcji `tempnam`, ponieważ generowane przez nią unikatowe identyfikatory (stanowiące nazwy plików tymczasowych) nie są losowe, co może powodować próby podszywania się pod innego użytkownika.

W przypadku osób, które akceptują *cookies*, masz prostą sprawę — możesz w plikach na serwerze pamiętać dotyczące ich informacje. Na każdej stronie będziesz mógł, po odczytaniu „ciastka", zidentyfikować użytkownika i odczytać na serwerze dotyczące go dane.

Sprawa się komplikuje, jeżeli użytkownik nie akceptuje *cookies*. Wiele serwisów w takim przypadku odmawia możliwości korzystania z tych funkcji, które wymagają identyfikacji użytkownika, wyświetlając prosty komunikat:

Twoja przeglądarka nie akceptuje cookies, na których opiera się działanie tego serwisu. Jeżeli korzystasz z przeglądarki… — i tu następuje szczegółowa informacja, jak w każdej z przeglądarek włączyć *cookies*. Jeżeli jednak konsekwentnie budujesz swój serwis, możesz pokusić się o to, by przekazywać pomiędzy jego stronami informację w inny sposób. Jednym z nich jest przekazywanie parametrów pomiędzy stronami metodą `GET`. Jeżeli każda wyświetlana strona jest generowana przez skrypt PHP, wystarczy, że będziesz używał adresów nie w postaci:

http://serwer.com.pl/skrypt.php, lecz:

http://serwer.com.pl/skrypt.php?id=identyfikatror,

i w każdym skrypcie, z dokładnością księgowego, odczytywał wartość zmiennej `$id` oraz przenosił identyfikator na dalsze strony. Może się jednak zdarzyć, że w jednym połączeniu w serwisie zapomnisz przekazać identyfikator, a wówczas użytkownik, który nim podąży, zgubi swoją tożsamość, a wraz z nią powiązanie z całą zawartością koszyka.

Istnieją również inne metody przekazywania informacji o identyfikatorze użytkownika. Niektóre z nich wymagają większego wkładu pracy przy konfiguracji serwera, a mniejszej od programisty serwisu.

Niesamowitym ułatwieniem w przenoszeniu informacji o użytkowniku pomiędzy stronami serwisu są wprowadzone w PHP w wersji 4. *sesje*. Jest to mechanizm, który spełnia dwie funkcje: po pierwsze — identyfikacji użytkownika, a po drugie — pamiętania dotyczących go informacji. Mechanizm sesji w rzeczywistości wykorzystuje połączenie *cookies* oraz wykorzystanie metody `GET` i pamiętanie danych w plikach tekstowych. Jeżeli użytkownik strony przyjmuje *cookies*, to są one wykorzystywane do pamiętania identyfikatora. Jeżeli nie, do jego przekazywania pomiędzy stronami stosuje się metodę `GET`.

Najważniejszy jest jednak fakt, że projektanta strony tak naprawdę nie interesuje, która metoda jest wykorzystywana przez danego użytkownika — odpowiednie działanie zapewnia mu mechanizm sesji. On operuje na bardziej abstrakcyjnym poziomie, przez co pisanie większego serwisu aplikacji może stać się przyjemnością.

Przed rozpoczęciem pracy z sesjami sprawdź w pliku *php.ini* wartość opcji `session.save_path`. Określa ona katalog, w którym przechowuje się pliki tymczasowe, dotyczące sesji. Jeżeli korzystasz z Krasnala, domyślnie wskazuje ona katalog *c:\usr\php5\sesje*.

ĆWICZENIE

4.19 Korzystanie z mechanizmu sesji I

Utwórz dwa dokumenty, które będą korzystać z mechanizmu sesji i wyświetlać identyfikator użytkownika.

Nawigację pomiędzy dokumentami umożliwi utworzenie połączeń. Sprawdź działanie sesji z włączoną i wyłączoną obsługą cookies.

Oba dokumenty będą mały identyczną treść:

4-19.php, 4-19a.php

```
<?
  session_start();
?>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Sesje</TITLE>
  </HEAD>
  <BODY>
    <?
       print session_id();

    ?>
    <BR><A HREF="4-19.php">4-19</A>, <A HREF="4-19a.php">4-19a</A>
  </BODY>
</HTML>
```

Na początku wykorzystaj funkcję `session_start`.

`session_start` Inicjacja mechanizmu sesji dla danego użytkownika.

Dzięki jej wywołaniu PHP rozpoczyna dla Twoich potrzeb identyfikację sesji użytkownika (czyli otwiera jego sesję). Jest mu przypisywany unikatowy identyfikator, który można odczytać funkcją `session_id`.

`session_id` Zwraca identyfikator sesji użytkownika.

Tu już wszystko — na końcu dokumentu znajdziesz jeszcze dwa połączenia do obu stron.

Przetestuj najpierw działanie mechanizmu sesji z wyłączoną obsługą `cookies`. Zwróć uwagę na adresy połączeń. Na końcu każdego adresu został automatycznie dopisany numer sesji jako parametr — to właśnie dzięki temu na każdej ze stron wyświetlony identyfikator sesji jest taki sam.

Włącz teraz obsługę „ciastek", ale zachowaj ostrzeganie o ich otrzymywaniu. Przy pierwszym oglądaniu dowolnej ze stron otrzymasz „przesyłkę" — identyfikator. Zaakceptuj to `cookie`. Zauważ, że teraz do połączeń nie dopisano parametru. Okazuje się, że nie jest on potrzebny, ponieważ jako użytkownik jesteś już rozpoznany w inny sposób.

Na dowolnej stronie serwisu, po wywołaniu funkcji `session_start`, będziesz w stanie korzystać z mechanizmu sesji.

Łatwe i przyjemne, prawda?

Identyfikacja użytkownika to nie wszystko, co sesje mają nam do zaoferowania. Oprócz tego bardzo łatwo można dla danego użytkownika pamiętać pewne istotne wartości i mieć do nich dostęp z dowolnej strony serwisu.

ĆWICZENIE

4.20 Korzystanie z mechanizmu sesji II

Popraw przykłady z poprzedniego ćwiczenia tak, by dokumenty wyświetlały również łączną liczbę odwiedzin danego użytkownika na obu stronach w ciągu danej sesji.

Oba dokumenty będą znów mały identyczną treść:

4-20.php, 4-20a.php

```
<?
   session_start();
   print session_id();

   if (isset($_SESSION['licznik'])) {
      $_SESSION['licznik']++;
   } else {
      $_SESSION['licznik'] = 1;
   }
```

```
?>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Sesje i zmienne</TITLE>
  </HEAD>
  <BODY>
     <?
       print "<P>Licznik: <B>".$_SESSION['licznik']."</B></P>";
    ?>
    <BR><A HREF="4-20.php">4-20</A>, <A HREF="4-20a.php">4-20a</A>
  </BODY>
</HTML>
```

W celu sprawdzenia, czy licznik już został zainicjalizowany, wykorzystaj funkcję `isset`. Jeżeli jest to potrzebne, zainicjuj licznik wartością 1, a w przeciwnym razie — zainkrementuj go.

`isset` Sprawdza, czy zmiennej przypisano już jakąś wartość

Zwróć uwagę na tablicę `$_SESSION` — przypisanie wartości któremuś z jej pól powoduje zapamiętanie wartości zmiennej dla danej sesji. Po odczytaniu dowolnego dokumentu serwisu, jeżeli użytkownik zostanie zidentyfikowany, wartość zmiennej będzie widoczna. Oczywiście dla jednej sesji możesz zapamiętać wiele zmiennych (każdą umieszczasz w tablicy `$_SESSION` pod osobnym kluczem).

Warto zauważyć, że działa to zarówno przy wyłączonej, jak i włączonej obsłudze *cookies*. Pamiętaj, że jeżeli uruchamiałeś już programy z poprzednich ćwiczeń — z ich włączonym przyjmowaniem, nie wystarczy ustawić odpowiedniej opcji w przeglądarce — wszak Twoja sesja już jest zarejestrowana i przeglądarka zaakceptowała „ciastko" wcześniej. Musisz jeszcze wyłączyć i włączyć przeglądarkę — domyślnie „sesyjne" *cookies* mają czas życia do momentu jej wyłączenia.

Zapoznaj się z zawartością katalogu, który określiłeś przy konfigurowaniu jako miejsce, gdzie będą pamiętane pliki tymczasowe dla sesji (w Twoim przypadku domyślnie będzie to katalog *c:\usr\php5\sesje*). Znajdziesz w nim pliki, których nazwa będzie rozpoczynała się znakami *sess_* — w nich są pamiętane informacje o sesjach. Odnajdź wśród nich ostatnio stworzony plik i zobacz, w jaki sposób są pamiętane wartości zmiennych dla sesji.

Łatwo zauważyć, jakie potężne możliwości daje wykorzystanie sesji, jednak warto zadać pytanie, na co należy uważać przy ich stosowaniu. Poniższy przykład pokaże Ci potencjalne zagrożenie.

ĆWICZENIE

4.21 Wykorzystywanie plików HTML

Sprawdź, co się stanie, jeżeli „wyjdziesz" poza PHP, wykorzystując w serwisie zwykły plik HTML.

Wykorzystasz podobne pliki, jak w poprzednim ćwiczeniu, jednak połączysz się też z jednym prostym plikiem HTML.

4-21.php, 4-21a.php

```
<?
   session_start();
   print session_id();

   if (isset($_SESSION['licznik'])) {
      $_SESSION['licznik']++;
   } else {
      $_SESSION['licznik'] = 1;
   }
?>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Sesje, zmienne i plik HTML</TITLE>
  </HEAD>
  <BODY>
    <?
       print "<P>Licznik: <B>".$_SESSION['licznik']."</B></P>";
            ?>
    <BR><A HREF="4-21.php">4-21</A>, <A HREF="4-21a.php">4-21a</A>,
        <A HREF="4-21b.htm">4-21b -- UWAGA!!!</A>
  </BODY>
</HTML>
```

Oto bardzo prosta treść pliku HTML:

4-21b.htm

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
```

```
    <TITLE>Sesje, zmienne i plik HTML</TITLE>
  </HEAD>
  <BODY>
    Plik HTML. Jeżeli masz wyłączone cookies, zgubisz tu sesję :(
    <BR><A HREF="4-21.php">4-21</A>, <A HREF="4-21a.php">4-21a</A>,
         <A HREF="4-21b.htm">4-21b -- UWAGA!!!</A>
  </BODY>
</HTML>
```

Przed przetestowaniem **koniecznie wyłącz** obsługę *cookies*. W przeglądarce Internet Explorer konfigurację *cookies* znajdziesz po kliknięciu menu *Narzędzia*/*Opcje internetowe* i wybraniu zakładki *Prywatność*. Cóż się dzieje? Dopóki poruszasz się w obrębie skryptów PHP, wszystko działa wspaniale. Wystarczy jednak przejść połączeniem do pliku HTML-owego i… gubisz wszystkie informacje o użytkowniku. Z adresów stron PHP znika parametr — numer sesji. Oczywiście w przypadku, gdy użytkownik akceptuje *cookies*, w dokumentach PHP zostanie poprawnie rozpoznany. Wykorzystując sesje, musisz jednak pamiętać, że nie każdy korzysta z dobrodziejstw „ciastek" i stosując zwykłe dokumenty HTML możesz mieć kłopot.

W rzeczywistości gubiłbyś dane użytkownika także w każdym skrypcie PHP, który nie używa funkcji `session_start`. Musisz więc albo konsekwentnie ją stosować wszędzie, albo po prostu w konfiguracji (w pliku *php.ini*) ustawić wartość `session.auto_start` na 1:

```
session.auto_start        = 1
```

Pojawia się pytanie, co się dzieje w przypadku przekazywania informacji o sesji w postaci formularzy. Odpowiedź może być zaskakująca. Nawet jeżeli użytkownik nie akceptuje *cookies*, do formularzy są dodawane automatycznie ukryte pola, które przenoszą informację o sesji.

ĆWICZENIE

4.22 Wykorzystywanie metod: POST i GET

Utwórz dokumenty testowe z formularzami. Wykorzystaj obie metody: POST i GET. Sprawdź, jak działa przekazywanie informacji o sesji.

Oba skrypty są bardzo podobne, a różnią się jedynie metodą przekazywania danych POST i GET, a także tablicą, z której dane są odczytywane z formularza ($_POST i $_GET).

4-22.php

```
<?
   session_start();
   print session_id();

   if (isset($_SESSION['licznik'])) { $_SESSION['licznik']++; } else {
   $_SESSION['licznik'] = 1; }
?>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Sesje i formularze - metoda POST</TITLE>
  </HEAD>
  <BODY>
    <?
       print "<P>Licznik: <B>".$_SESSION['licznik']."</B></P>";
       if ($imie) { print "<P>Wpisane imię: <B>".$_POST['imie'].
       "</B></P>"; }
    ?>

    <FORM ACTION="4-22.php" METHOD="POST">
    Wpisz imię: <INPUT TYPE="text" NAME="imie">
    <INPUT TYPE=SUBMIT VALUE="Wyślij">
    </FORM>

    <BR><A HREF="4-22.php">4-22 - Formularz z metodą POST</A>
    <BR><A HREF="4-22a.php">4-22a - Formularz z metodą GET</A>

  </BODY>
</HTML>
```

W przypadku, gdy przeglądarka użytkownika akceptuje *cookies*, nie ma problemu. Wyłącz jednak ich obsługę i sprawdź działanie obu formularzy. Jeżeli podejrzysz źródło, zauważysz, że w metodzie POST automatycznie dodano w nich pola ukryte, przekazujące informacje o sesji.

Wynika z tego, że możesz korzystać z formularzy bez obawy, iż informacje o sesji ulegną zapomnieniu.

Chociaż w poniższych ćwiczeniach wykorzystano bardzo prosty przykład, wyraźnie widać, że wykorzystanie sesji w PHP może okazać się niezwykle przydatne. Możesz je wykorzystać, nie tylko projektując strony *e-commerce* (koszyk zamówień jest najbardziej typowym przykładem użycia sesji), ale także w licznych innych zastosowaniach.

Baza danych

To już nieco wyższa szkoła jazdy. Projektując aplikacje webowe, musisz zadać sobie pytanie, czy obecna — lub przyszła — ilość informacji, które są pamiętane, nie powoduje, że powinieneś zainteresować się bazą danych.

Na szczęście nie musisz inwestować w oprogramowanie. Mimo iż istnieją bardzo drogie, komercyjne systemy, są dostępne również mniejsze, darmowe dla celów niekomercyjnych rozwiązania, doskonale sprawdzające się w małych i średnich serwisach internetowych. Jednym z nich jest MySQL, który chyba najczęściej bywa łączony z PHP, dlatego zajmę się nim w tym rozdziale.

Baza danych jest po prostu pewnym zbiorem informacji. MySQL jest systemem zarządzania bazą danych (SZBD), który umożliwia w łatwy sposób dostęp do informacji i ich modyfikację.

Podstawową jednostką organizacyjną w bazie danych jest *tabela*. Możesz ją sobie wyobrazić jako tablicę, w której wierszach są umieszczane *rekordy*, opisujące pojedyncze obiekty. Poszczególne kolumny tabeli opisują *pola* danego rekordu, które zawierają odpowiednie dotyczące go informacje. Poniżej znajdziesz ilustrację w postaci tablicy prostej tabeli o nazwie `ksiazkatelefoniczna`, którą mógłbyś chcieć wykorzystać w swoim systemie do pamiętania numerów telefonów swoich znajomych.

Nr	imie	nazwisko	telefon
1	Jan	Kowalski	32589442
2	Adam	Nowak	23423445
3	Kazimierz	Polak	12387554
4	Anna	Górska	69666651

Jeżeli zainstalowałeś Krasnala, MySQL-a masz już poprawnie skonfigurowanego. Jedyne, co musisz zrobić, to uruchomić program *Krasnal Start* z menu *Start*/*Programy*/*KRASNAL Serv*. W przeciwieństwie do serwera Apache nie zobaczysz wizualnych efektów działania — MySQL zainstaluje się jako proces systemowy.

Administracja bazą danych jest łatwa dzięki programowi PHPMyAdmin — napisanemu w PHP systemowi administracji bazą danych. Ten program także masz w swoim systemie, pora więc rozpocząć działanie.

ĆWICZENIE

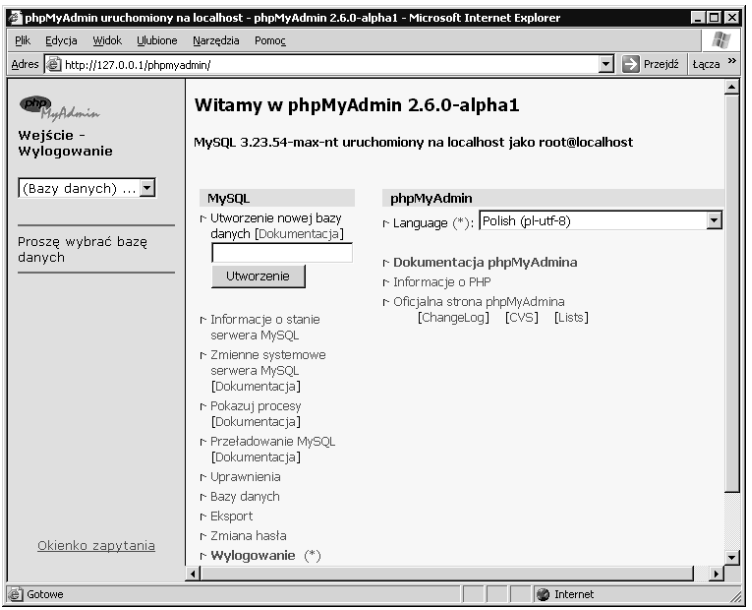
4.23 Tworzenie bazy danych w PHPMyAdmin

Uruchom program PHPMyAdmin. Utwórz nową bazę danych o nazwie cwphp, a w niej tabelę ksiazkatelefoniczna.

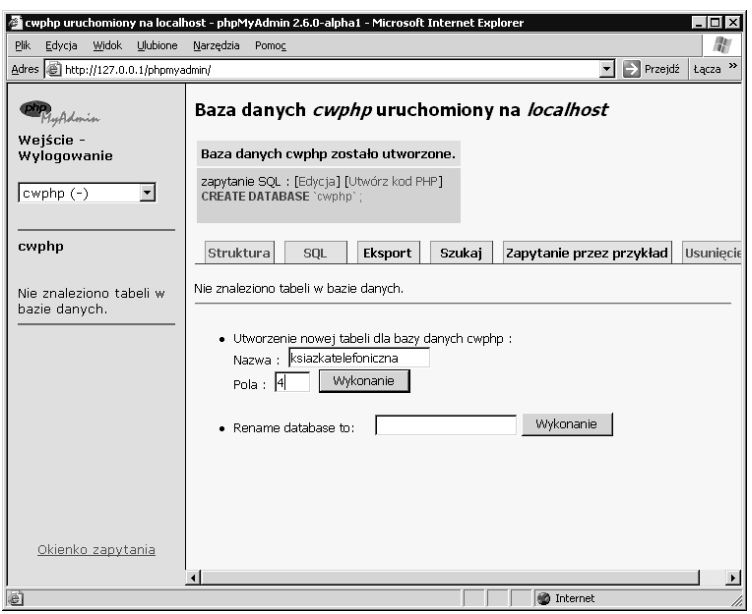
Aby uruchomić program PHPMyAdmin, połącz się z adresem *http://localhost/phpmyadmin* (powinieneś mieć już uruchomiony serwer Apache i MySQL-a). Krasnal zabezpiecza bazę przed niepowołanym dostępem — użyj nazwy użytkownika *root* i hasła *krasnal*. Ukaże się okienko programu administracyjnego. W celu założenia nowej bazy danych wpisz jej nazwę w odpowiednim polu i kliknij przycisk *Utworzenie* (rysunek 4.1). Utwórz więc bazę o nazwie *cwphp*.

Po utworzeniu bazy danych pojawi się ona w lewej ramce. Jeżeli utworzysz jakieś tabele, będą widoczne po rozwinięciu gałęzi bazy danych. Przystąp do tworzenia tabeli. Odpowiednie pole znajduje się w dolnej części prawej ramki. W określonych polach formularza wpisz nazwę tabeli (`ksiazkatelefoniczna`) i liczbę kolumn w tabeli, czyli pól rekordu. W Twoim przypadku jest to `4`. Spójrz na ilustrację 4.2, by zobaczyć, jak przebiega proces zakładania tabeli w bazie.

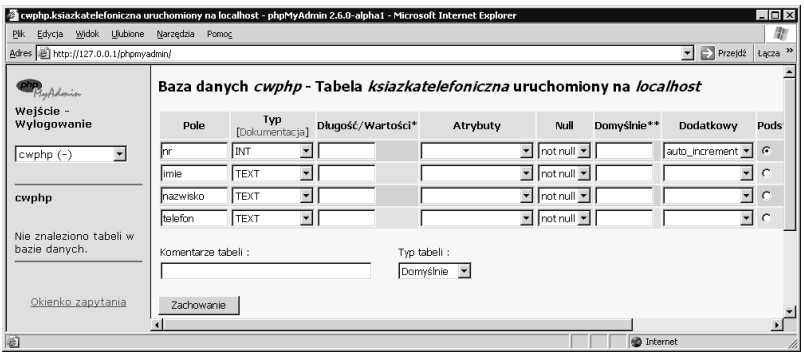
Po kliknięciu przycisku *Wykonanie* pojawia się okno, w którym należy określić pola tabeli (rysunek 4.3). Pola `imie`, `nazwisko` i `telefon` będą typu tekstowego. W kolumnie `Pole` formularza nadaj odpowiednie nazwy. Jako `Typ` wybierasz `TEXT`. Ponieważ nie chcesz, by pola mogły mieć pustą wartość, w kolumnie `Null` określasz wartość `not null`.



Rysunek 4.1. *Rozpoczynasz tworzyć bazę danych*



Rysunek 4.2. *Tworzenie tabeli*

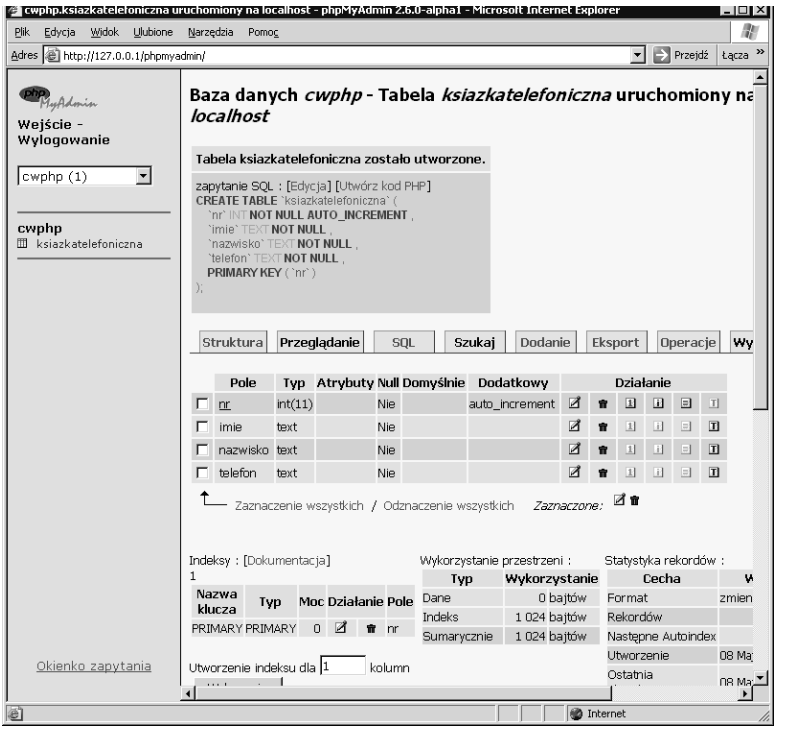


Rysunek 4.3. *Określanie pól tabeli*

Bardziej skomplikowanie przedstawia się sytuacja z pierwszym polem. Ponieważ dane w nim zawarte są liczbami, nadajesz polu typ `INT`. Może pojawić się pytanie, do czego chcesz to pole wykorzystywać. Otóż nie będziesz w żaden sposób za jego pomocą liczyć czy porządkować rekordów w tabeli. Potrzebujesz go, by utworzyć *klucz* tabeli: unikatowy identyfikator, poprzez który można się będzie odwoływać do danego rekordu (może się to okazać potrzebne do określania relacji pomiędzy tabelami). Nie trzeba takiego klucza nadawać samodzielnie, wystarczy, żeby sama baza nadała polu odpowiednią wartość. Ustawiasz więc następujące cechy *Dodatkowy:* `auto_increment` oraz *Podstawowy* (to cecha klucza). Będziesz dzięki temu mieć możliwość szybkiego wyszukiwania informacji pod kątem tego pola. Jego zaznaczenie spowoduje, że baza zapamięta specjalne informacje (indeks), które pozwolą na znacznie szybszy dostęp do danych.

Na rysunku 4.3 możesz zobaczyć sposób definiowania pól formularza dla Twojej tabeli. Rysunek 4.4 przedstawia natomiast gotowy efekt: zdefiniowaną tabelę. Zauważ, że zdefiniowana tabela pojawiła się w menu w lewej ramce.

Po założeniu tabeli, używając programu PHPMyAdmin, możesz wykonywać na niej podstawowe operacje, na przykład przeglądania, dodawania, usuwania i modyfikowania rekordów.



Rysunek 4.4. *Tabela została założona*

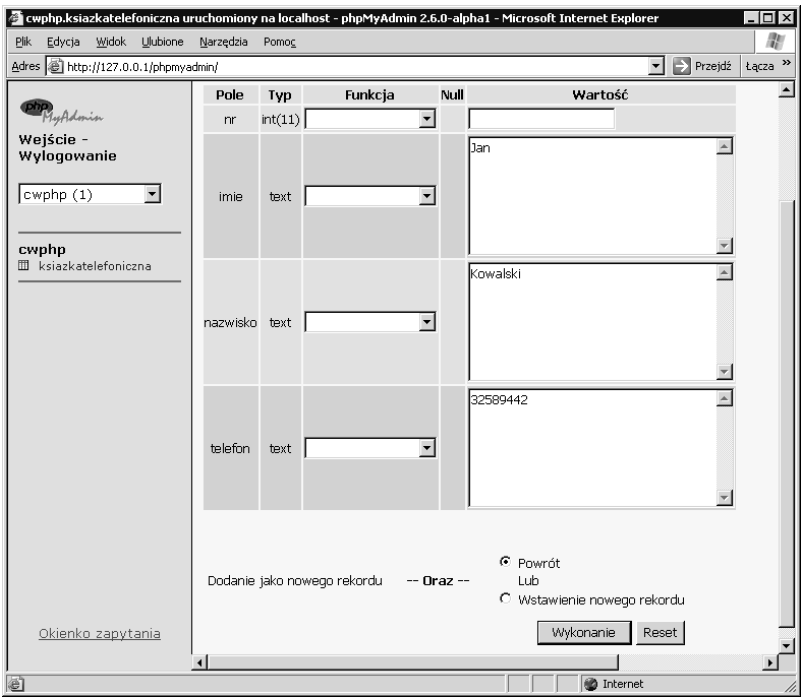
ĆWICZENIE

4.24 Wpisywanie danych za pomocą PHPMyAdmin

Korzystając z programu PHPMyAdmin, wpisz dane do tabeli ksiazkatelefoniczna.

W lewym menu wybierz żądaną tabelę w lewej ramce, a następnie w prawej opcję *Dodanie*. Pojawia się formularz (rysunek 4.5), w który możesz wpisać rekord. Wypełnij jedynie kolumnę `Wartość`, pozostawiając wolne pole dla `nr` (tabelę zdefiniowałeś tak, że wartość zostanie określona automatycznie).

Po wpisaniu danych wciśnij przycisk *Wykonanie*. W podobny sposób wpisz następne rekordy.



Rysunek 4.5. *Wstawianie rekordu*

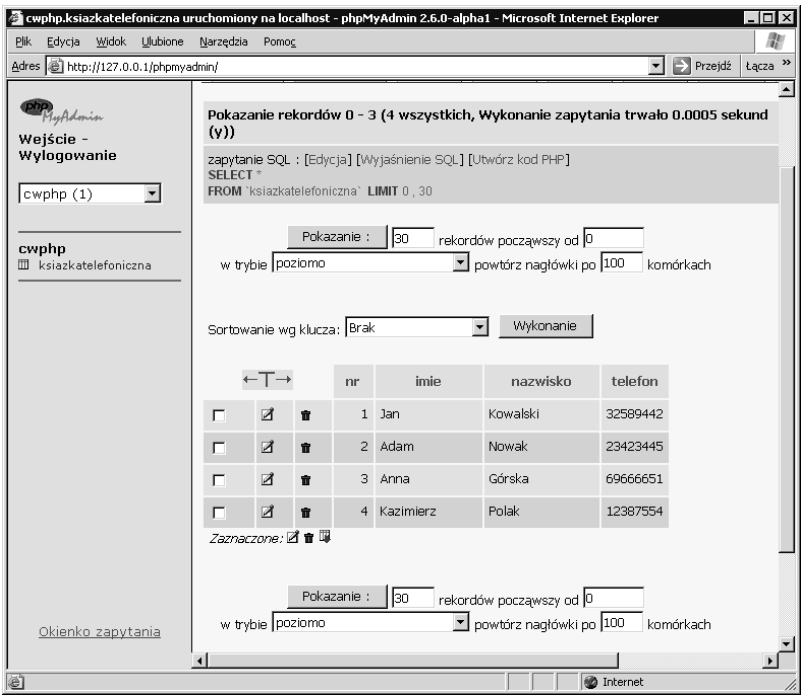
ĆWICZENIE

4.25 Przeglądanie danych za pomocą PHPMyAdmin

Korzystając z programu PHPMyAdmin, sprawdź zawartość tabeli ksiazkatelefoniczna.

W lewym menu wybierz żądaną tabelę w lewej ramce, a następnie w prawej opcję *Przeglądanie*. Zobaczysz zawartość tabeli (jeżeli będzie miała wiele rekordów, zapoznasz się z jej częścią i będziesz miał możliwość przewijania).

Jak widzisz (rysunek 4.6), opcja *Przeglądanie* pozwala Ci na edycję i usuwanie rekordów z tablicy. Przeanalizuj to w ramach dodatkowego ćwiczenia.



Rysunek 4.6. *Opcja Przeglądanie — przeglądanie wpisywanych danych*

Jeżeli będziesz usuwał i wprowadzał dane do tabeli, zauważysz szybko, że liczba porządkowa (pole *nr*) nie będzie stanowiła spójnej numeracji od 1 do liczby rekordów, lecz pojawią się „dziury". Nie należy się tym martwić — tak właśnie ma być. Warto przypomnieć sobie, do czego jest potrzebne to pole. Nie dlatego, by numerować rekordy, lecz by opatrzyć je indeksem unikatowym, stałym dla danego rekordu i nieopatrzonym „historią" — to znaczy niezwiązanym z innym, uprzednio usuniętym rekordem.

Jak widzisz, za pomocą PHPMyAdmin możesz dowolnie modyfikować dane z bazy danych MySQL. Pomimo tego, iż wiele serwisów zawiera własne moduły administracyjne pozwalające na edycję dotyczących ich danych, na początek możesz zająć się tym, co widać po stronie

klienta, pozostawiając kwestię administracji temu programowi. Docelowo jednak powinieneś utworzyć dla swojego serwisu własny moduł administracyjny, w którym wedle własnego uznania będziesz mógł dbać o dane serwisu.

Administracja bazą danych MySQL za pomocą programu PHPMyAdmin okazała się bardzo łatwa. Pamiętaj jednak, że baza została założona w celu wykorzystania na stronach WWW.

Zajęcie się bazą danych z poziomu PHP nie jest trudne. Za pomocą PHP masz możliwość pozyskiwania informacji z bazy danych, dodawania nowych, usuwania niepotrzebnych, a nawet dokonywania czynności administracyjnych, takich jak zakładanie nowych baz danych, tworzenie tabel i ich modyfikacja itd. Być może zwróciłeś uwagę, że PHPMyAdmin jest faktycznie aplikacją napisaną w PHP. Już widziałeś jego potężne możliwości. To, co było możliwe za jego pomocą, mniejszym lub większym nakładem pracy możesz zaprogramować za pomocą PHP.

ĆWICZENIE

4.26 Wyświetlanie danych z bazy

Za pomocą PHP połącz się z bazą danych MySQL. Doprowadź do tego, by dane z tabeli ksiazkatelefoniczna zostały wyświetlone na stronie WWW.

Operacja na bazie danych MySQL (podobnie jak i na innych) opiera się na zadawaniu zapytań w języku *SQL* (ang. *Structured Query Language*). Tak naprawdę wszystkie wykonane w poprzednich ćwiczeniach operacje na bazie danych (zakładanie bazy danych, tworzenie tabeli, dodawanie rekordów, wyświetlanie zawartości tabeli) były realizowane za pomocą właśnie takich zapytań. Zwróć uwagę na zapis, znajdujący się na rysunku 4.6.:

```
SELECT * FROM ksiazkatelefoniczna LIMIT 0, 30
```

Jest to nic innego, jak zapytanie do bazy danych, nakazujące wyświetlić wszystkie pola z tablicy `ksiazkatelefoniczna` dla pierwszych 30 rekordów. Dzięki wydaniu bazy danych takiego polecenia otrzymałeś wynik ćwiczenia 4.25.

Spróbuj zrealizować to samo za pomocą własnego programu PHP.

4-26.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Czytanie z bazy danych</TITLE>
  </HEAD>
  <BODY>
    <? // Odczyt z bazy danych składa się z przyłączenia się do MySQL,
       // wybrania bazy danych, zadania pytania SQL i przetworzenia
          wyników.

      mysql_connect ("localhost", "root", "krasnal") or
        die ("Nie można połączyć się z MySQL");
      mysql_select_db ("cwphp") or
        die ("Nie można połączyć się bazą cwphp");

      $wynik = mysql_query ("SELECT * FROM ksiazkatelefoniczna;") or
        die ("błąd w pytaniu");
      print "<TABLE CELLPADDING=5 BORDER=1>";
      print "<TR><TD><B>Imię</B></TD><TD><B>Nazwisko</B></TD>";
      print "<TD><B>Telefon</B></TD></TR>\n";


      while ($rekord = mysql_fetch_assoc ($wynik)) {
        $nr = $rekord['nr'];
        $imie = $rekord['imie'];
        $nazwisko = $rekord['nazwisko'];
        $telefon = $rekord['telefon'];

      print "<TR><TD>$imie</TD><TD>$nazwisko</TD>";
      print "<TD>$telefon</TD></TR>\n";
      }
      print "</TABLE>";

    ?>

  </BODY>
</HTML>
```

Operacja dostępu do bazy danych składa się z kilku etapów, które powinny zostać wykonane w odpowiedniej kolejności i żadnego z nich nie można pominąć.

Pierwszą operacją jest nawiązanie połączenia z MySQL-em. Posługujesz się w tym celu funkcją `mysql_connect`.

`mysql_connect` Nawiązanie połączenia z MySQL-em.

Pierwszym parametrem tej funkcji jest nazwa hosta. Jeśli pracujesz na lokalnym komputerze, wpisz `localhost`. Jako drugi argument wprowadź nazwę użytkownika. Ponieważ na razie nie definiowałeś użytkowników, wykorzystasz z domyślnego użytkownika `root`. Trzecim parametrem jest hasło. Po instalacji Krasnala hasło domyślne `roota` brzmi `krasnal`.

W przyszłości podczas wykorzystania bazy danych nie na testowym, lecz na rzeczywistym serwerze, koniecznie załóż specjalnego użytkownika, który będzie miał dostęp do danych na potrzeby serwisu. Określ dla niego niebanalne hasło oraz nadaj mu odpowiednie uprawnienia.

Zwróć uwagę na konstrukcję:

```
funkcja (parametry) or die ("komunikat");
```

Oznacza ona wykonanie funkcji, a w przypadku niepowodzenia — zakończenie działania programu z odpowiednim komunikatem. Przy funkcjach operujących na bazie danych wykorzystaj tę konstrukcję, by w przypadku niepowodzenia wiedzieć, co się dzieje, i nie wykonywać dalej niepotrzebnych operacji.

Następnie należy wybrać bazę danych, używając funkcji `mysql_select_db`. Wybierasz `cwphp`.

`mysql_select_db` Wybór bazy danych.

Możesz już zadać pytanie do bazy danych. Posłuż się poznanym już zapytaniem SQL-owym (używając funkcji `mysql_query`):

```
SELECT * FROM ksiazkatelefoniczna;
```

Po zadaniu zapytania można już odczytywać jego wyniki. Jedną z typowych metod jest pętla:

```
while ($rekord = mysql_fetch_assoc ($wynik)) {
  operacje na rekordzie
}
```

Pętla odczytuje kolejne rekordy, które są wynikiem zapytania, zapisując je w tablicy `$rekord`. Komórki tej tablicy zawierają kolejne pola wyników zapytania, dostęp do nich jest możliwy za pomocą nazw kolumn tabeli użytych jako klucze tablicy.

Znak * w zapytaniu `SELECT` oznacza, że interesują Cię wszystkie pola. Z tego powodu tablica będzie zawierała wartości `$rekord['nr']` — numer, `$rekord['imie']` — imię, `$rekord['nazwisko']` — nazwisko, `$rekord['telefon']` — telefon.

W pętli odczytujesz te wartości i wyświetlasz je, formatując je w tabeli.

Inną często wykonywaną operacją na danych jest usuwanie rekordów. Można je łatwo wykonać, posługując się zapytaniem SQL-owym, które ma postać:

```
DELETE FROM nazwa_tabeli WHERE warunek;
```

Spróbuj usunąć rekord z tabeli, używając programu PHPMyAdmin, i zobacz, jak wygląda zapytanie, które usuwa dane w przypadku tabeli `ksiazkatelefoniczna`.

ĆWICZENIE

4.27 Usuwanie danych z bazy

Uzupełnij program z poprzedniego ćwiczenia o opcję usuwania danych.

Tabelę wyświetlającą dane uzupełnisz o połączenie „skasuj". Wywoła ono ten sam skrypt, z parametrem `co`, określającym, że należy przeprowadzić usuwanie danych, oraz `id`, podającym numer rekordu do usunięcia.

4-27.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Czytanie z bazy danych i usuwanie rekordów</TITLE>
  </HEAD>
  <BODY>
    <? // Aby usunąć rekord, przekazujesz informacje do skryptu metodą GET
       // i usuwasz SQL-owym zapytaniem DELETE.

      mysql_connect ("localhost", "root", "krasnal") or
        die ("Nie można połączyć się z MySQL");
      mysql_select_db ("cwphp") or
        die ("Nie można połączyć się z bazą cwphp");
```

```
    if ($_GET['co'] == 'skasuj') {
      $wynik = mysql_query
           ("DELETE FROM ksiazkatelefoniczna WHERE nr = '".$_GET
           ['id']."';");
    }

    $wynik = mysql_query ("SELECT * FROM ksiazkatelefoniczna;") or
      die ("błąd w pytaniu");

    print "<TABLE CELLPADDING=5 BORDER=1>";
    print "<TR><TD><B>Imię</B></TD><TD><B>Nazwisko</B></TD>";
    print "<TD><B>Telefon</B></TD><TD></TD></TR>\n";

    while ($rekord = mysql_fetch_assoc ($wynik)) {
      $nr = $rekord['nr'];
      $imie = $rekord['imie'];
      $nazwisko = $rekord['nazwisko'];
      $telefon = $rekord['telefon'];

      print "<TR><TD>$imie</TD><TD>$nazwisko
      </TD><TD>$telefon</TD><TD>";
      print "<A HREF=\"4-27.php?co=skasuj&id=$nr\">skasuj
      </A></TD></TR>\n";
    }
    print "</TABLE>";

  ?>

 </BODY>
</HTML>
```

Jeżeli przed wyświetleniem danych skrypt rozpozna, że należy przeprowadzić usuwanie danych, uruchamia usuwające zapytanie SQL. Po jego realizacji są wyświetlane pozostałe rekordy.

Skoro umiesz już dane przeglądać i usuwać, pora nauczyć się je dopisywać. Pomoże w tym SQL-owe zapytanie w postaci:

```
INSERT INTO nazwa_tabeli (spis_pól) VALUES (wartości);
```

Zwróć uwagę, jak wygląda zapytanie SQL-owe podczas dodawania rekordu za pomocą programu PHPMyAdmin.

ĆWICZENIE

4.28 Dodawanie rekordów do bazy danych

Popraw skrypt z poprzedniego ćwiczenia tak, by umożliwiał również dodawanie nowych rekordów.

Pod wyświetloną zawartością bazy przygotujesz formularz z możliwością wpisania nowego rekordu. W ukrytym polu ustaw zmienną `co` (będzie określać operację, w przypadku dopisywania rekordu — `dodaj`). Dopisywanie jest realizowane jedynie wtedy, gdy każde z trzech pól zostanie wypełnione (nie chcesz mieć przecież niekompletnej książki telefonicznej).

4-28.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Czytanie z bazy danych, usuwanie i dodawanie rekordów</TITLE>
  </HEAD>
  <BODY>
    <? // Dodawanie rekordu zrealizowano za pomocą zapytania INSERT.
       // Dane są wczytywane w przygotowanym formularzu.
      mysql_connect ("localhost", "root", "krasnal") or
        die ("Nie można połączyć się z MySQL");
      mysql_select_db ("cwphp") or
        die ("Nie można połączyć się z bazą cwphp");

      if ($_POST['co'] == 'dodaj') {         // dodawanie rekordu
        if ($_POST['imie'] && $_POST['nazwisko'] && $_POST['telefon']) {
          $query = "INSERT INTO ksiazkatelefoniczna (imie, nazwisko, ";
          $query .= "telefon) VALUES ('".$_POST['imie']."', '".$_POST
          ['nazwisko']."',
          '".$_POST['telefon']."');";
          $wynik = mysql_query ($query);
        }
      } elseif ($_GET['co'] == 'skasuj') {  // usuwanie
        $wynik = mysql_query
             ("DELETE FROM ksiazkatelefoniczna WHERE nr = '".$_GET
             ['id']."';");
      }

      $wynik = mysql_query ("SELECT * FROM ksiazkatelefoniczna;") or
        die ("błąd w pytaniu");

      print "<TABLE CELLPADDING=5 BORDER=1>";
      print "<TR><TD><B>Imię</B></TD><TD><B>Nazwisko</B></TD>";
      print "<TD><B>Telefon</B></TD><TD></TD></TR>\n";
```

```
    while ($rekord = mysql_fetch_assoc ($wynik)) {
      $nr = $rekord['nr'];
      $imie = $rekord['imie'];
      $nazwisko = $rekord['nazwisko'];
      $telefon = $rekord['telefon'];

      print "<TR><TD>$imie</TD><TD>$nazwisko</TD><TD>$telefon</TD><TD>";
      print "<A HREF=\"4-28.php?co=skasuj&id=$nr\">skasuj
      </A></TD></TR>\n";
    }
    print "</TABLE>";
    print '<FORM METHOD="POST">Nowy rekord:';
    print '<INPUT TYPE="hidden" NAME="co" VALUE="dodaj"><TABLE>';
    print '<TR><TD>Imię:</TD><TD><INPUT TYPE="text" ';
    print 'NAME="imie"></TD></TR><TR><TD>Nazwisko:</TD><TD><INPUT ';
    print 'TYPE="text" NAME="nazwisko"></TD></TR>';
    print '<TR><TD>Telefon:</TD>';
    print '<TD><INPUT TYPE="text" NAME="telefon"></TD></TR>';
    print '</TABLE><INPUT TYPE="submit" VALUE="Dodaj"></FORM>';
  ?>

  </BODY>
</HTML>
```

Warto zwrócić uwagę, że w formularzu w znaczniku `FORM` nie ma w ogóle atrybutu `ACTION`. Jeżeli nie wskażesz skryptu, który ma zostać wywołany do obsługi formularza, zostanie użyty bieżący.

Twoja bardzo prosta książka telefoniczna jest już prawie gotowa. Ponieważ jednak czasem znajomi zmieniają telefony, warto byłoby ją wyposażyć w możliwość edycji danych. Poprawę danych umożliwia zapytanie `UPDATE`:

```
UPDATE nazwatabeli SET pole1='wartosc1', pole2='wartosc2'... WHERE warunek;
```

ĆWICZENIE

4.29 Edycja rekordów w bazie danych

Uzupełnij program, który obsługuje książkę telefoniczną, o edycję rekordów.

W tabeli zawierającej wyświetlane rekordy poza połączeniem pozwalającym na usunięcie dowolnego z nich, wstaw też drugie, umożliwiające przejście do edycji rekordu. Dane (zmienne `co` i `id`) przekażesz

metodą GET. Zmiennej $_GET['co'] nadaj wartość "edytuj", co będzie oznaczało, że należy wyświetlić formularz edycji rekordu.

Aplikacja nieco się skomplikuje. Zmienna $_GET['co'] będzie mogła przyjąć dwie nowe wartości, które musisz obsłużyć. Pierwsza z nich to wspomniane już "edytuj". W takim przypadku należy odczytać z bazy danych konkretny rekord (udaje się to dzięki wykorzystaniu zapytania SELECT z określonym warunkiem WHERE nr='$_GET['id'] '). Po odczytaniu danych (jednego rekordu) wyświetlasz formularz, wstawiając dla pól domyślne (odczytane z bazy) wartości danych. Zmiennej $_GET['co'] (przekazywanej dalej w polu ukrytym, czyli stającej się zmienną $_POST ['co']) nadajesz wartość "popraw".

Jeżeli odkryjesz, że zmienna $_POST['co'] przybiera wartość "popraw", oznacza to, że odczytałeś z formularza poprawione dane. Wykonujesz więc operację UPDATE na bazie. Musisz jej dokonać tylko na jednym rekordzie (jego numer także został w poprzednim formularzu przekazany przez ukryte pole), więc korzystasz z warunku WHERE nr='$_POST['id'] '.

4-29.php

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
    <TITLE>Działająca książka telefoniczna</TITLE>
  </HEAD>
  <BODY>
    <? // Aplikacja daje nam możliwość odczytu, usuwania, dodawania
       // i poprawy danych w bazie.

      mysql_connect ("localhost", "root", "krasnal") or
        die ("Nie można połączyć się z MySQL");
      mysql_select_db ("cwphp") or
        die ("Nie można połączyć się bazą cwphp");

      if ($_POST['co'] == 'dodaj') {  // dodawanie rekordu
        if ($_POST['imie'] && $_POST['nazwisko'] && $_POST['telefon']) {
          $query = "INSERT INTO ksiazkatelefoniczna (imie, nazwisko, ";
          $query .= "telefon) VALUES ('".$ _POST['imie']."',
          '".$_POST['nazwisko']."', '".$ _POST['telefon']."');";
          $wynik = mysql_query ($query);
        }
      } elseif ($_POST['co'] == 'popraw') {  // poprawianie rekordu
        if ($_POST['imie'] && $_POST['nazwisko'] && $_POST['telefon']) {
          $query = "UPDATE ksiazkatelefoniczna SET imie='".$_POST
          ['imie']."',nazwisko=";
```

```
    $query .= "'".$_POST['nazwisko']."', telefon='".$_POST
    ['telefon']."' WHERE
    nr='".$_POST['id']."';";
    $wynik = mysql_query ($query);
  }
} elseif ($_GET['co'] == 'edytuj') {  // przygotowanie do poprawek
   $query = "SELECT * FROM ksiazkatelefoniczna where nr='".$_GET
   ['id']."';";
   $wynik = mysql_query ($query);
   $rekord = mysql_fetch_assoc ($wynik);
   $nr = $rekord['nr']; $imie = $rekord['imie']; $nazwisko =
   $rekord['nazwisko'];
   $telefon = $rekord['telefon'];
   print '<FORM METHOD="POST">Poprawa rekordu:';
   print '<INPUT TYPE="hidden" NAME="co" VALUE="popraw">';
   print '<INPUT TYPE="hidden" NAME="id" VALUE="'.$nr.'"><TABLE>';
   print '<TR><TD>Imię:</TD><TD><INPUT TYPE="text" ';
   print 'NAME="imie" VALUE="'.$imie.'"></TD></TR><TR><TD>Nazwisko:';
   print '</TD><TD><INPUT TYPE="text" NAME="nazwisko"';
   print 'VALUE="'.$nazwisko;
   print '"></TD></TR><TR><TD>Telefon:</TD><TD><INPUT TYPE="text"';
   print 'NAME="telefon" VALUE="'.$telefon.'"></TD></TR>';
   print '</TABLE><INPUT TYPE="submit" VALUE="Popraw"></FORM>';
} elseif ($_GET['co'] == 'skasuj') {  // usuwanie
  $wynik = mysql_query
       ("DELETE FROM ksiazkatelefoniczna WHERE nr = '".$_GET
       ['id']."';");
}

$wynik = mysql_query ("SELECT * FROM ksiazkatelefoniczna;");

print "<TABLE CELLPADDING=5 BORDER=1>";
print "<TR><TD><B>Imię</B></TD><TD><B>Nazwisko</B></TD>";
print "<TD><B>Telefon</B></TD><TD></TD><TD></TD></TR>\n";

while ($rekord = mysql_fetch_assoc ($wynik)) {
  $nr = $rekord['nr'];
  $imie = $rekord['imie'];
  $nazwisko = $rekord['nazwisko'];
  $telefon = $rekord['telefon'];

  print "<TR><TD>$imie</TD><TD>$nazwisko</TD><TD>$telefon</TD><TD>";
  print "<A HREF=\"4-29.php?co=skasuj&id=$nr\">skasuj</A></TD><TD>";
  print "<A HREF=\"4-29.php?co=edytuj&id=$nr\">edytuj</A></TD>
  </TR>\n";
}
print "</TABLE>";
print '<FORM METHOD="POST">Nowy rekord:';
print '<INPUT TYPE="hidden" NAME="co" VALUE="dodaj"><TABLE>';
print '<TR><TD>Imię:</TD><TD><INPUT TYPE="text" ';
print 'NAME="imie"></TD></TR><TR><TD>Nazwisko:</TD><TD><INPUT ';
```

```
      print 'TYPE="text" NAME="nazwisko"></TD></TR><TR><TD>Telefon:</TD>';
      print '<TD><INPUT TYPE="text" NAME="telefon"></TD></TR>';
      print '</TABLE><INPUT TYPE="submit" VALUE="Dodaj"></FORM>';
    ?>

  </BODY>
</HTML>
```

Jeżeli chciałbyś, by wyświetlane dane były uporządkowane względem nazwiska, wystarczy, że poprawisz nieco zapytanie wybierające wszystkie dane na następującą postać:

```
SELECT * FROM ksiazkatelefoniczna ORDER BY nazwisko;
```

Należałoby jeszcze uzupełnić aplikację o wyszukiwanie danych. W tym celu trzeba wprowadzić formularz, który pozwoli na wpisanie wyszukiwanej frazy (pole nazwij `fraza`). Następnie należy przeprowadzić selekcję danych. Można to wykonać na dwa sposoby. Szybszym i pewniejszym jest zaangażowanie do wyszukiwania silnika bazy danych. Odpowiednie skonstruowanie zapytania `SELECT` (a dokładniej jego sekcji `WHERE`) spowoduje, że zapytanie zwróci jedynie te rekordy, które pasują do szukanej frazy.

W tym przypadku wykorzystasz inną metodę, dzięki której poznasz sposób na wyszukiwanie dowolnych podciągów w ciągu znaków. Zapytanie `SELECT` zwróci Ci — jak w poprzednich przykładach — wszystkie rekordy. Przeprowadzisz analizę każdego z nich i wyświetlisz tylko te, które odpowiadają frazie.

Pętla `while` przybiera więc postać:

4-29a.php (fragment)

```
...
while ($rekord = mysql_fetch_assoc ($wynik)) {
  $nr = $rekord['nr'];
  $imie = $rekord['imie'];
  $nazwisko = $rekord['nazwisko'];
  $telefon = $rekord['telefon'];

  if (!($_POST['fraza']) || (stristr ($imie.$nazwisko, $_POST['fraza']))) {
    print "<TR><TD>$imie</TD><TD>$nazwisko</TD><TD>$telefon</TD><TD>";
    print "<A HREF=\"4-29.php?co=skasuj&id=$nr\">skasuj</A></TD><TD>";
    print "<A HREF=\"4-29.php?co=edytuj&id=$nr\">
    edytuj</A></TD></TR>\n";
  }
}
...
```

W celu zbadania podobieństwa frazy z łańcuchem wykorzystujesz funkcję `stristr`.

`stristr`	Wyszukuje wystąpienie łańcucha stanowiącego drugi argument w pierwszym, bez zwracania uwagi na wielkość znaków.

Działający przykład znajdziesz w pliku *4-29a.php*. Można go jeszcze uzupełnić o odpowiednie przetworzenie polskich znaków diakrytycznych. Argumenty funkcji `stristr` możesz próbować najpierw pozbawić polskich znaków za pomocą napisanej przez siebie funkcji, by użytkownikowi jeszcze łatwiej było uzyskać interesujące go informacje.

W przykładzie, który był prezentowany w ćwiczeniach dotyczących bazy danych, może pojawić się problem, gdy użytkownik będzie próbował użyć we wpisywanych przez siebie polach znaków, które w zapytaniach SQL odgrywają specjalną rolę (na przykład apostrofu, który ogranicza łańcuch znakowy). W związku z tym bardzo dobrym zwyczajem jest — przed zadaniem zapytania SQL — użycie na wpisanych danych funkcji, która „zabezpieczy" dane, opatrując niektóre znaki ukośnikami. Jeśli zapisujesz dane w bazie, to po odczycie otrzymasz ciąg bez dodatkowych ukośników. Czasami może jednak być potrzebne konwertowanie ciągu w drugą stronę, np. kiedy samodzielnie zapisujesz dane w plikach własnego formatu. Pomocne okazują się tutaj dwie funkcje:

`addslashes`	Poprzedza niektóre znaki argumentu, będącego łańcuchem, znakiem ukośnika, na przykład w celu umożliwienia bezpiecznego zapisu do bazy danych.
`stripslashes`	Przywraca łańcuch, przetworzony funkcją `addslashes`, do stanu pierwotnego.

Zapytania SQL-owe mogą mieć bardzo rozbudowaną treść. Mogą odwoływać się do kilku tabel i wykonywać na danych skomplikowane operacje.

Jeżeli w przyszłości będziesz miał wątpliwości, czy w rozwiązaniu problemu wykorzystać mechanizmy bazy danych, czy też oprogramować je „na piechotę", wykonując na bazie jedynie prostą operację,

nie wahaj się. Jeżeli będziesz w stanie, zrób to za pomocą SQL-a. Duża część odpowiedzialności za sprawne działanie spadnie z Ciebie na silnik bazy danych.

Pamiętaj, że niezwykle ważne, szczególnie przy dużych projektach, jest poprawne zaprojektowanie struktury bazy danych (właściwe zdefiniowanie tabel i powiązań między nimi, określenie kluczy i indeksów). Musisz także dbać o zapewnienie odpowiednich uprawnień (systemu użytkowników i haseł). W ten sposób masz szansę na utworzenie bezpiecznej i dobrze działającej bazy danych.

Poznałeś jedynie podstawowe zagadnienia z zakresu współpracy PHP z bazą danych i SQL-em. Nieco więcej przykładów zobaczysz w ćwiczeniach z ostatniego rozdziału książki. Jeżeli jednak chcesz myśleć o bardziej zaawansowanych zastosowaniach, powinieneś poznać SQL-a bliżej. Dlatego godne polecenia są znakomite książki:

- *Bazy danych i MySQL. Od podstaw*, Richard Stones, Neil Matthew,
- *PHP i MySQL. Tworzenie stron WWW. Wydanie drugie. Vademecum profesjonalisty*, Luke Welling, Laura Thomson,
- *MySQL. Leksykon kieszonkowy*, George Reese,
- *PHP i MySQL. Dynamiczne strony WWW. Szybki start*, Larry Ullman,
- *SQL dla każdego*, Rafe Coburn,
- *SQL w mgnieniu oka*, Ben Forta.