PHP i MySQL. Witryna WWW oparta na bazie danych. Wydanie III

Autor: Kevin Yank
Tłumaczenie: Sławomir Dzieniszewski, Paweł Janociński
ISBN: 83-7361-967-4
Tytuł oryginału: Build Your Own Database Driven Website Using PHP and MySQL, 3rd Edition
Format: B5, stron: 336

Zbuduj witrynę WWW, do której użytkownicy będą powracać codziennie

- Zainstaluj i skonfiguruj PHP i MySQL
- Poznaj zasady programowania w języku PHP
- Zaprojektuj bazę danych dla witryny WWW
- Wyświetl artykuły z bazy danych w oknie przeglądarki internetowej

Co zrobić, żeby wśród setek tysięcy witryn WWW użytkownicy zapamiętali właśnie naszą? Co sprawi, że będą do niej wracać? Atrakcyjny projekt graficzny to tylko jeden z czynników wpływających na odbiór witryny przez odwiedzających. Nawet najbardziej profesjonalnie zaprojektowana grafika nie przyciągnie internautów na stronę, na której dzień po dniu będą znajdować te same informacje. W jaki sposób rozwiązać kwestię aktualizowania treści witryny? Edycja plików HTML i mechanizmy SSI to rozwiązania zdające egzamin w przypadku niewielkich serwisów WWW. Dla większych witryn najlepszym rozwiązaniem jest przechowywanie treści stron w bazie danych i stworzenie mechanizmu pozwalającego na ich łatwą modyfikację.

Książka „PHP i MySQL. Witryna WWW oparta na bazie danych. Wydanie III" to przewodnik dla programistów, którzy chcą stworzyć własny system zarządzania treścią witryny WWW. Opisuje sposób realizacji takiego projektu za pomocą najpopularniejszej obecnie technologii – języka PHP i bazy danych MySQL. Przedstawia sposób instalacji PHP i MySQL-a w różnych systemach operacyjnych oraz podstawy korzystania z bazy danych i programowania w języku PHP. Nauczysz się przygotowywać strukturę tabel dla witryny WWW i tworzyć skrypty PHP, za pomocą których będziesz mógł edytować, formatować i wyświetlać artykuły z bazy danych w oknie przeglądarki WWW. Nauczysz się też administrować bazą danych MySQL i korzystać z mechanizmów obsługi sesji w PHP.

- Instalacja PHP i MySQL-a w Windows, Linuksie i Mac OS X
- Praca z MySQL-em
- Podstawowe zasady programowania w PHP
- Projektowanie relacyjnej bazy danych dla witryny WWW
- Tworzenie systemu edycji artykułów
- Formatowanie tekstów i wyświetlanie ich na stronie WWW
- Budowanie złożonych zapytań w języku SQL
- Korzystanie z danych binarnych w MySQL-u
- Stosowanie mechanizmów obsługi sesji i cookies w PHP

Jeśli chcesz, aby artykuły na Twojej witrynie WWW były zawsze aktualne, wykorzystaj system zarządzania treścią, który samodzielnie stworzysz.

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

Spis treści

Rozdział 4.

Publikowanie w sieci WWW danych przechowywanych w bazie MySQL

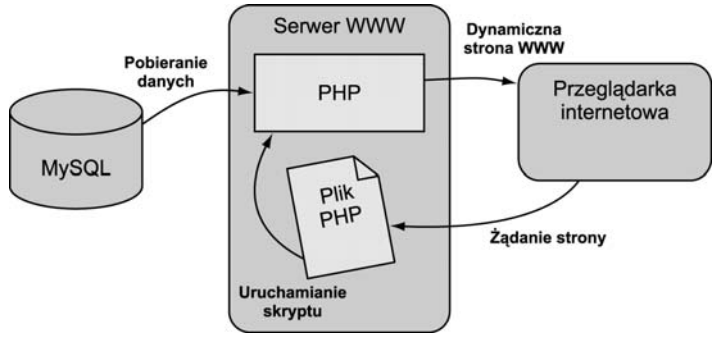
Nareszcie, właśnie na to czekaliśmy! W tym rozdziale dowiemy się, jak pobierać informacje przechowywane w bazie danych i wyświetlać je na stronie WWW, aby mieli do nich dostęp wszyscy użytkownicy. Do tej pory zainstalowaliśmy i nauczyliśmy się podstaw działania MySQL, systemu zarządzania relacyjnymi bazami danych oraz PHP, języka skryptowego działającego po stronie serwera. Teraz dowiemy się, jak połączyć oba te narzędzia, by za ich pomocą przygotować witrynę WWW opartą na bazie danych!

Rzut oka na podstawowy mechanizm

Zanim przystąpimy do dalszych rozważań, warto przypomnieć sobie, jaki jest nasz główny cel. Mamy do dyspozycji dwa wspaniałe narzędzia: język skryptowy PHP i system zarządzania bazami danych MySQL. Pora teraz na naukę, jak połączyć je ze sobą.

Podstawowym celem tworzenia witryny WWW opartej na bazie danych jest przechowywanie zawartości witryny w bazie danych, tak aby można ją było stamtąd pobierać dynamicznie, tworząc na poczekaniu strony WWW, do których użytkownicy witryny zyskają dostęp za pomocą zwykłej przeglądarki internetowej. Tak więc po jednej stronie mamy odwiedzającego, który pobiera stronę za pośrednictwem przeglądarki internetowej i spodziewa się otrzymać standardowy dokument HTML. A po drugiej stronie zawartość witryny, rozproszoną po jednej lub więcej tabelach w bazie danych MySQL, która to baza akceptuje tylko polecenia w formie zapytań SQL.

Tak jak zostało to przedstawione na rysunku 4.1, język skryptowy PHP pełni funkcję pośrednika, który potrafi posługiwać się oboma językami. Najpierw przetwarza żądanie strony i pobiera dane z bazy MySQL, a następnie porządkuje je dynamicznie, przygotowując ładnie sformatowaną stronę HTML, której oczekuje przeglądarka. Język PHP pozwala przygotować wszelkie aspekty związane z prezentacją naszej witryny (zaprojektować oprawę graficzną i układ strony) w formie „szablonów" napisanych w zwykłym kodzie HTML. Kiedy przystępujemy do wypełnienia tych szablonów treścią, znowu wykorzystujemy kod PHP, by połączyć się z bazą danych MySQL i — używając zapytań SQL takich jak te, które testowaliśmy na tabeli `kawaly` w rozdziale 2., „Wprowadzenie do systemu MySQL" — pobrać odpowiednią zawartość i wyświetlić ją we właściwych miejscach szablonów.



Rysunek 4.1. *Schemat pokazujący, w jaki sposób język PHP pobiera dane z bazy MySQL, by przygotować strony WWW*

Aby przedstawić to jeszcze przejrzyściej, wyjaśnijmy, co dzieje się, gdy użytkownik odwiedza naszą witrynę WWW opartą na bazie danych:

1. Przeglądarka internetowa odwiedzającego żąda strony WWW, wysyłając standardowy adres URL.
2. Program serwera WWW (Apache, IIS czy jakiegokolwiek innego) ustala, że żądany plik jest skryptem PHP i serwer rozpoczyna interpretowanie pliku, używając do tego celu wbudowanego modułu (dodatku) obsługującego języka PHP.
3. Odpowiednie polecenia PHP (których musimy się dopiero nauczyć) łączą się z bazą danych MySQL i żądają zawartości, która powinna pojawić się na przygotowywanej stronie WWW.
4. Baza danych MySQL odpowiada, przesyłając skryptowi PHP żądaną zawartość.
5. Skrypt PHP zapisuje pobraną zawartość w jednej lub więcej zmiennych PHP, a następnie za pomocą znanej już nam instrukcji `echo` umieszcza ją w odpowiednich miejscach strony WWW.
6. Dodatek obsługujący PHP kończy pracę, zwracając przygotowaną przez siebie stronę kodu HTML serwerowi WWW.
7. Serwer WWW przesyła tę stronę z kodem HTML do przeglądarki internetowej, tak jakby to był zwykły plik HTML. Nie przesyła jednak gotowego, statycznego pliku HTML, lecz kod zwrócony przez moduł interpretujący skrypt PHP.

Łączenie się z bazą MySQL za pomocą PHP

Zanim spróbujemy pobrać z bazy danych MySQL treść, która zostanie opublikowana na stronie WWW, musimy dowiedzieć się, jak wewnątrz skryptu PHP ustanowić połączenie z systemem MySQL. Jak pamiętamy, w rozdziale 2., „Wprowadzenie do systemu MySQL", korzystaliśmy z programu `mysql`, który pozwalał nam nawiązywać takie połączenie wprost z wiersza poleceń. PHP jednak nie musi przywoływać żadnego specjalnego programu, ponieważ obsługa łączenia się z serwerem MySQL jest bezpośrednio wbudowana w ten język. Połączenie tworzy wbudowana funkcja `mysql_connect`.

```
mysql_connect(adres, nazwa_użytkownika, hasło)
```

W tym wzorcu polecenie *adres* to adres IP lub nazwa hosta komputera, na którym działa serwer MySQL (`'localhost'` jeśli funkcjonuje na tym samym komputerze co serwer WWW), a *nazwa_użytkownika* i *hasło* to odpowiednia nazwa użytkownika i hasło, których używamy do łączenia się z serwerem MySQL, tak jak w rozdziale 2., „Wprowadzenie do systemu MySQL".

Jak już wspomniano, gdy funkcje PHP są przywoływane, zazwyczaj zwracają jakąś wartość. Czytelnikom, którzy nie pamiętają, przypominam, że chodzi o szczegół wymieniony w rozdziale 3., „Wprowadzenie do języka PHP", gdy po raz pierwszy przywoływaliśmy funkcję. Poza wykonywaniem pewnego użytecznego zadania większość funkcji zwraca również jakąś wartość. Wartość tę można zachować w zmiennej, by później korzystać z niej w skrypcie. Przedstawiona tutaj funkcja `mysql_connect` na przykład zwraca liczbę, która pozwala zidentyfikować właśnie ustanowione połączenie. Ponieważ zamierzamy korzystać później z tego połączenia, powinniśmy zachować tę wartość. Oto praktyczny przykład takiego polecenia połączenia się z serwerem MySQL:

```
$dbcnx = mysql_connect('localhost', 'root', 'mypasswd');
```

Jak już wspomniano, wartości tych trzech parametrów funkcji mogą się różnić w zależności od serwera MySQL, z którym się łączymy. Warto zwrócić uwagę na to, że wartość zwrócona przez funkcję `mysql_connect` (którą będziemy nazywać *identyfikatorem połączenia*) przechowywana jest w zmiennej `$dbcnx`.

Ponieważ serwer MySQL jest zupełnie niezależnym programem, musimy uwzględnić sytuację, w której serwer będzie z jakiegoś powodu niedostępny lub nieosiągalny, na przykład dlatego, że wystąpiły jakieś problemy z połączeniem sieciowym lub użyliśmy niewłaściwej kombinacji nazwy użytkownika i hasła. W takich przypadkach funkcja `mysql_connect` nie zwróci identyfikatora połączenia (bo połączenie nie zostało nawiązane), lecz wartość `false` (fałsz). Dzięki temu możemy skorzystać z instrukcji `if` i odpowiednio zareagować, gdy nie uda się nawiązać połączenia:

```
$dbcnx = @mysql_connect('localhost', 'root', 'mypasswd');
if (!$dbcnx) {
  echo '<p>W tej chwili nie można nawiązać ' .
      'połączenia z serwerem bazy danych.</p>' );
  exit();
}
```

W przedstawionym kodzie pojawiły się trzy nowe, warte uwagi sztuczki. Po pierwsze, przed nazwą funkcji `mysql_connect` umieściliśmy symbol `@`. Wiele funkcji, w tym między innymi `mysql_connect`, w momencie niepowodzenie wyświetla automatycznie bardzo nieestetyczne komunikaty o błędach. Umieszczenie przed nazwą funkcji symbolu `@` — zwanego również *operatorem supresji błędów* (ang. *error suppression operator*) — nakazuje jej zakończyć działanie po cichu, bez wyświetlania standardowych komunikatów o błędach, dając nam tym samym szansę przygotowania własnego, bardziej przyjaznego komunikatu.

Drugi trik polega na umieszczeniu znaku wykrzyknika (`!`) przed nazwą zmiennej `$dbcnx` w warunku instrukcji `if`. Znak wykrzyknika jest *operatorem negacji* (ang. *negation operator*), który po prostu zamienia wartość logiczną `true` na wartość `false`, a wartość `false` na wartość `true`. Dzięki temu, jeśli nie uda się nawiązać połączenia i funkcja `mysql_connect` zwróci wartość `false`, wyrażenie `!$sbcnx` będzie miało wartość `true` (zostanie rozpoznane jako prawda) i uruchomione zostaną polecenia umieszczone w bloku instrukcji `if`. I przeciwnie, jeśli połączenie zostanie nawiązane, identyfikator połączenia przechowywany w zmiennej `$dbcnx` zostanie rozpoznany jako wartość `true` (w języku PHP za wartość „prawda” uznawane są wszystkie wartości różne od zera). Wyrażenie `!$dbcnx` zostanie więc rozpoznane jako fałsz, a zatem instrukcja `if` nie zostanie wykonana.

Ostatni z nowych trików to funkcja `exit`; jest ona jednocześnie pierwszym w tej książce przykładem funkcji, którą można przywoływać bez żadnych parametrów. Jeśli przywołamy ją nie podając żadnych parametrów, to PHP przerwie w tym miejscu odczytywanie strony. Jest to całkiem dobra reakcja na nieudaną próbę nawiązania połączenia z bazą danych, ponieważ w większości przypadków bez nawiązania połączenia i tak nie uda nam się wyświetlić na stronie żadnych użytecznych informacji.

Podobnie jak w rozdziale 2., „Wprowadzenie do systemu MySQL”, gdy już połączenie zostanie nawiązane, kolejnym krokiem będzie wybór bazy danych, z którą chcemy pracować. Załóżmy, że interesuje nas baza kawałów, którą tworzyliśmy w rozdziale 2., „Wprowadzenie do systemu MySQL”. Baza ta nosiła nazwę `ijdb`. Wybór bazy w kodzie PHP wymaga po prostu przywołania kolejnej funkcji:

```
mysql_select_db('ijdb', $dbcnx);
```

Skorzystaliśmy zatem ze zmiennej `$dbcnx`, przechowującej identyfikator połączenia z bazą danych, by poinformować funkcję, z którego połączenia z bazą danych powinna skorzystać. Prawdę powiedziawszy parametr ten jest opcjonalny. Jeśli go pominiemy, funkcja automatycznie skorzysta z identyfikatora ostatniego otwartego połączenia. Funkcja `mysql_select_db` zwraca wartość `true`, jeśli uda się jej połączyć z bazą i `false`, jeśli pojawią się jakieś błędy. Podobnie jak poprzednio wyrazem rozwagi będzie skorzystanie z instrukcji `if`, aby zareagować na ewentualne błędy:

```
if (!@mysql_select_db('ijdb')) {
  exit('<p>Nie można w tej chwili ' .
       'zlokalizować bazy kawałów.</p>');
}
```

Warto zauważyć, że tym razem zamiast przypisywać wartość zwróconą przez funkcję do zmiennej i potem sprawdzać, czy jest ona prawdą, czy fałszem, po prostu wykorzy-

stałem w warunku same przywołanie funkcji. Na pozór wygląda to trochę nietypowo, ale jest to dość często używany skrótowy zapis. Aby ustalić, czy warunek jest spełniony, czy nie, interpreter PHP wykona funkcję, a potem sprawdzi zwróconą przez nią wartość — czyli zrobi właśnie to, na czym nam zależy.

Kolejnym skrótem, który zastosowaliśmy, jest przywołanie funkcji `exit` z parametrem w postaci łańcucha tekstu. Jeśli przyzwiemy funkcję `exit` w ten sposób, zadziała ona podobnie jak instrukcja `echo`, z tą tylko różnicą, że zakończy wykonywanie skryptu po wyświetleniu tekstu. Jednym słowem przywołanie funkcji `exit` z łańcuchem tekstu jako parametrem jest odpowiednikiem użycia instrukcji `echo`, a następnie przywołania funkcji `exit` bez żadnych parametrów, tak jak w poprzednim przykładzie ilustrującym korzystanie z funkcji `mysql_connect`.

Gdy już ustanowimy połączenie i wybierzemy odpowiednią bazę danych, będziemy mogli skorzystać z danych przechowywanych w bazie.

Wysyłanie zapytań SQL za pomocą języka PHP

W rozdziale 2., „Wprowadzenie do systemu MySQL”, łączyliśmy się z serwerem bazy danych MySQL za pomocą programu o nazwie `mysql`, który pozwalał na wpisywanie zapytań (poleceń) SQL i natychmiastowe oglądanie ich wyników. W języku PHP istnieje całkiem podobny mechanizm, mianowicie funkcja `mysql_query`:

```
mysql_query(zapytanie[, identyfikator_połączenia])
```

Tutaj *zapytanie* będzie po prostu łańcuchem tekstu zawierającym polecenie SQL, które chcemy wykonać. Podobnie jak w przypadku funkcji `mysql_select_db`, parametr identyfikatora połączenia jest opcjonalny.

To, co ta funkcja zwróci, zależy już od rodzaju wysłanego zapytania. W przypadku większości poleceń SQL funkcja `mysql_query` zwraca wartość `true` lub `false`, by zasygnalizować (odpowiednio), czy zapytanie zostało wykonane z powodzeniem, czy też nie. Przeanalizujmy następujący przykład, w którym spróbujemy utworzyć tabelę `kawal`, taką jak w rozdziale 2., „Wprowadzenie do systemu MySQL”:

```
$sql = 'CREATE TABLE kawal (
      id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
      tekstkawalu TEXT,
      datakawalu DATE NOT NULL
    )';
if (@mysql_query($sql)) {
  echo '<p>Tabela kawal utworzona!</p>';
} else {
  exit('<p>Nie udało się utworzyć tabeli kawal: ' .
      mysql_error() . '</p>');
}
```

Ponownie zastosowaliśmy trik z użyciem operatora @, aby zablokować wszelkie automatyczne komunikaty o błędach zwracane przez funkcję mysql_query i zamiast tego wyświetlić bardziej przyjazny komunikat o błędach. Użyta tutaj funkcja mysql_error zwraca łańcuch tekstu, który opisuje ostatni komunikat o błędzie wysłany przez serwer MySQL.

W przypadku zapytań DELETE, INSERT i UPDATE (służą one do modyfikowania danych) system MySQL notuje również liczbę wierszy tabeli (pozycji), które zostały zmienione przez zapytanie. Rozważmy takie zapytanie SQL, którego używaliśmy już w rozdziale 2., „Wprowadzenie do systemu MySQL", by zmienić daty wszystkich kawałów, zawierających słowo „kurczak":

```
$sql = "UPDATE kawal SET datakawalu='1994-04-01'
    WHERE tekstkawalu LIKE '%kurczak%'";
```

Kiedy wykonamy to zapytanie, będziemy mogli skorzystać z funkcji mysql_affected_rows, by poznać liczbę wierszy, które zostały zmienione przez to zapytanie UPDATE:

```
if (@mysql_query($sql)) {
  echo '<p>Zapytanie UPDATE aktualizowało ' . mysql_affected_rows() .
      ' wierszy.</p>';
} else {
  exit('<p>Błąd podczas aktualizacji z pomocą UPDATE: ' . mysql_error() .
      '</p>');
}
```

Zapytania SELECT traktowane są trochę inaczej, ponieważ pobierają bardzo dużo danych i język PHP musi dostarczać jakichś sposobów obsługiwania tych informacji.

Obsługa zbiorów wyników zapytania SELECT

W przypadku większości zapytań SQL funkcja mysql_query zwracać będzie albo wartość true (prawda, w przypadku sukcesu), albo false (fałsz, w razie niepowodzenia). Gdy jednak wykonujemy zapytanie SELECT, to nie wystarczy. Jak pamiętamy, zapytania SELECT służą do oglądania danych przechowywanych w bazie danych. Dlatego też oprócz informacji, czy zapytanie zakończyło się powodzeniem, czy nie, język PHP musi również w jakiś sposób odebrać zwrócone wyniki zapytania. Dlatego też w przypadku zapytania SELECT funkcja mysql_query zwraca liczbę będącą identyfikatorem *zbioru wyników* (ang. *result set*), który zawiera wszystkie wiersze (pozycje, rekordy) zwrócone przez zapytanie. Jeśli zapytanie z jakichś powodów się nie powiedzie, zwracana jest wartość false.

```
$result = @mysql_query('SELECT tekstkawalu FROM kawal');
if (!$result) {
  exit('<p>Błąd podczas wykonywania zapytania: ' . mysql_error() .
      '</p>');
}
```

Zakładając, że w trakcie przetwarzania nie pojawił się żaden błąd, przedstawiony kod zapisze w zmiennej `$result` pewną liczbę. Liczba ta odpowiada zbiorowi wyników zawierającemu teksty wszystkich kawałów przechowywanych w tabeli `kawal`. Ponieważ nie ma praktycznie żadnych ograniczeń co do liczby dowcipów przechowywanych w bazie, zwrócony zbiór wyników może być całkiem pokaźny.

Jak już wspomniano, pętla `while` jest bardzo przydatną strukturą sterującą, gdy musimy pracować z dużymi zestawami danych. Oto szkic kodu, który umożliwi przetwarzanie jeden po drugim kolejnych wierszy w zbiorze wyników:

```
while ($row = mysql_fetch_array($result)) {
  // przetwarzaj odpowiednio wiersz...
}
```

Warunek wykorzystany w pętli `while` nie przypomina warunków, z którymi zetknęliśmy się do tej pory. Dlatego poświęcę mu kilka słów wyjaśnienia. Przyjrzyjmy się warunkowi tak, jak by był on osobną instrukcją:

```
$row = mysql_fetch_array($result);
```

Funkcja `mysql_fetch_array` przyjmuje jako parametr liczbę identyfikującą zbiór wyników (w tym przypadku przechowywaną w zmiennej `$result`) i zwraca kolejny wiersz ze zbioru wyników w postaci tablicy (więcej informacji na temat tablic można znaleźć w rozdziale 3., „Wprowadzenie do języka PHP”). Gdy wreszcie funkcja `mysql_fetch_array` dojdzie w zbiorze wyników do końca i nie znajdzie kolejnego wiersza, zwróci wartość `false`.

Powyższa instrukcja przypisuje zmiennej wiersza `$row` pewną wartość, ale jednocześnie cała instrukcja przyjmuje tę samą wartość. To pozwala nam użyć tej instrukcji jako warunku w pętli `while`. Ponieważ pętla `while` będzie wykonywana dopóty, dopóki warunek nie przyjmie wartości `false`, pętla ta będzie wykonywana tak długo, jak długo funkcja `mysql_fetch_array` zdoła pobierać kolejne wiersze, a za każdym nawrotem pętli zmiennej `$row` będzie przypisywana wartość kolejnego wiersza. Pozostaje nam jedynie znaleźć sposób na pobieranie w każdym nawrocie pętli wartości zapisanych w zmiennej tablicowej `$row`.

Wiersze wyników zwracane przez funkcję `mysql_fetch_array` przechowywane są w postaci tablic asocjacyjnych. Indeksami takiej tablicy będą nazwy kolejnych kolumn tabeli zwracanych w zbiorze wyników. Jeśli zmienna `$row` reprezentuje pojedynczy wiersz z naszego zbioru wyników, to zapis `$row['tekstkawalu']` odwoływać się będzie do wartości kolumny `tekstkawalu` w tym wierszu. Oto więc jak powinna wyglądać pętla `while`, jeśli chcielibyśmy wyświetlić tekst wszystkich kawałów przechowywanych w naszej bazie:

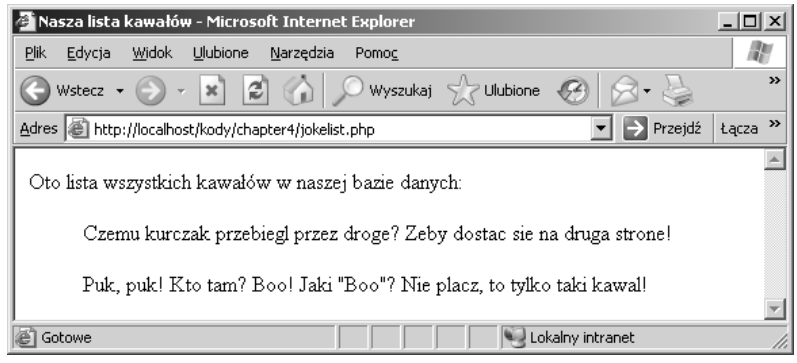
```
while ($row = mysql_fetch_array($result)) {
  echo '<p>' . $row['tekstkawalu'] . '</p>';
}
```

Podsumowując, poniżej został przedstawiony kompletny kod PHP strony WWW, która będzie się łączyć z naszą bazą danych, pobierać tekst wszystkich kawałów w bazie danych i wyświetlać je w osobnych akapitach HTML:

Listing 4.1. *jokelist.php*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Nasza lista kawałów</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-2" />
</head>
<body>
<?php

// Połącz się z serwerem bazy danych
$dbcnx = @mysql_connect('localhost', 'root', 'mypasswd');
if (!$dbcnx) {
  exit('<p>W tej chwili nie można nawiązać ' .
      'połączenia z serwerem bazy danych.</p>' );
}

// Wybierz bazę danych z kawałami
if (!@mysql_select_db('ijdb')) {
  exit('<p><p>Nie można w tej chwili ' .
      'zlokalizować bazy kawałów.</p>');
}

?>
<p>Oto lista wszystkich kawałów w naszej bazie danych:</p>
<blockquote>
<?php

// Zażądaj tekstu wszystkich kawałów
$result = @mysql_query('SELECT tekstkawalu FROM kawal');
if (!$result) {
  exit('<p> Błąd podczas wykonywania zapytania: ' . mysql_error() . '</p>');
}

// Wyświetl tekst każdego kawału w osobnym akapicie
while ($row = mysql_fetch_array($result)) {
  echo '<p>' . $row['tekstkawalu'] . '</p>';
}

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

Rysunek 4.2 pokazuje wygląd strony, gdy dodamy do naszej bazy kilka kawałów.



Rysunek 4.2. *Wszystkie perełki z mojego repertuaru w jednym miejscu!*

Wstawianie danych do bazy

W tej części rozdziału zostaną przedstawione narzędzia, które umożliwiają odwiedzającym dodawanie kolejnych kawałów do naszej bazy danych. Ambitni czytelnicy mogą podjąć próbę samodzielnego rozwiązania zadania, zanim jeszcze przystąpią do dalszej lektury. Zaprezentuję wprawdzie *trochę* nowego materiału, ale nasza aplikacja będzie głównie korzystać z rozwiązań, z którymi zetknęliśmy się już do tej pory.

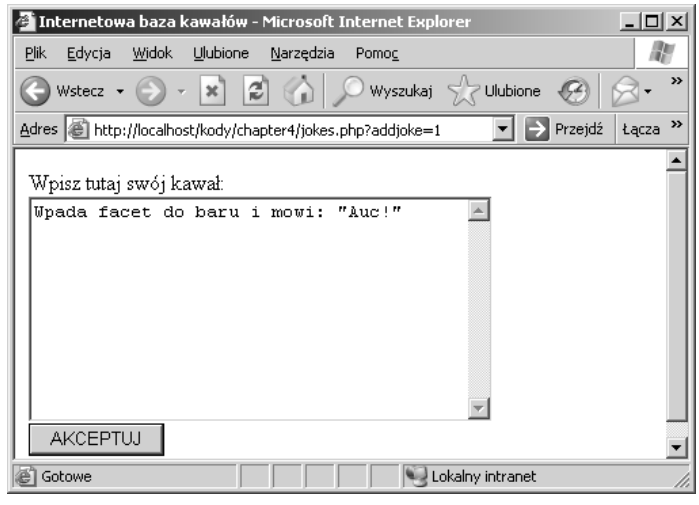
Aby umożliwić odwiedzającym dodawanie nowych kawałów do bazy, oczywiście potrzebny nam będzie odpowiedni formularz. Oto kod formularza, który spełni to zadanie:

Listing 4.2. *jokes.php (fragment)*

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<label>Wpisz tutaj swój kawał:<br />
<textarea name="tekstkawalu" rows="10" cols="40">
</textarea></label><br />
<input type="submit" value="AKCEPTUJ" />
</form>
```

Rysunek 4.3 pokazuje, jak formularz ten będzie wyglądać w oknie przeglądarki.

Tak jak poprzednio, formularz po zatwierdzeniu załaduje właściwie tę samą stronę (ponieważ w atrybucie `action` formularza, definiującym akcję podejmowaną po jego wypełnieniu, użyliśmy zmiennej `$_SERVER['PHP_SELF']` ładującej ten sam formularz jeszcze raz), z jedną tylko różnicą: do nowego żądania zostanie dodana pewna zmienna. Zmienna ta, o nazwie `joketext`, będzie zawierać tekst kawału, który został wpisany w polu tekstowym i pojawi się w tablicach `$_POST` i `$_REQUEST` automatycznie generowanych przez PHP.



Rysunek 4.3. *Kolejna perełka humoru wprowadzona do bazy*

Aby wstawić zatwierdzony kawał do bazy danych, skorzystamy z funkcji `mysql_query`, by za jej pomocą uruchomić zapytanie `INSERT`. Wykorzystuje ono wartość przechowywaną w polu `$_POST['tekstkawalu']`, by umieścić odpowiedni tekst w kolumnie `tekstkawalu` przywołanej w zapytaniu:

Listing 4.3. *jokes.php (fragment)*

```
if (isset($_POST['tekstkawalu'])) {
  $joketext = $_POST['tekstkawalu'];
  $sql = "INSERT INTO kawal SET
      tekstkawalu='$joketext',
      datakawalu=CURDATE()";
  if (@mysql_query($sql)) {
    echo '<p>Twój kawał został dodany.</p>';
  } else {
    echo '<p>Błąd podczas dodawania kawału: ' .
        mysql_error() . '</p>';
  }
}
```

Jedyna nowa sztuczka pojawiająca się w tym kodzie została wytłuszczona. Wykorzystujemy tutaj funkcję `CURDATE()` systemu MySQL, by przypisać kolumnie `datakawalu` bieżącą datę. System MySQL udostępnia wiele takich przydatnych funkcji, ale omówimy je dopiero wtedy, kiedy będą nam potrzebne. Dokładny opis funkcji oferowanych przez system MySQL można znaleźć w dodatku B, „Funkcje MySQL”.

Mamy już więc kod, który umożliwi użytkownikowi wpisywanie kawałów do naszej bazy danych. Pozostało nam jedynie dodać go w wygodny sposób do naszej strony prezentującej listę kawałów. Ponieważ większość użytkowników zapewne zechce tylko przejrzeć już wpisane kawały, nie będziemy zaśmiecać im strony wielkim, brzydkim formularzem, chyba że ktoś wprost wyrazi zainteresowanie możliwością dodania nowego dowcipu. Z tego powodu naszą aplikację najlepiej zaimplementować jako stronę wielozadaniową. Oto pełny kod skryptu:

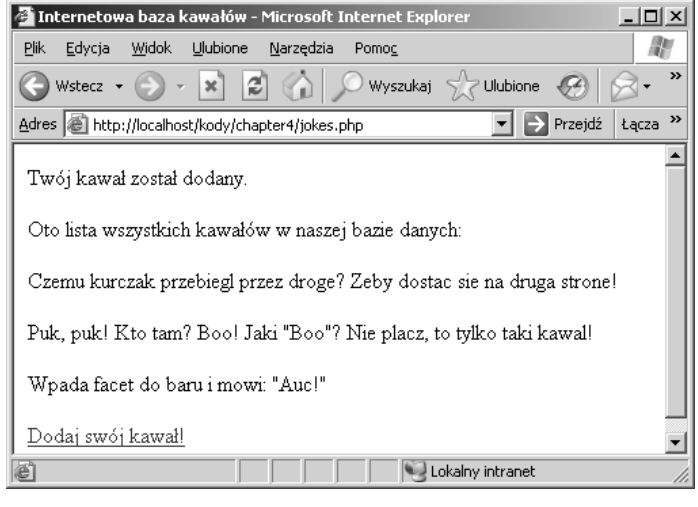
Listing 4.4. *jokes.php*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Internetowa baza kawałów</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-2" />
</head>
<body>

<?php if (isset($_GET['addjoke'])): // Użytkownik chce dodać kawał
?>

<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<label>Wpisz tutaj swój kawał:<br />
<textarea name="tekstkawalu" rows="10" cols="40">
</textarea></label><br />
<input type="submit" value="AKCEPTUJ" />
</form>

<?php else: // Jeśli nie, wyświetlamy domyślną stronę

  // Połącz się z serwerem bazy danych
  $dbcnx = @mysql_connect('localhost', 'root', 'mypasswd');
  if (!$dbcnx) {
    exit('<p>W tej chwili nie można nawiązać ' .
        'połączenia z serwerem bazy danych.</p>' );
  }

  // Wybierz bazę danych z kawałami
  if (!@mysql_select_db('ijdb')) {
    exit('<p><p>Nie można w tej chwili ' .
        'zlokalizować bazy kawałów.</p>');
  }

  // Jeśli użytkownik wprowadził kawał,
  // dodaj żart do bazy danych.
  if (isset($_POST['tekstkawalu'])) {
    $joketext = $_POST['tekstkawalu'];
    $sql = "INSERT INTO kawal SET
        tekstkawalu='$joketext',
        datakawalu=CURDATE()";
    if (@mysql_query($sql)) {
      echo '<p>Twój kawał został dodany.</p>';
    } else {
      echo '<p>Błąd podczas dodawania kawału: ' .
          mysql_error() . '</p>';
    }
  }

  echo '<p>Oto lista wszystkich kawałów w naszej bazie danych:</p>';

  // Zażądaj tekstu wszystkich kawałów
  $result = @mysql_query('SELECT tekstkawalu FROM kawal');
  if (!$result) {
    exit('<p> Błąd podczas wykonywania zapytania: ' . mysql_error() . '</p>');
```

```
    }

    // Wyświetl tekst każdego kawału w osobnym akapicie
    while ($row = mysql_fetch_array($result)) {
      echo '<p>' . $row['tekstkawalu'] . '</p>';
    }

    // To łącze po kliknięciu wyświetli stronę
    // z formularzem umożliwiającym dodanie kawału.
    echo '<p><a href="' . $_SERVER['PHP_SELF'] .
        '?addjoke=1">Dodaj swój kawał!</a></p>';

endif;
?>
</body>
</html>
```

Warto gwoli testu załadować tę stronę do swojej przeglądarki i spróbować dodać do bazy kawał lub dwa. Strona, która powinna się pojawić po dodaniu dowcipu, została pokazana na rysunku 4.4.

Rysunek 4.4. *Patrz mamciu! Żadnego SQL-a!*



I o to chodzi! Za pomocą pojedynczego pliku zawierającego dość prosty kod PHP możemy dodawać do naszej bazy MySQL nowe kawały i oglądać te, które już się tam znajdują.

Praca domowa

Nasza „praca domowa” polegać będzie na umieszczeniu obok każdego akapitu z kawałem łącza zatytułowanego *Usuń ten kawał*, które po kliknięciu usunie ten kawał z bazy danych i wyświetli nową, zaktualizowana listę dowcipów. Zanim do tego przystąpimy, kilka podpowiedzi:

- Można to nadal bez problemu zrobić na jednej wielozadaniowej stronie WWW.
- Konieczne będzie skorzystanie z polecenia `DELETE` języka SQL, za pomocą którego usuwaliśmy już obiekty z bazy w rozdziale 2. „Wprowadzenie do systemu MySQL”.
- I kwestia najtrudniejsza: aby usunąć ten a nie inny kawał, konieczne będzie jednoznaczne zidentyfikowanie go. Kolumna identyfikatora `id` w tabeli `kawal` służy właśnie do tego celu. Aby usunąć dowcip, należy wraz z żądaniem usunięcia kawału przesłać również jego identyfikator. Znakomitym miejscem na umieszczenie tej wartości jest łańcuch zapytania odpowiedniego łącza *Usuń*.

Ci, którym wydaje się, że już znają odpowiedź, albo też po prostu chcieliby znaleźć rozwiązanie, powinni po prostu przewrócić stronę. Powodzenia!

Podsumowanie

W tym rozdziale poznaliśmy kilka nowych funkcji języka PHP, które umożliwiają współpracę z serwerem relacyjnych baz danych MySQL. Korzystając z tych funkcji zbudowaliśmy naszą pierwszą witrynę WWW opartą na bazie danych, która publikowała w sieci naszą bazę danych kawałów `ijdb` oraz pozwalała użytkownikom na dodawanie do niej nowych żartów.

W rozdziale 5., zatytułowanym „Projektowanie relacyjnych baz danych”, powrócimy do wiersza poleceń systemu MySQL. Nauczymy się, jak realizować zasady dobrego projektowania baz danych oraz jak korzystać z bardziej zaawansowanych zapytań SQL, by wyświetlać na stronie bardziej złożone informacje oraz umożliwić naszym użytkownikom podpisywanie kawałów swoim imieniem!

Rozwiązanie naszej „pracy domowej”

Przedstawiam rozwiązanie „pracy domowej” zaproponowanej powyżej. Oto zmiany wprowadzone w poprzednim skrypcie PHP, które umożliwią dodanie łącza *Usuń ten kawał* obok każdego z kawałów na stronie:

- W poprzednim przykładzie przesyłaliśmy zmienną `addjoke` z łączem *Dodaj swój kawał!* umieszczonym u dołu strony, sygnalizując w ten sposób skryptowi PHP, że zamiast zwykłej listy kawałów powinien wyświetlić formularz umożliwiający dodanie kolejnego. W podobny sposób prześlemy zmienną `deletejoke` wraz z łączem *Usuń ten kawał*, by zasygnalizować, że dany kawał powinien zostać usunięty.
- Dla każdego kawału powinniśmy wraz z zawartością kolumny `tekstkawalu` pobierać z bazy danych wartość kolumny `id`, żeby wiedzieć, jaki identyfikator ma każdy z kawałów zapisanych w bazie i wyświetlonych na stronie.

- W zmiennej `$_GET['deletejoke']` zapiszemy identyfikator kawału, który usuwamy. W tym celu będziemy wstawiać do kodu HTML łącza *Usuń ten kawał* przy każdym z kawałów wartość identyfikatora tego kawału pobraną z bazy danych.
- Gdy strona będzie ładowana, korzystając z instrukcji `if` sprawdzimy, czy pozycji `$_GET['deletejoke']` przypisana jest jakaś konkretna wartość (uczynimy to za pomocą funkcji `isset`). Jeśli tak, użyjemy wartości przypisanej tej pozycji (czyli identyfikatora kawału, który ma zostać usunięty) w zapytaniu `DELETE` języka SQL, które usunie odpowiedni kawał.

Oto kompletny kod rozwiązania. W razie jakichkolwiek wątpliwości warto zasięgnąć informacji na *SitePoint Forums* (*http://www.sitepoint.com/forums/*)!

Listing 4.5. *challenge.php*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Internetowa baza kawałów</title>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-2" />
</head>
<body>
<?php if (isset($_GET['addjoke'])): // Użytkownik chce dodać kawał
?>

<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<label>Wpisz tutaj swój kawał:<br />
<textarea name="tekstkawalu" rows="10" cols="40">
</textarea></label><br />
<input type="submit" value="AKCEPTUJ" />
</form>

<?php else: // Jeśli nie, wyświetlamy domyślną stronę

  // Połącz się z serwerem bazy danych
  $dbcnx = @mysql_connect('localhost', 'root', 'mypasswd');
  if (!$dbcnx) {
    exit('<p>W tej chwili nie można nawiązać ' .
        'połączenia z serwerem bazy danych.</p>' );
  }

  // Wybierz bazę danych z kawałami
  if (!@mysql_select_db('ijdb')) {
    exit('<p><p>Nie można w tej chwili ' .
        'zlokalizować bazy kawałów.</p>');
  }

  // Jeśli użytkownik wprowadził kawał,
  // dodaj żart do bazy danych.
  if (isset($_POST['tekstkawalu'])) {
    $joketext = $_POST['tekstkawalu'];
    $sql = "INSERT INTO kawal SET
        tekstkawalu='$joketext',
```

```
        datakawalu=CURDATE()";
    if (@mysql_query($sql)) {
      echo '<p>Twój kawał został dodany.</p>';
    } else {
      echo '<p>Błąd podczas dodawania kawału: ' .
          mysql_error() . '</p>';
    }
  }

  // Jeśli kawał ma być wyrzucony,
  // usuń go z bazy danych.
  if (isset($_GET['deletejoke'])) {
    $jokeid = $_GET['deletejoke'];
    $sql = "DELETE FROM kawal
        WHERE id=$jokeid";
    if (@mysql_query($sql)) {
      echo '<p>Kawał został usunięty.</p>';
    } else {
      echo '<p>Błąd podczas usuwania kawału: ' .
          mysql_error() . '</p>';
    }
  }

  echo '<p>Oto lista wszystkich kawałów w naszej bazie danych:</p>';

  // Zażądaj tekstu wszystkich kawałów
  $result = @mysql_query('SELECT tekstkawalu FROM kawal');
  if (!$result) {
    exit('<p> Błąd podczas wykonywania zapytania: ' . mysql_error() . '</p>');
  }

  // Wyświetl tekst każdego kawału w osobnym akapicie
  // umieszczając obok łącze "Usuń ten kawał".
  while ($row = mysql_fetch_array($result)) {
    $jokeid = $row['id'];
    $joketext = $row['joketext'];
    echo '<p>' . $joketext .
        ' <a href="' . $_SERVER['PHP_SELF'] .
        '?deletejoke=' . $jokeid . '">' .
        'Usuń ten kawał</a></p>';
  }

  // To łącze po kliknięciu wyświetli stronę
  // z formularzem umożliwiającym dodanie kawału.
  echo '<p><a href="' . $_SERVER['PHP_SELF'] .
      '?addjoke=1">Dodaj kawał!</a></p>';

endif;
?>
</body>
</html>
```