Tworzenie serwisów WWW. Pierwsza pomoc

Autorzy: Maria Sokół, Radosław Sokół
ISBN: 978-83-246-1073-0
Format: A5, stron: 112

- Osobisty serwer HTTP – instalacja, konfiguracja, zabezpieczanie
- Podstawy języka PHP – tworzenie dynamicznych stron WWW
- Korzystanie z baz danych – wprowadzanie, modyfikacja i kasowanie informacji
- Używanie mechanizmu AJAX – natychmiastowa reakcja strony na działania użytkownika

Przyznaj się – nudzi Cię już tworzenie zwyczajnych, statycznych stron internetowych? Masz apetyt na więcej i chętnie podjąłbyś wyzwanie polegające na stworzeniu całego, dynamicznie zmieniającego się układu stron? Ta książka to recepta na Twoje potrzeby! Dzięki niej masz wreszcie możliwość wykreowania interaktywnego, funkcjonalnego serwisu WWW, który nie tylko udostępni użytkownikom sensownie posegregowane, wyczerpujące informacje i nie narazi ich na konieczność ciągłego odświeżania strony, a przy tym nie utrudni Ci jego obsługi. Spieszymy z fachową pierwszą pomocą – z niezbędną wiedzą!

- Pakiet XAMPP – pobieranie, instalacja, moduł administracyjny
- Serwer WWW – testowanie i podstawy umieszczania stron
- Język PHP – zmienne, warunki, tablice i inne
- Interakcja z użytkownikiem – wprowadzanie danych do programu i ich archiwizacja
- Podstawy obsługi bazy MySQL – zakładanie, zarządzanie, konta użytkowników
- Baza danych MySQL – nowa tabela, prawa dostępu, połączenie z poziomu kodu PHP
- Baza danych a strona WWW – współdziałanie i połączenia
- Zapytania SQL – możliwość wyszukiwania i segregacji danych
- Technologia AJAX – sprytne sztuczki do wykorzystania

Twoja recepta na profesjonalne tworzenie nowoczesnych serwisów WWW!

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

Spis treści

25. Jak nawiązywać połączenie z bazą danych z poziomu kodu PHP?

Baza danych obsługiwana za pomocą specyficznego narzędzia nie jest specjalnie przydatna. Owszem, jej możliwości wyszukiwania i grupowania danych mogą być naprawdę wielkie, jeżeli jednak użytkownik bazy nie ma wobec niej takich wymagań, prosty arkusz kalkulacyjny będzie wygodniejszy w obsłudze i umożliwi szybsze wprowadzenie danych.

Siłą baz danych jest jednak możliwość bezpośredniej współpracy z własnymi stronami WWW. Za pomocą kilku poleceń języka PHP można uzyskać możliwość odczytywania danych z bazy, filtrowania tych danych według zadanych kryteriów, dodawania nowych rekordów lub modyfikowania już istniejących, a na koniec usuwania wybranych rekordów danych.

Zanim jednak będzie można z poziomu programu PHP odwołać się do danych zapisanych w bazie, należy nawiązać połączenie z serwerem bazy danych. Odpowiada za to instrukcja `mysql_connect()`, której parametrami są:

- nazwa lub adres IP komputera-serwera,
- nazwa konta użytkownika,
- hasło konta użytkownika.

Zwracana wartość to identyfikator połączenia. Jeżeli będzie równy `false`, połączenie nie mogło zostać nawiązane. Kod nawiązujący połączenie to zatem:

```
$baza = mysql_connect('localhost', 'baza', 'helion');
if ($baza === false) die('Nie można było nawiązać połączenia
z bazą '
                         .'z powodu błędu: ' . mysql_error());
```

Funkcja `mysql_error()` zwraca tekstowy opis ostatnio wykrytego błędu w komunikacji z bazą danych. W razie problemów z połączeniem instrukcja `die()` spowoduje przerwanie działania całego programu („śmierć" strony WWW: stąd nazwa funkcji) i wypisanie w ramach strony komunikatu o błędzie uzupełnionego o kod błędu odczytany z bazy.

Wskazówka

Ze względów bezpieczeństwa dobrze jest błędy zwracane przez funkcję `mysql_error()` wyświetlać na stronie tylko w czasie pracy nad kodem. Gdy strona działa już poprawnie, należy usunąć komunikaty, a zostawić tylko kod przerywający działanie programu. Na podstawie komunikatów błędów włamywacz może bowiem się domyślić, jaka technika włamania byłaby odpowiednia w danym przypadku.

Gdy zostanie już nawiązane połączenie z bazą danych, należy zdefiniować sposób kodowania przesyłanych znaków. Najlepiej jest wybrać uniwersalny standard kodowania UTF-8:

```
mysql_query('SET NAMES "utf8"');
```

Teraz musimy zadecydować, której bazy będziemy używać w trakcie całego połączenia. Oczywiście, decyzję można zmieniać w czasie pracy (a nawet selektywnie wybierać tabele z dowolnych baz), jednak w większości przypadków jednokrotne dokonanie wyboru zaraz po nawiązaniu połączenia całkowicie wystarcza. Do wyboru bazy danych służy instrukcja `mysql_select_db()`, której parametrem jest nazwa bazy danych; zwracana wartość oznacza, czy wybór się udał (`true`), czy też był z jakiegoś powodu niemożliwy (`false`):

```
$ok = mysql_select_db('szkola');
if ($ok === false) die('Nie można było wybrać bazy danych '
                       .'z powodu błędu: ' . mysql_error());
```

Teraz można już wydawać polecenia odczytujące, dodające lub modyfikujące rekordy danych; zostaną one omówione w kolejnych punktach. Na koniec pracy należy jednak zamknąć połączenie z bazą. Służy do tego instrukcja `mysql_close()`:

```
mysql_close($baza);
```

Cały szkielet programu korzystającego z bazy danych będzie zatem wyglądał następująco:

```
$baza = mysql_connect('localhost', 'baza', 'helion');
if ($baza === false) die('Nie można było nawiązać połączenia
z bazą '
                        .'z powodu błędu: ' . mysql_error());
$ok = mysql_select_db('szkola');
if ($ok === false) die('Nie można było wybrać bazy danych '
                      .'z powodu błędu: ' . mysql_error());

// --- tutaj instrukcje korzystające z bazy danych ---

mysql_close($baza);
```

Szkielet ten nie będzie już powtarzany w kolejnych ćwiczeniach: po prostu instrukcje korzystające z bazy danych muszą znaleźć się między sekwencją instrukcji nawiązujących połączenie z bazą a instrukcją `mysql_close()` zamykającą to połączenie.

26. Jak umieszczać na stronie WWW **dane pochodzące z bazy?**

Wbrew pozorom odczytanie danych z bazy i ich wyświetlenie zazwyczaj wymaga znacznie więcej kodu niż pozostałe, na pierwszy rzut oka bardziej skomplikowane operacje. Gdy jednak pozna się schemat wykorzystywany za każdym razem, gdy chce się wyświetlić dane z bazy na stronie, ilość kodu przestaje mieć znaczenie.

Pobranie danych wymaga przesłania bazie danych **zapytania** (ang. *query*) SQL. W odpowiedzi na zapytanie baza danych odsyła zbiór rekordów spełniających to zapytanie. Zadaniem programu jest odczytanie każdego rekordu (jeden po drugim) — to, co program zrobi z tymi rekordami, jest już kwestią fantazji programisty.

Zacznijmy od przygotowania zapytania. Zapytanie to po prostu tekst polecenia przesyłanego bazie danych; dlatego przygotujemy je w zmiennej tekstowej tak, jakbyśmy zapisywali kawałek tekstu w pamięci. Zapytanie może mieć na przykład następującą postać:

```
$zapytanie = 'SELECT imie, nazwisko, telefon FROM osoby';
```

Zapytanie to czyta się w następujący sposób: pobierz (`SELECT`) kolumny `imie`, `nazwisko` i `telefon` z tabeli (`FROM`) o nazwie `osoby`, z aktualnie wybranej bazy danych.

Teraz trzeba **wykonać** to zapytanie. Służy do tego instrukcja `mysql_query()`, pobierająca jako parametr tekst zapytania, a zwracająca wynik lub — w przypadku błędu — wartość logiczną `false`:

```
$odpowiedz = mysql_query($zapytanie);
if ($odpowiedz === false) die('Nie można było odebrać danych
z bazy '
                              .'z powodu błędu: ' . mysql_
error());
```

W tym momencie można już korzystać z odczytanych danych. Oczywiście, zajmują one nieco pamięci, gdy więc już nie będą potrzebne (na przykład wszystkie zostaną wyświetlone na stronie), należy zwolnić pamięć odpowiedzi za pomocą instrukcji `mysql_free_result()`:

```
mysql_free_result($odpowiedz);
```

Interpretację odpowiedzi serwera trzeba realizować w pętli, by po kolei — rekord po rekordzie — odczytywać przesłane dane i wyświetlać je na stronie. Zacznijmy od przygotowania szkicu tabeli:

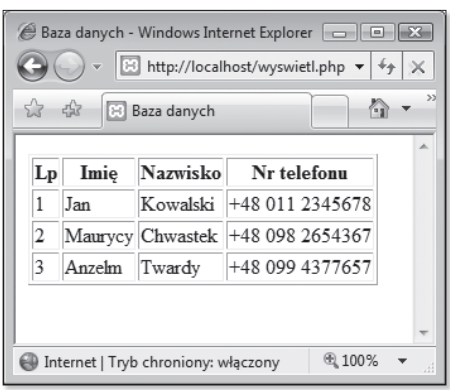
```
<table border="1">
 <thead>
  <tr>
   <th>Lp</th>
   <th>Imię</th>
   <th>Nazwisko</th>
   <th>Nr telefonu</th>
  </tr>
 </thead>
 <tbody>
 </tbody>
</table>
```

Teraz wewnątrz elementu `<tbody>`, stanowiącego „treść" naszej tabeli, musimy wstawić fragment kodu PHP odczytującego dane zwrócone jako rezultat zapytania:

```
<tbody>
<?php
$lp = 0;
while ($rekord = mysql_fetch_assoc($odpowiedz)) {
   $lp = $lp + 1;
   $imie = $rekord['imie'];
   $nazwisko = $rekord['nazwisko'];
   $nrtelefonu = $rekord['telefon'];
   echo '  <tr>'."\r\n";
   echo '   <td>'.$lp.'</td>'."\r\n";
   echo '   <td>'.$imie.'</td>'."\r\n";
   echo '   <td>'.$nazwisko.'</td>'."\r\n";
   echo '   <td>'.$nrtelefonu.'</td>'."\r\n";
   echo '  </tr>'."\r\n";
}
?>
 </tbody>
```

Instrukcja `mysql_fetch_assoc()` pobiera kolejny rekord odpowiedzi i umieszcza go w tablicy języka PHP, pozwalając odwoływać się do poszczególnych elementów za pomocą nazw pól — znakomicie upraszcza to programowanie. Pętla `while` będzie działała tak długo, aż kolejny odczytany rekord nie będzie miał wartości logicznej `false` oznaczającej, że napotkany został koniec danych. Wewnątrz pętli następuje przepisanie zawartości poszczególnych pól, przechowywanej w tablicy `$rekord`, do samodzielnych zmiennych (nie jest to konieczne — tutaj kod został przygotowany z myślą o większej czytelności, a nie wydajności czy zwartości). Następnie generowany jest kawałek kodu XHTML tworzący nowy wiersz tabeli.

Po zapisaniu tego fragmentu i wyświetleniu strony zobaczysz, że program faktycznie działa i wyświetla na Twojej własnej stronie WWW dane pobrane z poważnej bazy danych SQL (rysunek 26.1). Czy było to takie trudne? Chyba nie.



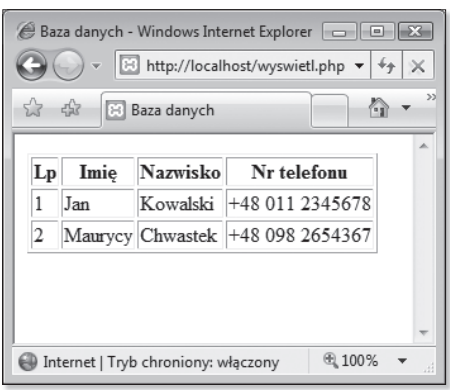
Rysunek 26.1. Działająca strona WWW pobiera dane z bazy danych

Spróbujmy teraz zmodyfikować zapytanie tak, aby baza filtrowała wyniki i wyświetlała wyłącznie dane uczniów, ignorując nauczycieli i pracowników technicznych. Wymaga to tylko drobnej zmiany w tekście zapytania SQL:

```
$zapytanie = 'SELECT imie, nazwisko, telefon FROM osoby WHERE
stanowisko="u"';
```

Zapytanie (wraz z dopisanym, pogrubionym powyżej fragmentem) czytamy: pobierz (`SELECT`) kolumny `imie`, `nazwisko` i `telefon` z tabeli (`FROM`) o nazwie `osoby`, uwzględniając tylko te rekordy (`WHERE`), w których wartość pola `stanowisko` jest równa `u`.

Jak widać, większość „czarnej roboty" można zrzucić na bazę danych (rysunek 26.2). Jeżeli zastanawiało Cię kiedyś, ile pracy musieli włożyć programiści dużych serwisów internetowych, by oferowały one możliwość przeglądania, sortowania i filtrowania wielkich zbiorów danych, teraz masz już odpowiedź: wymagało to dopisania kilku wierszy kodu, modyfikujących treść zapytania SQL. Resztę pracy wykonuje sama baza danych.

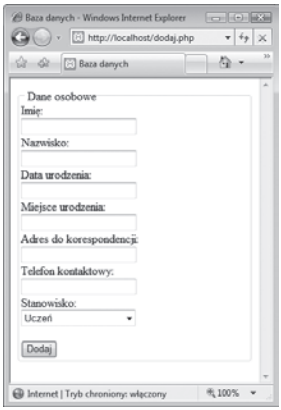


Rysunek 26.2. Ta sama strona po dodaniu kilku liter w tekście zapytania — już potrafi filtrować dane pobierane z bazy

27. Jak wprowadzać za pomocą strony WWW **nowe dane do bazy?**

Dodanie informacji do bazy danych okazuje się całkiem proste — najwięcej pracy trzeba włożyć w opracowanie formularza XHTML, w którym użytkownik strony będzie wpisywał informacje. Zacznijmy zatem od przygotowania takiego formularza umożliwiającego dodawanie nowych osób do bazy:

```
<form name="dodawanie" method="POST">
 <fieldset>
  <legend>Dane osobowe</legend>
  <label for="imie">Imię:</label><br />
  <input name="imie" id="imie" maxlength="50" /><br />
  <label for="nazwisko">Nazwisko:</label><br />
  <input name="nazwisko" id="nazwisko" maxlength="80" /><br />
  <label for="dataurodzenia">Data urodzenia:</label><br />
  <input name="dataurodzenia" id="dataurodzenia" maxlength="20"
/><br />
  <label for="miejsceurodzenia">Miejsce urodzenia:</label><br />
  <input name="miejsceurodzenia" id="miejsceurodzenia"
maxlength="40" /><br />
  <label for="adreskor">Adres do korespondencji:</label><br />
  <input name="adreskor" id="adreskor" maxlength="100" /><br />
  <label for="telefon">Telefon kontaktowy:</label><br />
  <input name="telefon" id="telefon" maxlength="20" /><br />
  <label for="stanowisko">Stanowisko:</label><br />
  <select name="stanowisko" id="stanowisko">
   <option value="u">Uczeń</option>
   <option value="n">Nauczyciel</option>
   <option value="t">Pracownik techniczny</option>
  </select><br /> <br />
  <input type="submit" value="Dodaj" />
 </fieldset>
</form>
```



Rysunek 27.1. Gotowy formularz XHTML

Po kliknięciu przycisku *Dodaj* na formularzu (rysunek 27.1) wywołana zostanie ta sama strona PHP, jednak tym razem zostaną przesłane do niej informacje zawarte w formularzu. Musimy zatem zareagować na otrzymane dane i wykonać zapytanie SQL, którego rezultatem będzie dodanie nowego rekordu.

Wskazówka

Może się wydawać, że zapytanie SQL powinno tylko **pytać** serwer bazy danych o informacje. Zapytaniem SQL nazywamy jednak **każdą** operację zlecaną bazie danych, niezależnie od tego, czy pytamy bazę danych o informacje, czy też umieszczamy w niej dane lub modyfikujemy istniejące rekordy.

Na początku strony PHP sprawdzimy zatem, czy użytkownik przesłał choćby samo imię. Jeśli tak — nawiążemy połączenie z bazą danych i przygotujemy się do wysyłki. To ten sam kod, który wprowadzaliśmy przy pobieraniu danych z bazy:

```
<?php
if (isset($_POST['imie'])) {
   $baza = mysql_connect('localhost', 'baza', 'helion');
   if ($baza === false) die('Nie można było nawiązać połączenia
z bazą '
                            .'z powodu błędu: ' . mysql_error());
   mysql_query('SET NAMES "utf8"');
   $ok = mysql_select_db('szkola');
   if ($ok === false) die('Nie można było wybrać bazy danych '
                          .'z powodu błędu: ' . mysql_error());

   // --- tutaj będą instrukcje obsługujące wysyłkę danych ---

   mysql_close($baza);
}
?>
```

Teraz wystarczy przygotować tekst zapytania SQL i zrealizować to zapytanie. Niestety, musimy przy okazji zabezpieczyć też naszą bazę przed atakami typu *SQL Injection*, które mogą spowodować niewyobrażalne straty informacji lub umożliwić nieuprawniony dostęp do danych. Warto też przy okazji sprawdzić, czy użytkownik nie zapomniał wypełnić jednego z pól: lepiej nakazać mu się cofnąć i sprawdzić dane, niż dodać do bazy niekompletny wpis. Zacznijmy zatem od przepisania pól do zmiennych pomocniczych:

```
$imie = $_POST['imie'];
$nazwisko = $_POST['nazwisko'];
$dataurodzenia = $_POST['dataurodzenia'];
$miejsceurodzenia = $_POST['miejsceurodzenia'];
$adreskor = $_POST['adreskor'];
$telefon = $_POST['telefon'];
$stanowisko = $_POST['stanowisko'];
```

Teraz sprawdzimy wszystkie pola (zasadniczo zatem zmienne). Nie trzeba sprawdzać pola `telefon`, gdyż w zasadzie osoba może nie posiadać telefonu, oraz pola `stanowisko`, gdyż powinno zostać wybrane z listy (zresztą sama baza danych odrzuci niepoprawne wartości tego pola). Resztę trzeba zweryfikować:

```
if (($imie == '') || ($nazwisko == '') || ($dataurodzenia ==
'')
|| ($miejsceurodzenia == '') || ($adreskor == '')) {
   echo '<p style="color: red;">Nie wypełniłeś jednego
 z pól!</p>';
   echo '<p>Cofnij się i popraw swoją omyłkę.</p>';
} else {
   // --- tutaj wysyłka danych do bazy ---
}
```

Sprawdzamy tutaj, czy którekolwiek z pól jest puste. Jeżeli choć jedno okaże się niewypełnione, wyświetlamy komunikat o błędzie i kontynuujemy wyświetlanie pustego formularza (pozostawiam Tobie zmodyfikowanie strony tak, by wprowadzone wcześniej wartości zostały ponownie wyświetlone w polach formularza). Jeżeli wszystko będzie w porządku, przejdziemy do kodu dodającego wpis do bazy.

Kolejny krok będzie polegał na zabezpieczeniu się przed atakami *SQL Injection*:

```
$imie = mysql_real_escape_string($imie);
$nazwisko = mysql_real_escape_string($nazwisko);
$dataurodzenia = mysql_real_escape_string($dataurodzenia);
$miejsceurodzenia = mysql_real_escape_string($miejsceurodz
enia);
$adreskor = mysql_real_escape_string($adreskor);
$telefon = mysql_real_escape_string($telefon);
$stanowisko = mysql_real_escape_string($stanowisko);
```

Teraz dopiero możemy przystąpić do stworzenia zapytania SQL:

```
$zapytanie = 'INSERT INTO osoby SET '
            .'imie="'.$imie.'", '
            .'nazwisko="'.$nazwisko.'", '
            .'dataurodzenia="'.$dataurodzenia.'", '
            .'miejsceurodzenia="'.$miejsceurodzenia.'", '
            .'adreskor="'.$adreskor.'", '
            .'telefon="'.$telefon.'", '
            .'stanowisko="'.$stanowisko.'"';
```

Zapytanie może się wydawać na pierwszy rzut oka skomplikowane, ale jest to spowodowane wyłącznie dużą liczbą operacji łączenia tekstu i zmiennych. Tak naprawdę zapytanie przybierze następującą postać:

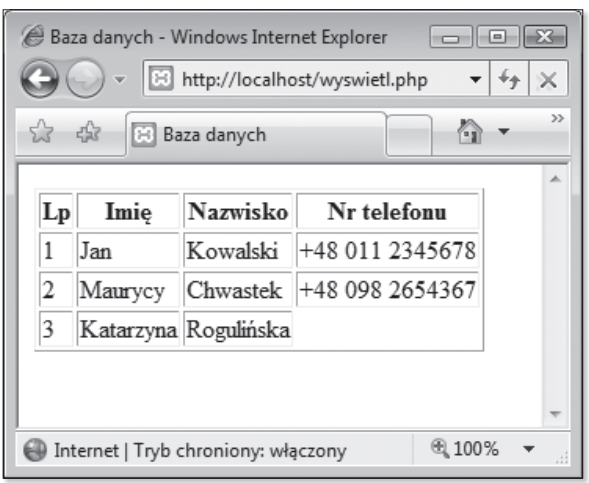
```
INSERT INTO osoby SET imie="Jan", nazwisko="Kowalski",
dataurodzenia="1973-08-11", ...
```

co czytamy: wstaw (`INSERT`) do tabeli (`INTO`) o nazwie `osoby` nowy rekord, ustawiając (`SET`) wartości pól: `imie` na „Jan”, `nazwisko` na „Kowalski”, `dataurodzenia` na „1973-08-11”…

Teraz wystarczy wykonać zapytanie i sprawdzić, czy się powiodło. Nawet nie trzeba zwalniać pamięci — zapytanie `INSERT` nie zwraca (poza ewentualną informacją o błędzie) wyniku, o który należałoby się martwić:

```
        $ok = mysql_query($zapytanie);
        if ($ok === false) die('Nie można było dodać danych do bazy
'
                                .'z powodu błędu: ' . mysql_error());
```

Gotowe! Gdy teraz wypełnimy formularz i klikniemy *Dodaj*, pozornie nic się nie stanie, jednak wystarczy ponownie odwiedzić stronę wyświetlającą zawartość bazy, by zauważyć, że został do niej dopisany nowy rekord (rysunek 27.2).



Rysunek 27.2. W tabeli danych pojawił się nowy wpis, wprowadzony naszym własnym formularzem

Ze względu na to, że poskładanie kodu strony z powyższych fragmentów może być kłopotliwe, przyda się na pewno kompletna jego treść. To dużo kodu — jednak głównie ze względu na powtarzalność (obsługujemy wiele pól danych) oraz zabezpieczenia. Samej obsługi bazy danych jest tutaj niewiele:

```
<html>
<head><title>Baza danych</title></head>
<body>

<?php
if (isset($_POST['imie'])) {
   $imie = $_POST['imie'];
   $nazwisko = $_POST['nazwisko'];
   $dataurodzenia = $_POST['dataurodzenia'];
   $miejsceurodzenia = $_POST['miejsceurodzenia'];
   $adreskor = $_POST['adreskor'];
   $telefon = $_POST['telefon'];
   $stanowisko = $_POST['stanowisko'];
   if (($imie == '') || ($nazwisko == '') || ($dataurodzenia == '')
   || ($miejsceurodzenia == '') || ($adreskor == '')) {
      echo '<p style="color: red;">Nie wypełniłeś jednego z pól!</p>';
      echo '<p>Cofnij się i popraw swoją omyłkę.</p>';
   } else {
      $imie = mysql_real_escape_string($imie);
      $nazwisko = mysql_real_escape_string($nazwisko);
      $dataurodzenia = mysql_real_escape_string($dataurodzenia);
      $miejsceurodzenia = mysql_real_escape_string($miejsceurodzenia);
      $adreskor = mysql_real_escape_string($adreskor);
      $telefon = mysql_real_escape_string($telefon);
      $stanowisko = mysql_real_escape_string($stanowisko);
      $baza = mysql_connect('localhost', 'baza', 'helion');
      if ($baza === false) die('Nie można było nawiązać połączenia z bazą '
                               .'z powodu błędu: ' . mysql_error());
      mysql_query('SET NAMES "utf8"');
      $ok = mysql_select_db('szkola');
      if ($ok === false) die('Nie można było wybrać bazy danych '
                             .'z powodu błędu: ' . mysql_error());
```

```
        $zapytanie = 'INSERT INTO osoby SET '
                    .'imie="'.$imie.'", '
                    .'nazwisko="'.$nazwisko.'", '
                    .'dataurodzenia="'.$dataurodzenia.'", '
                    .'miejsceurodzenia="'.$miejsceurodzenia.'", '
                    .'adreskor="'.$adreskor.'", '
                    .'telefon="'.$telefon.'", '
                    .'stanowisko="'.$stanowisko.'"';
        $ok = mysql_query($zapytanie);
        if ($ok === false) die('Nie można było dodać danych do bazy '
                               .'z powodu błędu: ' . mysql_error());
        mysql_close($baza);
    }
}
?>

<form name="dodawanie" method="POST">
 <fieldset>
  <legend>Dane osobowe</legend>
  <label for="imie">Imię:</label><br />
  <input name="imie" id="imie" maxlength="50" /><br />
  <label for="nazwisko">Nazwisko:</label><br />
  <input name="nazwisko" id="nazwisko" maxlength="80" /><br />
  <label for="dataurodzenia">Data urodzenia:</label><br />
  <input name="dataurodzenia" id="dataurodzenia" maxlength="20" /><br />
  <label for="miejsceurodzenia">Miejsce urodzenia:</label><br />
  <input name="miejsceurodzenia" id="miejsceurodzenia" maxlength="40" /><br />
  <label for="adreskor">Adres do korespondencji:</label><br />
  <input name="adreskor" id="adreskor" maxlength="100" /><br />
  <label for="telefon">Telefon kontaktowy:</label><br />
  <input name="telefon" id="telefon" maxlength="20" /><br />
  <label for="stanowisko">Stanowisko:</label><br />
```

Ciąg dalszy z poprzedniej strony:

```
  <select name="stanowisko" id="stanowisko">
   <option value="u">Uczeń</option>
   <option value="n">Nauczyciel</option>
   <option value="t">Pracownik techniczny</option>
  </select><br /> <br />
  <input type="submit" value="Dodaj" />
 </fieldset>
</form>

</body>
</html>
```