



Wydanie V

Szybki start ►

Larry Ullman

PHP i MySQL

Dynamiczne
strony WWW



Helion 

Tytuł oryginału: PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide (5th Edition)

Tłumaczenie: Piotr Rajca

ISBN: 978-83-283-4466-2

Authorized translation from the English language edition, entitled: VISUAL QUICKPRO GUIDE: PHP AND MYSQL FOR DYNAMIC WEB SITES, Fifth Edition; ISBN 0134301846; by Larry Ullman; published by Pearson Education, Inc, publishing as Peachpit Press. Copyright © 2018 by Larry Ullman.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Polish language edition published by HELION S.A. Copyright © 2018.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz HELION SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

HELION SA

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/phmys5.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/phmys5>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

	Wprowadzenie	11
Rozdział 1.	Wprowadzenie do PHP	27
	Podstawy składni	28
	Przesyłanie danych do przeglądarki internetowej	32
	Wstawianie komentarzy	36
	Co to są zmienne?	40
	Łącuchy	44
	Łączenie łańcuchów	47
	Liczby	49
	Stałe	52
	Apostrof kontra cudzysłów	55
	Proste sposoby usuwania błędów	58
	Podsumowanie i kontynuacja	60
Rozdział 2.	Programowanie w PHP	61
	Tworzenie formularza w języku HTML	62
	Obsługa formularza HTML	67
	Wyrażenia warunkowe i operatory	71
	Weryfikacja danych z formularza	75
	Co to są tablice?	81
	Pętle for i while	96
	Podsumowanie i kontynuacja	99
Rozdział 3.	Tworzenie dynamicznych stron WWW	101
	Stosowanie plików zewnętrznych	102
	Wyświetlanie i obsługa formularzy, po raz wtóry	111
	Tworzenie formularzy z pamięcią	117

	Tworzenie własnych funkcji	121
	Podsumowanie i kontynuacja	138
Rozdział 4.	Wprowadzenie do MySQL-a	139
	Elementy bazy danych i ich nazwy	140
	Wybór typu kolumny	142
	Wybór innych właściwości kolumn	146
	Korzystanie z serwera MySQL	149
	Podsumowanie i kontynuacja	156
Rozdział 5.	Wprowadzenie do SQL-a	157
	Tworzenie baz danych i tabel	158
	Wstawianie rekordów	161
	Wybieranie danych	166
	Wyrażenia warunkowe	168
	Stosowanie LIKE i NOT LIKE	171
	Sortowanie wyników zapytania	173
	Ograniczanie wyników zapytania	175
	Aktualizacja danych	177
	Usuwanie danych	179
	Stosowanie funkcji	181
	Podsumowanie i kontynuacja	192
Rozdział 6.	Projektowanie baz danych	193
	Normalizacja	194
	Tworzenie indeksów	207
	Stosowanie różnych typów tabel	210
	Języki i MySQL	212
	Strefy czasowe a MySQL	217
	Ograniczenia klucza obcego	223
	Podsumowanie i kontynuacja	230
Rozdział 7.	Zaawansowany SQL i MySQL	231
	Złączenia	232
	Grupowanie wyników zapytania	242
	Zaawansowane wybieranie danych	246
	Wyszukiwanie FULLTEXT	250
	Optymalizacja zapytań	258
	Wykonywanie transakcji	262
	Szyfrowanie baz danych	265
	Podsumowanie i kontynuacja	268

Rozdział 8.	Obsługa i usuwanie błędów	269
	Typy błędów i ich usuwanie	270
	Wyświetlanie błędów PHP	276
	Sterowanie raportowaniem błędów PHP	278
	Tworzenie własnych funkcji obsługi błędów	281
	Techniki usuwania błędów z kodu PHP	286
	Techniki usuwania błędów SQL i MySQL	290
	Podsumowanie i kontynuacja	292
Rozdział 9.	PHP i MySQL	293
	Modyfikacja szablonu	294
	Nawiązywanie połączenia z serwerem MySQL	296
	Wykonywanie prostych zapytań	301
	Odczytywanie wyników zapytania	310
	Bezpieczeństwo zapytań	314
	Zliczanie zwróconych rekordów	319
	Aktualizacja rekordów w PHP	322
	Podsumowanie i kontynuacja	330
Rozdział 10.	Popularne techniki programistyczne	331
	Przekazywanie wartości do skryptu	332
	Stosowanie ukrytych pól formularzy	336
	Edycja istniejących rekordów	342
	Stronicowanie wyników zapytań	349
	Wyświetlanie tabel z możliwością sortowania	357
	Podsumowanie i kontynuacja	362
Rozdział 11.	Tworzenie aplikacji internetowych	363
	Wysyłanie poczty elektronicznej	364
	Obsługa przesyłania plików	370
	Skrypty PHP i JavaScript	382
	Nagłówki HTTP	390
	Funkcje daty i czasu	396
	Wykonywanie transakcji	400
	Podsumowanie i kontynuacja	406
Rozdział 12.	Ciasteczka i sesje	407
	Strona logowania	408
	Funkcje logowania	411
	Posługiwanie się ciasteczkami	416
	Sesje	430
	Zwiększanie bezpieczeństwa sesji	438
	Podsumowanie i kontynuacja	442

Rozdział 13.	Zabezpieczenia	443
	Zapobieganie spamowi	444
	Walidacja plików według typu	451
	Walidacja plików na podstawie typu	457
	Zapobieganie atakom XSS	461
	Stosowanie rozszerzenia Filter	464
	Zapobieganie wstrzykiwaniu SQL	468
	Zabezpieczanie haseł w PHP	475
	Podsumowanie i kontynuacja	484
Rozdział 14.	Wyrażenia regularne Perl	485
	Skrypt testujący	486
	Definiowanie prostych wzorców	490
	Stosowanie kwantyfikatorów	493
	Klasy znaków	495
	Wyszukiwanie wszystkich dopasowań	498
	Stosowanie modyfikatorów	502
	Dopasowywanie i zastępowanie wzorców	504
	Podsumowanie i kontynuacja	508
Rozdział 15.	Wprowadzenie do jQuery	509
	Czym jest jQuery?	510
	Dołączanie jQuery do stron WWW	512
	Stosowanie jQuery	515
	Wybieranie elementów stron	518
	Obsługa zdarzeń	521
	Operacje na DOM	525
	Stosowanie Ajaxa	531
	Podsumowanie i kontynuacja	544
Rozdział 16.	Wprowadzenie do programowania obiektowego	545
	Informacje podstawowe i składnia	546
	Korzystanie z MySQL-a	549
	Klasa DateTime	564
	Podsumowanie i kontynuacja	572
Rozdział 17.	Forum dyskusyjne — przykład	573
	Baza danych	574
	Szablony	583
	Strona główna	591
	Strona forum	592
	Strona wątku	597

Wstawianie wiadomości	602
Podsumowanie i kontynuacja	612
Rozdział 18. Rejestracja użytkowników — przykład	613
Tworzenie szablonu	614
Skrypty konfiguracyjne	620
Tworzenie strony głównej	628
Rejestracja	630
Aktywacja konta	640
Logowanie i wylogowywanie się	643
Zarządzanie hasłami	650
Podsumowanie i kontynuacja	660
Dodatek A Instalacja	661
Instalacja w systemie Windows	662
Instalacja w systemie macOS	665
Zarządzanie użytkownikami MySQL	667
Testowanie instalacji	672
Konfigurowanie PHP	675
Konfiguracja serwera Apache	678
Skorowidz	689

9

PHP i MySQL

Teraz, gdy dysponujesz już podstawowymi wiadomościami na temat PHP, SQL-a i MySQL-a, pora użyć tych technologii razem. Silna integracja PHP z MySQL-em jest jednym z głównych powodów ich popularności wśród programistów. Będziesz zaskoczony, jak łatwo można tworzyć aplikacje przy użyciu tych technologii.

W tym rozdziale wykorzystam bazę *sitename* (utworzoną w rozdziale 5., „Wprowadzenie do SQL-a”) i opracuję w PHP interfejs umożliwiający wykonywanie operacji na tabeli **users**. Informacje i przykłady zawarte w tym rozdziale dadzą Ci podstawy do tworzenia wszelkiego rodzaju aplikacji PHP-MySQL, ponieważ pewne uniwersalne reguły są w każdym przypadku takie same.

Zanim przystąpisz do lektury tego rozdziału, powinieneś dobrze opanować materiał przedstawiony w pierwszych ośmiu. Dotyczy to również poznania technik usuwania i obsługi błędów przedstawionych w poprzednim rozdziale. Pamiętaj, że w celu wykonywania przykładów przedstawionych w tym rozdziale będziesz potrzebował działającego serwera internetowego obsługującego PHP, a także dostępu do serwera MySQL.

W tym rozdziale

Modyfikacja szablonu	294
Nawiązywanie połączenia z serwerem MySQL	296
Wykonywanie prostych zapytań	301
Odczytywanie wyników zapytania	310
Bezpieczeństwo zapytań	314
Zliczanie zwróconych rekordów	319
Aktualizacja rekordów w PHP	322
Podsumowanie i kontynuacja	330

Modyfikacja szablonu

Ponieważ wszystkie strony, jakie utworzę w tym i w kolejnym rozdziale, będą częścią tej samej aplikacji internetowej, warto poświęcić trochę czasu na opracowanie dla nich odpowiedniego szablonu. Zamiast tworzyć go od zera, wykorzystam szablon przygotowany w rozdziale 3., „Tworzenie dynamicznych stron WWW”, wprowadzając jedynie drobne modyfikacje w łączach nagłówka.

Aby utworzyć plik nagłówkowy:

1. Otwórz plik *naglowek.html* (listing 3.2) w edytorze tekstów lub IDE.

2. Zmień listę łączy w poniższy sposób (listing 8.1).

```
<ul class="nav navbar-nav">
  <li class="active"><a href=
    "index.php">Strona domowa</a>
  </li>
  <li><a href="rejestracja.php">
    Zarejestruj się</a></li>
  <li><a href=
    "pokaz_uzytkownikow.php">
    Użytkownicy</a></li>
  <li><a href=
    "zmiana_hasla.php">Zmień
    hasło</a></li>
</ul>
```

Wszystkie przykłady zamieszczone w tym rozdziale będą używać stron rejestracji, wyświetlania zarejestrowanych użytkowników i zmiany hasła. Natomiast łącza do stron kalendarza i kalkulatora można usunąć.

Listing 9.1. Plik nagłówka stosowany w szablonie stron i zawierający nowe łącza nawigacyjne

```
1 <!DOCTYPE html>
2 <html lang="pl">
3 <head>
4 <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1">
7 <title><?php echo $page_title; ?></title>
8 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/
  3.3.7/css/bootstrap.min.css" integrity="sha384-
  BVYiSiFeK1dGmJRAKycuHAHRg320mUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
  crossorigin="anonymous">
9 <link href="css/sticky-footer-navbar.css" rel="stylesheet">
10 </head>
11 <body>
12 <nav class="navbar navbar-default navbar-fixed-top">
13   <div class="container">
14     <div class="navbar-header"><a class="navbar-brand" href="#">
      Twoja witryna</a></div>
15     <div id="navbar" class="collapse navbar-collapse">
16       <ul class="nav navbar-nav">
17         <li class="active"><a href="index.php">Strona domowa</a></li>
18         <li><a href="rejestracja.php">Zarejestruj się</a></li>
19         <li><a href="pokaz_uzytkownikow.php">Użytkownicy</a></li>
20         <li><a href="zmiana_hasla.php">Zmień hasło</a></li>
21       </ul>
22     </div>
23   </div>
24 </nav>
25 <div class="container">
26 <!-- Listing 9.1 - naglowek.html -->
```

3. Zapisz plik jako *naglowek.html*.
4. Załaduj nowy plik nagłówka na serwer internetowy do tego samego katalogu *includes*, w którym znajdzie się też plik *stopka.html* (listing 3.3) oraz *layout.css* (dostępny w przykładach do pobrania z <ftp://ftp.helion.pl/przyklady/phmys5.zip>).
5. Przetestuj nowy plik nagłówka, uruchamiając w przeglądarce skrypt *index.php* (patrz **A**).

Wskazówki

- ◆ Aby obejrzeć strukturę tworzonej witryny, zajrzyj do ramki „Zarządzanie dokumentami” w następnym podrozdziale.
- ◆ Nazwy plików pełniących rolę szablonów mogą mieć dowolne rozszerzenia, w tym także *.inc* i *.php*.



A Dynamicznie generowana strona domowa

Nawiązywanie połączenia z serwerem MySQL

Zanim będziesz mógł odwoływać się do MySQL-a, musisz połączyć się z serwerem. Umożliwia to funkcja `mysqli_connect()`:

```
$dbc = mysqli_connect (host, użytkownik,  
↳hasło, nazwa_bazy);
```

To, jak powinny wyglądać pierwsze trzy argumenty przekazywane do funkcji (nazwa hosta, nazwa użytkownika i hasło), zależy od uprawnień, jakie przydzielono poszczególnym użytkownikom na serwerze MySQL. Więcej informacji na ten temat znajdziesz w dodatku A, „Instalacja”. Najczęściej parametrowi *host* nadawać będziesz wartość *localhost*, choć nie zawsze (podanie innej nazwy hosta pozwala połączyć się z bazą obsługiwaną na innym serwerze).

Czwartym z argumentów jest nazwa bazy danych. Określenie wartości tego argumentu stanowi odpowiednik wykonania polecenia *USE nazwa_bazy* za pomocą klienta *mysql*.

Jeżeli uda się nawiązać połączenie, do bazy MySQL będzie można odwoływać się za pośrednictwem zmiennej *\$dbc* (oczywiście jeśli chcesz, możesz jej nadać dowolną inną nazwę). Będziesz ją stosować jako pierwszy argument większości funkcji PHP współpracujących z MySQL-em.

Jeśli pojawi się problem z nawiązaniem połączenia, możesz wywołać funkcję `mysqli_connect_error()`, zwracającą komunikat o błędzie połączenia. Funkcja ta nie wymaga żadnego argumentu, zatem wywołasz ją w poniższy sposób:

```
mysqli_connect_error();
```

Po nawiązaniu połączenia z bazą danych musisz jeszcze ustawić odpowiednie kodowanie; użyj do tego funkcji `mysqli_set_charset()`:

```
mysqli_set_charset($dbc, 'utf8');
```

Wartość określająca kodowanie — czyli drugi argument wywołania funkcji — powinna odpowiadać kodowaniu zastosowanemu w skrypcie PHP oraz uporządkowaniu alfabetycznemu używanemu w bazie danych (patrz rozdział 6., „Projektowanie baz danych”); jeśli tego nie zrobisz, dane mogą być przesyłane z wykorzystaniem domyślnego kodowania, co może wywołać problemy.

Aby rozpocząć korzystanie z MySQL-a w PHP, stworzę specjalny skrypt, który będzie nawiązywać połączenie z serwerem MySQL. Następnie wszystkie pozostałe skrypty wymagające tego połączenia będą dołączać ten plik.

Aby nawiązać połączenie i wybrać bazę danych:

1. W edytorze tekstów lub IDE utwórz nowy dokument PHP, któremu nadasz nazwę *mysqli_connect.php* (listing 9.2).

```
<?php # Listing 9.2 - mysqli_connect.php
```

Plik ten będzie dołączany przez inne skrypty PHP, zatem nie musi zawierać kodu HTML.

2. Umieść nazwę hosta, nazwę użytkownika, hasło i nazwę bazy danych w stałych.

```
DEFINE ('DB_USER', 'uzytkownik');  
DEFINE ('DB_PASSWORD', 'haslo');  
DEFINE ('DB_HOST', 'localhost');  
DEFINE ('DB_NAME', 'sitename');
```

Ze względów bezpieczeństwa wolę umieszczać tego typu dane w stałych (dzięki temu nie można ich zmienić), ale nie jest to wymagane. Ogólnie rzecz biorąc, przypisanie tych wartości do jakichś zmiennych pozwala oddzielić parametry konfiguracyjne od funkcji, które je wykorzystują, ale to również nie jest obowiązkowe.

W trakcie pisania skryptu zmień te cztery wartości na takie, które są odpowiednie dla Twojej bazy danych, lub użyj nazwy użytkownika i hasła, które otrzymałeś dla bazy danych używanej na wynajmowanym serwerze. Możesz również skorzystać ze wskazówek zamieszczonych w dodatku A, stworzyć użytkownika mającego dostęp do bazy *sitename* i wstawić jego parametry do powyższego kodu. Obojętnie, które rozwiązanie zastosujesz, pamiętaj, aby użyć wartości, co do których jesteś absolutnie pewien, że będą działać z używanym przez Ciebie serwerem.

3. Połącz się z MySQL-em.

```
$dbc = @mysqli_connect (DB_HOST, DB_USER,  
↳DB_PASSWORD, DB_NAME) OR  
↳die('Brak połączenia z MySQL: ' .  
↳mysqli_connect_error() );
```

Jeżeli tylko funkcji `mysqli_connect()` uda się nawiązać połączenie z MySQL-em, zwróci ona identyfikator zasobu reprezentujący otworzone połączenie. Zostanie on przypisany do zmiennej `$dbc`, dzięki której, wywołując funkcje dotyczące MySQL-a, będę mógł precyzyjnie określić, o które połączenie mi chodzi.

Wywołanie funkcji zostało poprzedzone operatorem wyłączającym komunikaty o błędach (`@`). Dzięki temu komunikat o ewentualnym błędzie nie zostanie wyświetlony w przeglądarce (co w tym konkretnym przypadku jest wskazane, ponieważ błąd zostanie obsłużony za pomocą klauzuli `OR die()`).

Ciąg dalszy na następnej stronie.

Listing 9.2. Skrypt `mysqli_connect.php` będzie wykorzystywany przez wszystkie pozostałe skrypty aplikacji. Skrypt ten nawiązuje połączenie z serwerem MySQL, wybiera bazę danych i ustawia kodowanie

```
1 <?php # Listing 8.2 - mysqli_connect.php  
2  
3 // Plik zawiera informacje potrzebne do uzyskania dostępu do bazy danych.  
4 // Tworzy połączenie z serwerem MySQL, wybiera bazę danych  
5 // i ustawia kodowanie.  
6  
7 // Tworzymy stałe z informacjami potrzebnymi do nawiązania połączenia z bazą.  
8 define ('DB_USER', 'uzytkownik');  
9 define ('DB_PASSWORD', 'haslo');  
10 define ('DB_HOST', 'localhost');  
11 define ('DB_NAME', 'sitename');  
12  
13 // nawiązujemy połączenie  
14 $dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME)  
    OR die ('Brak połączenia z serwerem MySQL: ' . mysqli_connect_error() );  
15  
16 // określamy kodowanie  
17 mysqli_set_charset($dbc, 'utf8');
```

Jeśli funkcja `mysqli_connect()` nie zdoła zwrócić identyfikatora zasobu, to wykonana zostanie klauzula `OR die()` (ponieważ pierwsza część warunku będzie mieć wartość `FALSE`, zatem konieczne będzie wykonanie drugiej). Jak wyjaśniłem w poprzednim rozdziale, funkcja `die()` kończy działanie skryptu. Parametrem tej funkcji może być łańcuch, który zostanie wyświetlony przez przeglądarkę. W naszym przykładzie łańcuch ten jest kombinacją komunikatu *Brak połączenia z serwerem MySQL* i szczegółowej informacji o błędzie MySQL (patrz **A**). Rozwiązanie takie ułatwia usuwanie błędów pojawiających się w trakcie tworzenia witryny.

4. Ustaw kodowanie:

```
mysqli_set_charset($dbc, 'utf8');
```

Ostatnią operacją wykonywaną w tym skrypcie jest ustawienie sposobu kodowania, który będzie używany w całej przyszłej komunikacji z serwerem MySQL.

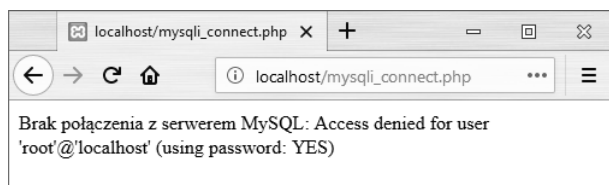
5. Zapisz plik pod nazwą *mysqli_connect.php*.

Ponieważ plik ten zawiera poufne informacje związane z dostępem do bazy danych, zastosowałem w jego przypadku rozszerzenie *.php*. Dzięki temu nawet jeśli użytkownik wykona go w swojej przeglądarce, to nie zobaczy rzeczywistej zawartości strony.

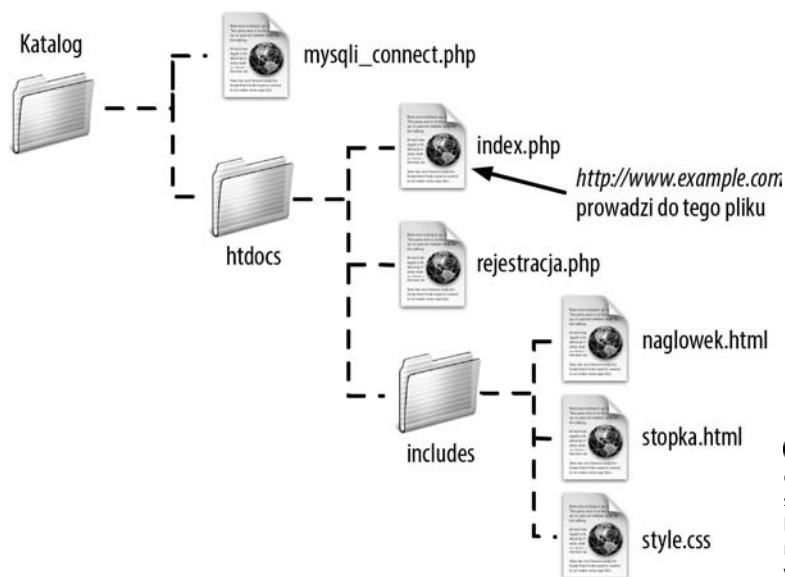
Być może zwróciłeś także uwagę, że w tym skrypcie brakuje zamykającego znacznika PHP, czyli `?`. To dopuszczalne (o ile tylko skrypt kończy się blokiem kodu PHP) i korzystne ze względów, które wyjaśnię w kolejnych rozdziałach.

6. Wgraj plik na serwer, najlepiej do katalogu leżącego poza strukturą katalogów zawierających strony WWW (patrz **B**).

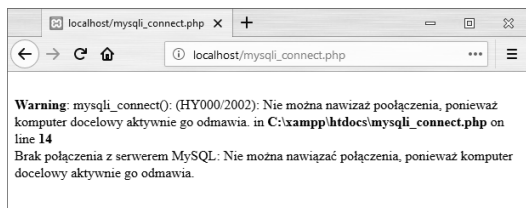
Ponieważ plik ten zawiera poufne informacje, to powinien być przechowywany w bezpiecznym miejscu. Najlepiej, jeśli umieścisz go w jednym z katalogów powyżej struktury katalogów zawierających strony WWW.



A Jeżeli w trakcie wykonywania skryptu pojawi się problem z nawiązaniem połączenia z serwerem MySQL, wyświetlony zostanie odpowiedni komunikat, a wykonanie skryptu zostanie zakończone



B Wizualna reprezentacja dokumentów znajdujących się na serwerze WWW. Plik `mysqli_connect.php` nie jest przechowywany w katalogu głównym (`htdocs`)



C Jeżeli skrypt nawiązujący połączenie z bazą danych działa prawidłowo, powinien zwracać pustą stronę WWW (ponieważ nie generuje on w ogóle kodu HTML)

Dzięki temu nie będzie on dostępny z przeglądarki internetowej. Patrz ramka „Zarządzanie dokumentami”.

7. Umieść tymczasowo kopię tego pliku w katalogu przeznaczonym na strony WWW i przetestuj skrypt w przeglądarce internetowej (patrz **C**).

Aby móc przetestować skrypt, będziesz chciał umieścić jego kopię w takim miejscu na serwerze, do którego można się odwoływać za pośrednictwem przeglądarki internetowej. Jeżeli skrypt działa prawidłowo, zobaczysz pustą stronę WWW (patrz **C**). Jeżeli natomiast ujrzysz komunikat *Access denied...* lub podobny (patrz **A**), będzie to oznaczało, że wprowadzona przez Ciebie nazwa użytkownika, nazwa hosta lub hasło są nieprawidłowe albo że nie masz wystarczających uprawnień, aby połączyć się z daną bazą.

Usuń tymczasową kopię pliku z katalogu publicznego.

Zarządzanie dokumentami

O planowaniu struktury witryny mówiłem już w rozdziale 3., gdy tworzyłem pierwszą aplikację internetową. Teraz, gdy zacznę wykorzystywać skrypt nawiązujący połączenie z bazą, zagadnienie to nabierze jeszcze większego znaczenia.

Gdyby informacje potrzebne do połączenia się z bazą danych (nazwa użytkownika, hasło, nazwa hosta i nazwa bazy danych) wpadły w niepowołane ręce, mogłyby zostać użyte do wykradzenia informacji lub zniszczenia całej zawartości bazy. Dlatego też nie ma żadnej przesady w chronieniu pliku *mysqli_connect.php*.

Najważniejszym zaleceniem odnośnie do bezpieczeństwa jest przechowywanie tego rodzaju plików poza katalogiem przeznaczonym na strony internetowe. Jeżeli na przykład folder *htdocs* widoczny na rysunku **B** jest właśnie takim katalogiem (innymi słowy, prowadzi do niego adres podobny do *www.example.com*), to przechowywanie skryptu *mysqli_connect.php* poza poddrzewem tego katalogu gwarantuje, że nikt nigdy nie dostanie się do tego pliku za pomocą przeglądarki. Co prawda, z założenia kod źródłowy PHP nie jest nigdy przesyłany bezpośrednio do przeglądarki (tylko generowana przez niego strona), ale ostrożności nigdy za wiele. Zatem jeśli nie masz uprawnień, aby umieszczać pliki poza katalogiem *htdocs*, nie oznacza to jeszcze końca świata, ale jest mniej bezpieczne.

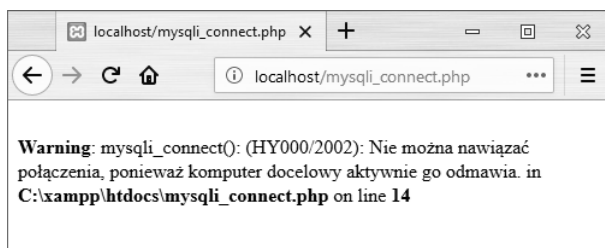
Zalecam także stosowanie dla tego rodzaju skryptów rozszerzenia *.php*. Prawidłowo skonfigurowany serwer po prostu wykona ten kod, a nie będzie wyświetlał go na ekranie. Natomiast przy bezpośrednim odwołaniu do pliku *.inc* jego zawartość zostałaby wyświetlona w przeglądarce.

Wskazówki

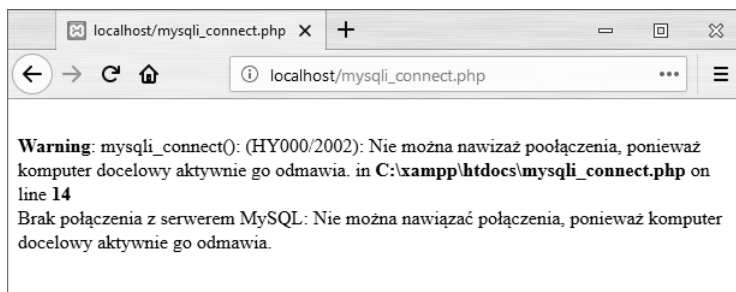
- ◆ W Twoich skryptach PHP powinny zadziałać te same wartości, których używałeś w rozdziale 5. do zalogowania się do monitora mysql.
 - ◆ Jeżeli otrzymasz komunikat o błędzie mówiący, że `mysqli_connect()` jest niezdefiniowaną funkcją, będzie to oznaczało, że obsługa MySQL-a nie została wkompiLOWana w Twoją wersję PHP. Informacje na temat instalacji znajdziesz w dodatku A.
 - ◆ Jeżeli po uruchomieniu skryptu zobaczysz komunikat o błędzie *Can't connect...* (patrz **D**), to prawdopodobnie serwer MySQL nie został uruchomiony.
- ◆ Na rysunku **E** przedstawiona została sytuacja wystąpienia błędu, gdy wywołanie funkcji `mysqli_connect()` nie zostało poprzedzone operatorem `@`.
 - ◆ Jeśli nawiązując połączenie z serwerem MySQL, nie chcesz od razu wybrać bazy danych, możesz pominąć ostatni argument wywołania funkcji `mysqli_connect()`:

```
$dbc = mysqli_connect (host,  
użytkownik, hasło);
```

Następnie, gdy będzie to konieczne, wybierz bazę danych za pomocą wywołania `mysqli_select_db($dbc, nazwa_bazy);`



D Innym powodem, dla którego PHP nie połączy się z serwerem MySQL (oprócz nieprawidłowych: nazwy użytkownika, hasła lub informacji o serwerze), może być to, że ten ostatni nie został w ogóle uruchomiony



E Jeżeli nie zastosujesz operatora `@`, to wyświetlony zostanie zarówno komunikat o błędzie PHP, jak i komunikat przekazany funkcji `die()`

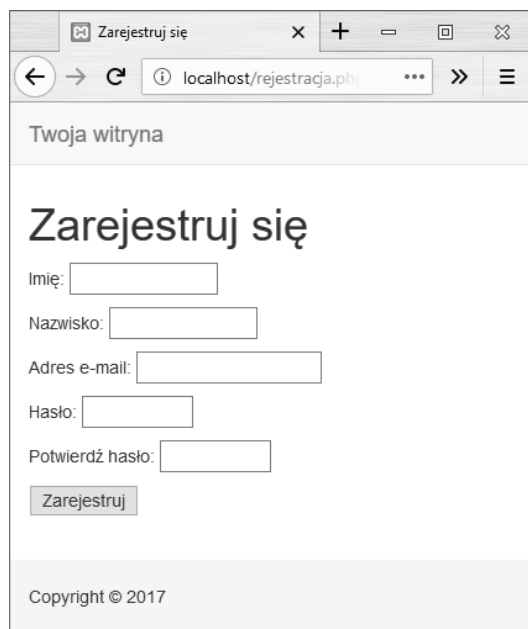
Wykonywanie prostych zapytań

Gdy uda Ci się już połączyć z wybraną bazą danych, możesz rozpocząć wykonywanie zapytań. Mogą to być proste operacje typu *wstawienie*, *aktualizacja* lub *usunięcie rekordu*, ale również złączenia zwracające wiele rekordów naraz. Niezależnie od typu polecenia SQL wykonuje się zawsze przy użyciu funkcji `mysqli_query()`:

```
$wynik = mysqli_query($dbc, $zapytanie);
```

Pierwszym argumentem tej funkcji jest połączenie z serwerem MySQL, a drugim samo zapytanie. W przypadku większych skryptów PHP zazwyczaj zapisują zapytanie w odrębnej zmiennej, o nazwie `$query` lub po prostu `$q`. Wtedy wykonanie zapytania będzie wyglądać jak poniżej:

```
$wynik = mysqli($dbc, $q);
```

The screenshot shows a web browser window with the address bar displaying 'localhost/rejestracja.php'. The page title is 'Zarejestruj się'. The form contains five input fields: 'Imię:', 'Nazwisko:', 'Adres e-mail:', 'Hasło:', and 'Potwierdź hasło:'. Below the fields is a 'Zarejestruj' button. At the bottom of the page, it says 'Copyright © 2017'.

A Formularz rejestracyjny

W przypadku prostych zapytań, które nie zwracają rekordów, takich jak `INSERT`, `UPDATE`, `DELETE` itp., w zmiennej `$wynik` zapisywana jest wartość `TRUE` lub `FALSE`, w zależności od tego, czy wykonanie zapytania przebiegło pomyślnie. Pamiętaj, że „pomyślne wykonanie” oznacza brak błędów podczas realizacji polecenia; nie oznacza to wcale, że zapytanie zwróciło oczekiwane wyniki — to będziesz musiał sprawdzić.

Jeżeli zaś chodzi o bardziej złożone zapytania, zwracające rekordy (z których najważniejsze są zapytania `SELECT`), `$wynik` przechowuje wszystkie zwrócone rekordy lub wartość `FALSE` — jeśli pojawił się jakiś błąd. Zatem, aby sprawdzić, czy zapytanie zakończyło się powodzeniem, możesz zastosować instrukcję warunkową:

```
$wynik = mysqli($dbc, $q);  
if ($wynik) { // Zapytanie wykonane!
```

Jeśli natomiast wykonanie zapytania zakończyło się niepowodzeniem, musiał wystąpić jakiś błąd MySQL. Aby się o nim dowiedzieć, wywołaj funkcję `mysqli_error()`:

```
echo mysqli_error($dbc);
```

Jedynym argumentem tej funkcji jest połączenie z bazą danych.

Po zakończeniu wszystkich operacji na bazie danych można wykonać ostatnią, opcjonalną czynność, to znaczy zamknąć połączenie z serwerem:

```
mysqli_close($dbc);
```

Wywołanie tej funkcji nie jest konieczne, ponieważ PHP automatycznie zamknie połączenie, kończąc wykonywanie skryptu. Jednak dobra praktyka programistyczna nakazuje użyć tej funkcji i jawnie zakończyć połączenie.

Aby zademonstrować omawiany proces, napiszę kolejny skrypt rejestracji użytkownika. Będzie on wyświetlał odpowiedni formularz (patrz **A**), obsługiwać go, przeprowadzać weryfikację wprowadzonych danych i umieszczać je w tabeli `users` bazy `sitename`.

Ciąg dalszy na następnej stronie.

Muszę ostrzec, że przedstawiony tu skrypt jest (w zależności od używanej wersji PHP oraz stosowanych ustawień) niebezpieczny, co wyjaśnię w dalszej części rozdziału.

Aby wykonywać proste zapytania:

1. W edytorze tekstów lub IDE utwórz nowy skrypt PHP (listing 8.3), któremu nadasz nazwę *rejestracja.php*.

```
<?php # Listing 9.3 - rejestracja.php
$page_title = 'Zarejestruj się!';
include ('includes/naglowek.html');
```

Skrypt ten wykorzystuje podstawowe koncepcje omówione w rozdziale 3., takie jak dołączanie plików, wyświetlanie i obsługa formularza przez tę samą stronę oraz zapamiętywanie wprowadzonych danych.

2. Utwórz wyrażenie warunkowe obsługujące dane przekazane z formularza i zainicjalizuj tablicę `$errors`.

```
if ($_SERVER['REQUEST_METHOD'] ==
    'POST') {
    $errors = [];
```

Ten skrypt będzie wyświetlał i obsługiwał formularz HTML. Powyższa instrukcja warunkowa sprawdza rodzaj użytego żądania i na jego podstawie określa sposób obsługi formularza (także to rozwiązanie opisałem dokładniej w rozdziale 3.). Zmiennej `$errors` będę używał do gromadzenia komunikatów o błędach (po jednym dla każdego elementu formularza, który nie został poprawnie wypełniony).

3. Sprawdź imię.

```
if (empty($_POST['first_name'])) {
    $errors[] = 'Zapomniałeś
    ↪podać swoje imię!';
} else {
    $fn = trim($_POST['first_name']);
}
```

Podobnie jak w rozdziale 3., sprawdzam za pomocą funkcji `empty()`, czy pole zostało wypełnione. Jeśli pole nie zostało wypełnione, to do tablicy `$errors` zostanie dodany komunikat o błędzie. W przeciwnym wypadku w zmiennej `$fn` znajdzie się wartość wprowadzona przez użytkownika po usunięciu zbędnych odstępów. Dzięki zastosowaniu tej zmiennej, której nazwa jest oczywiście skrótem od `first_name`, będzie mi znacznie łatwiej sformułować później zapytanie SQL.

4. Sprawdź nazwisko i adres e-mail.

```
if (empty($_POST['last_name'])) {
    $errors[] = 'Zapomniałeś podać
    ↪swoje nazwisko!';
} else {
    $ln = trim($_POST['last_name']);
}
if (empty($_POST['email'])) {
    $errors[] = 'Zapomniałeś podać
    ↪swoją adres e-mail!';
} else {
    $e = trim($_POST['email']);
}
```

Powyższe wiersze kodu są identyczne jak użyte do sprawdzenia pola zawierającego imię. Jeśli kontrola zawartości pól przebiegnie pomyślnie, to w obu wypadkach zostaną utworzone nowe zmienne.

Ciąg dalszy na stronie 305.

Listing 9.3. Skrypt rejestracyjny dodaje do bazy danych nowy rekord, wykonując polecenie INSERT

```
1 <?php # Listing 9.3 - rejestracja.php
2 // Ten skrypt wykonuje polecenie INSERT, by dodać wiersz do tabeli użytkowników.
3
4 $page_title = 'Zarejestruj się';
5 include ('includes/naglowek.html');
6
7 // Sprawdzamy, czy formularz został wysłany
8 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
9
10     $errors = array(); // inicjalizujemy tablicę błędów
11
12     // sprawdzamy imię
13     if (empty($_POST['first_name'])) {
14         $errors[] = 'Zapomniałeś podać swoje imię!';
15     } else {
16         $fn = trim($_POST['first_name']);
17     }
18
19     // sprawdzamy nazwisko
20     if (empty($_POST['last_name'])) {
21         $errors[] = 'Zapomniałeś podać swoje nazwisko!';
22     } else {
23         $ln = trim($_POST['last_name']);
24     }
25
26     // sprawdzamy adres e-mail
27     if (empty($_POST['email'])) {
28         $errors[] = 'Zapomniałeś podać swój adres e-mail!';
29     } else {
30         $e = trim($_POST['email']);
31     }
32
33     // sprawdzamy, czy użytkownik wprowadził hasło i czy w obu polach jest ono takie samo
34     if (!empty($_POST['pass1'])) {
35         if ($_POST['pass1'] != $_POST['pass2']) {
36             $errors[] = 'Za drugim razem wpisałeś inne hasło!';
37         } else {
38             $p = trim($_POST['pass1']);
39         }
40     } else {
41         $errors[] = 'Zapomniałeś wprowadzić hasło!';
42     }
43
44     if (empty($errors)) { // jeśli formularz poprawnie wypełniony...
45
46         // rejestrujemy użytkownika w bazie danych
47     }
```

Ciąg dalszy na następnej stronie.

Listing 9.3. Ciąg dalszy

```

48     require('../mysqli_connect.php'); //nawiązujemy połączenie z bazą danych
49
50     //tworzymy zapytanie
51     $q = "INSERT INTO users (first_name, last_name, email, pass,
        registration_date) VALUES ('$fn', '$ln', '$e', SHA2('$p',512), NOW() )";
52     $r = @mysqli_query ($dbc, $q); //wykonujemy zapytanie
53     if ($r) { //jeśli poprawnie wykonane...
54
55         //wyświetlamy komunikat
56         echo '<h1>Dziękujemy!</h1>';
57         <p>Zostałeś zarejestrowany. W rozdziale 12. będziesz mógł się zalogować!
        </p><p><br /></p>';
58
59     } else { //jeśli zapytanie nie zostało poprawnie wykonane...
60
61         //wyświetlamy komunikat dla użytkownika
62         echo '<h1>Błąd systemu</h1>';
63         <p class="error">Nie zostałeś zarejestrowany z powodu awarii
        naszego systemu. Przepraszamy za kłopot.</p>';
64
65         //wyświetlamy komunikat uruchomieniowy
66         echo '<p>' . mysqli_error($dbc) . '<br><br>Zapytanie: ' . $q .
        '</p>';
67
68     } //koniec instrukcji if ($r).
69
70     mysqli_close($dbc); //zamykamy połączenie z bazą danych
71
72     //dołączamy stopkę i kończymy działanie skryptu
73     include ('includes/stopka.html');
74     exit();
75
76 } else { //wyświetlamy błędy
77
78     echo '<h1>Błąd!</h1>';
79     <p class="error">Wystąpiły następujące błędy:<br>';
80     foreach ($errors as $msg) { //wyświetlamy każdy komunikat o błędzie
81         echo " - $msg<br />\n";
82     }
83     echo '</p><p>Spróbuj jeszcze raz.</p><p><br></p>';
84
85 } //koniec instrukcji if (empty($errors))
86
87 } //koniec głównej instrukcji warunkowej (wysłania formularza)
88 ?>
89 <h1>Zarejestruj się</h1>
90 <form action="rejestracja.php" method="post">
91     <p>Imię: <input type="text" name="first_name" size="15" maxlength="20"
        value="<?php if (isset($_POST['first_name'])) echo $_POST['first_name']; ?>"
        ></p>
92     <p>Nazwisko: <input type="text" name="last_name" size="15" maxlength="40"
        value="<?php if (isset($_POST['last_name'])) echo $_POST['last_name']; ?>"
        ></p>

```

Ciąg dalszy na następnej stronie.

5. Sprawdź hasło.

```
if (empty($_POST['pass1'])) {  
    if ($_POST['pass1'] ==  
        $_POST['pass2']) {  
        $errors[] = 'Za drugim razem  
        ↪ wpisałeś inne hasło!';  
    } else {  
        $p = trim($_POST['pass1']);  
    }  
} else {  
    $errors[] = 'Zapomniałeś  
    ↪ wprowadzić hasło!';  
}
```

Aby zweryfikować hasło, sprawdzam najpierw, czy w odpowiednie pole formularza wprowadzono w ogóle jakąś wartość. Następnie badam, czy wartość `$pass1` jest taka sama jak `$pass2` (a więc, czy użytkownik wprowadził dwukrotnie to samo hasło).

6. Sprawdź, czy można zarejestrować użytkownika

```
if (empty($errors)) {
```

Jeżeli wprowadzone przez użytkownika dane spełniają wszystkie warunki, główny warunek skryptu również jest spełniony i można bezpiecznie kontynuować. Jeśli nie, powinien zostać wyświetlony odpowiedni komunikat o błędzie (patrz punkt 11), a użytkownik powinien otrzymać jeszcze jedną szansę na zarejestrowanie się.

7. Dodaj użytkownika do bazy danych.

```
require_once('..mysqli_connect.php');
```

Pierwsza linijka kodu wstawia do skryptu zawartość pliku `mysqli_connect.php`, tworząc tym samym połączenie z MySQL-em i wybierając bazę danych. Możliwe, że będziesz musiał wprowadzić inną ścieżkę do pliku, zgodną z jego lokalizacją na Twoim serwerze (w obecnej postaci instrukcja zakłada, że plik `mysqli_connect.php` znajduje się w katalogu nadrzędnym względem katalogu zawierającego wykonywany skrypt).

8. Dodaj użytkownika do bazy danych.

```
$q = "INSERT INTO users (first_name,  
    ↪ last_name, email, pass,  
    ↪ registration_date) VALUES ('$fn',  
    ↪ '$ln', '$e', SHA1('$p'), NOW() )";  
$r = @mysqli_query ($dbc, $q);
```

Samo zapytanie SQL jest bardzo podobne do tych, które przedstawiłem w rozdziale 5. Funkcja `SHA2()` wykorzystywana jest do szyfrowania hasła, a `NOW()` ustawia czas rejestracji na bieżącą chwilę. (W rozdziale 13., „Zabezpieczenia”, przedstawię metodę szyfrowania i porównywania haseł na potrzeby rejestracji, której można używać w skryptach PHP).

Ciąg dalszy na następnej stronie.

Listing 9.3. Ciąg dalszy

```
93 <p>Adres e-mail: <input type="text" name="email" size="20" maxlength="80"  
    value="<?php if (isset($_POST['email'])) echo $_POST['email']; ?>" > </p>  
94 <p>Hasło: <input type="password" name="pass1" size="10" maxlength="20"  
    value="<?php if (isset($_POST['pass1'])) echo $_POST['pass1']; ?>" ></p>  
95 <p>Potwierdź hasło: <input type="password" name="pass2" size="10"  
    maxlength="20" value="<?php if (isset($_POST['pass2']))  
    echo $_POST['pass2']; ?>" ></p>  
96 <p><input type="submit" name="submit" value="Zarejestruj" ></p>  
97 </form>  
98 <?php include ('includes/stopka.html'); ?>
```

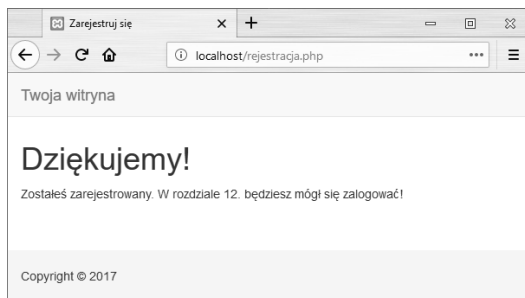
Po przypisaniu zapytania do zmiennej wykonuje je za pośrednictwem funkcji `mysqli_query()`. Przesyła ona polecenia języka SQL na serwer MySQL. Podobnie jak w skrypcie `mysqli_connect.php`, wywołanie funkcji `mysqli_query()` zostało poprzedzone operatorem `@`, aby zapobiec wyświetlaniu komunikatów o błędach w przeglądarce. Jeśli wystąpi błąd, to zostanie on obsłużony w następnym punkcie.

9. Wyświetl informację, czy rejestracja przebiegła pomyślnie.

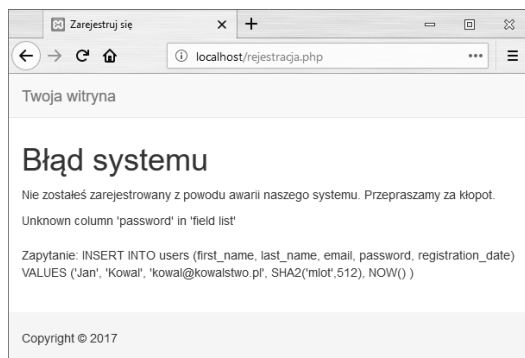
```
if ($r) {  
    echo '<h1>Dziękujemy!</h1>  
    <p>Zostałeś zarejestrowany.  
    W rozdziale 12. będziesz mógł  
    się zalogować!</p><p>  
    <br></p>';  
} else {  
    echo '<h1>Błąd systemu</h1>  
    <p class="error">Nie zostałeś  
    zarejestrowany z powodu awarii  
    naszego systemu. Przepraszamy  
    za kłopot.</p>';  
    echo '<p>' . mysqli_error($dbc)  
    . '<br><br>Zapytanie: '  
    . $q . '</p>';  
}
```

Zmienna `$r`, do której przypisywana jest wartość zwracana przez zapytanie `mysqli_query()`, może zostać użyta w wyrażeniu warunkowym, które sprawdzi, czy polecenie SQL zostało wykonane pomyślnie.

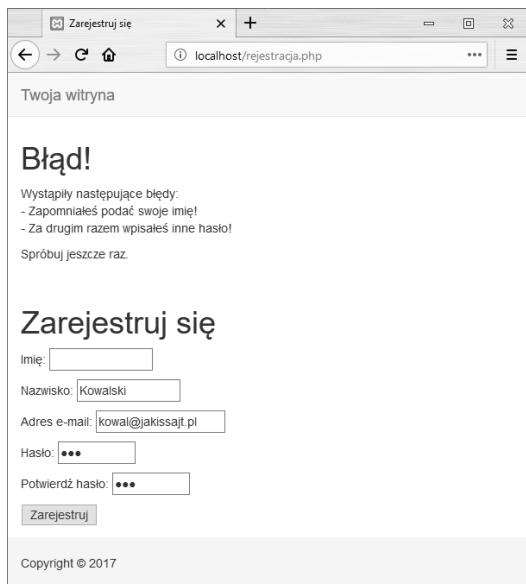
Jeżeli zmienna `$r` ma wartość `TRUE`, wyświetlany jest komunikat z podziękowaniem za rejestrację (patrz **B**). Jeżeli zmienna `$r` ma wartość `FALSE`, wyświetlone zostają komunikaty o błędach. Dla potrzeb procesu usuwania błędów komunikat składa się z wyników zwróconych przez funkcję `mysqli_error()` oraz tekstu wykonanego zapytania (patrz **C**). Takich informacji lepiej jednak nie wyświetlać na witrynie produkcyjnej.



- B** Jeżeli operacja zarejestrowania użytkownika w bazie danych przebiegnie pomyślnie, zostanie wyświetlony ten komunikat



- C** Zostaną wyświetlone wszelkie ewentualne komunikaty o błędach wygenerowane przez zapytanie oraz samo zapytanie



D Każdy błąd wykryty podczas weryfikacji danych jest raportowany, a użytkownik może ponownie wypełnić formularz

10. Zamknij połączenie z bazą danych i dokończ szablon HTML.

```
mysqli_close();
include('includes/stopka.html');
exit();
```

Zamknięcie połączenia nie jest wymagane, ale stanowi przykład dobrej praktyki programistycznej. Następnie dołączona zostaje stopka, a wykonanie skryptu kończy się (przez wywołanie funkcji `exit()`). Gdybym pominął te dwa ostatnie wiersze kodu, formularz rejestracji zostałby ponownie wyświetlony (co nie jest konieczne, jeśli rejestracja zakończyła się pomyślnie).

11. Zamknij główną instrukcję warunkową i wyświetl ewentualne komunikaty o błędach.

```
} else { //wyświetlamy błędy
    echo '<h1>Błąd!</h1>';
    <p class="error">Wystąpiły
        następujące błędy:<br>';
    foreach ($errors as $msg) {
        //wyświetlamy każdy komunikat o błędzie
        echo " - $msg<br>\n";
    }
    echo '</p><p>Spróbuj jeszcze
        raz.</p><p><br></p>';
} //koniec instrukcji if (empty($errors))
} //koniec głównej instrukcji warunkowej
(wysłania formularza)
```

Klauzula `else` zostaje wykonana, jeśli wystąpiły błędy. Wszystkie komunikaty o błędach zostają wyświetlone za pomocą pętli `foreach` (patrz **D**).

Ostatni nawias klamrowy zamyka główne wyrażenie warunkowe związane z wysłaniem danych formularza. Wyrażenie to ma prostą postać `if`, a nie `if-else`, aby formularz zawierał zawsze dane wcześniej wprowadzone (patrz również rozdział 3).

ciąg dalszy na następnej stronie.

12. Zamknij kod PHP i rozpocznij formularz HTML.

```
?>
<h1>Zarejestruj się</h1>
<form action="register.php"
  method="post">
  <p>Imię: <input type="text"
    name="first_name" size="15"
    maxlength="20" value="<?php if
      (isset($_POST['first_name']))
        echo $_POST['first_name']; ?>"
    /></p>
  <p>Nazwisko: <input type="text"
    name="last_name" size="15"
    maxlength="40" value="<?php if
      (isset($_POST['last_name']))
        echo $_POST['last_name']; ?>"
    /></p>
```

Formularz ten jest bardzo prosty, zawiera po jednym polu tekstowym dla każdej kolumny tabeli `users` (z wyjątkiem kolumn `user_id` oraz `registration_date`, które są wypełniane automatycznie). Dane wprowadzone w poszczególnych polach są zapisywane dzięki zastosowaniu kodu w postaci

```
value="<?php if
  (isset($_POST['first_name']))
    echo
      $_POST['first_name']; ?>"
```

Zalecam stosowanie takich samych nazw pól formularza jak odpowiadających im kolumn bazy danych. Dobrze będzie, jeśli pola formularza będą mieć również taką samą długość jak kolumny bazy danych. Oba te rozwiązania pozwalają ograniczyć możliwość pomyłek.

13. Dokończ formularz HTML.

```
<p>Adres e-mail: <input type="text"
  name="email" size="20"
  maxlength="80" value="<?php if
    (isset($_POST['email']))
      echo $_POST['email']; ?>" /></p>
<p>Hasło: <input type="password"
  name="pass1" size="10"
  maxlength="20" value="<?php if
    (isset($_POST['pass1']))
      echo $_POST['pass1']; ?>" ></p>
<p>Potwierdź hasło: <input
  type="password" name="pass2"
  size="10" maxlength="20"
  value="<?php if
    (isset($_POST['pass2']))
      echo $_POST['pass2']; ?>" ></p>
<p><input type="submit"
  name="submit" value="Zarejestruj"
  /></p>
</form>
```

Ta część formularza jest podobna do stworzonej w punkcie 12., lecz dodatkowo został w niej umieszczony przycisk przesyłający formularz.

W przypadku wprowadzania haseł nie muszę się stosować do sugestii dotyczącej ograniczania długości pola za pomocą atrybutu `maxlength` (patrz punkt 12.), ponieważ hasła są szyfrowane za pomocą funkcji `SHA2()`, która zawsze zwraca łańcuch o stałej długości. A ponieważ są dwa pola służące do podania hasła, zatem oba nie mogą jednocześnie mieć tej samej nazwy co kolumna w bazie danych.

14. Zakończ stronę stopką HTML.

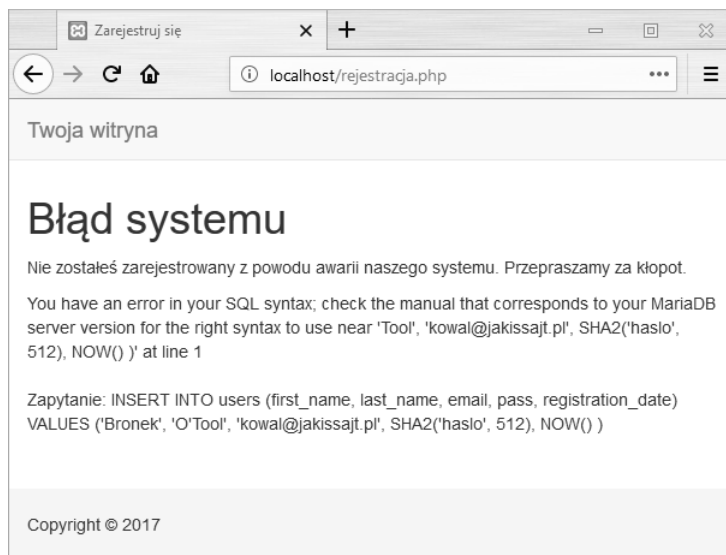
```
<?php
include ('includes/stopka.html');
?>
```

15. Zapisz plik pod nazwą *rejestracja.php*, wgraj go na serwer i przetestuj w przeglądarce internetowej.

Zwróć uwagę, że jeśli w którymś polu formularza wprowadzisz znak apostrofu, to prawdopodobnie spowoduje to błędy podczas wykonywania polecenia SQL (patrz **E**). Sposób rozwiązania tego problemu zostanie pokazany w podrozdziale „Bezpieczeństwo zapytań”.

Wskazówki

- ◆ Po wywołaniu skryptu możesz zawsze sprawdzić, czy zadziałał on jak trzeba, wywołując monitor mysql lub aplikację phpMyAdmin i oglądając zmiany wprowadzone w tabeli.
- ◆ Gdy korzystasz z PHP, nie powinieneś kończyć zapytań SQL średnikiem, tak jak robiłeś to w monitorze mysql. Jest to niegroźny, choć często popełniany błąd w pracy z MySQL-em. Jednak w przypadku innych baz danych, takich jak na przykład Oracle, uniemożliwia on wykonanie zapytania.
- ◆ Funkcja `mysqli_query()` zwraca wartość TRUE, jeżeli udało się wykonać jakieś zapytanie na bazie danych. Nie oznacza to jeszcze, że efekty tej operacji są zgodne z Twoimi oczekiwaniami. W kolejnych skryptach zobaczysz, jak można bardziej precyzyjnie określać efekty wykonania zapytania.
- ◆ Nie musisz koniecznie tworzyć zmiennej `$q` tak jak ja. Możesz wstawić treść zapytania bezpośrednio do funkcji `mysqli_query()`. Jednak w miarę jak Twoje zapytania będą się stawać coraz bardziej złożone, dojdiesz do momentu, w którym użycie zmiennej będzie jedynym rozwiązaniem.
- ◆ Za pomocą funkcji `mysqli_query()` można wykonać praktycznie każde zapytanie, które działa w monitorze mysql.
- ◆ Za pomocą funkcji `mysqli_multi_query()` można wykonywać wiele zapytań za jednym razem. Jednak składnia jest wtedy dość skomplikowana, zwłaszcza gdy zapytania zwracają wyniki.



E Apostrofy wprowadzone w polach formularza (w tym przypadku w nazwisku) spowodują konflikt z apostrofami używanymi do wyodrębniania wartości w zapytaniach SQL

Odczytywanie wyników zapytania

W poprzednim podrozdziale pokazałem Ci, jak można wykonywać proste zapytania na bazie danych MySQL. Przez *proste zapytania* rozumiem tu te, które zaczynają się od słów `INSERT`, `UPDATE`, `DELETE` lub `ALTER`. Mają one wszystkie jedną wspólną cechę — nie zwracają żadnych danych, a jedynie informację, czy udało się je wykonać. Z kolei zapytania `SELECT` generują informacje (na przykład rekordy), które muszą być obsługiwane przez funkcję PHP.

Podstawowym narzędziem, które to umożliwia, jest funkcja `mysqli_fetch_array()`. Pobiera ona zmienną, w której zapisano wynik zapytania (ja nadaję jej nazwę `$r`), i zwraca po jednym wierszu danych naraz w formacie tablicowym. Funkcję tę wykorzystuje się w pętli, uzyskując za każdym razem dostęp do kolejnego rekordu. Podstawowa konstrukcja językowa umożliwiająca odczytanie każdego rekordu zwróconego przez zapytanie wygląda tak:

```
while ($row = mysqli_fetch_array($r)) {  
    // Zrób coś z $row.  
}
```

Praktycznie zawsze będziesz używać pętli `while`, aby pobrać wyniki zapytania `SELECT`.

Funkcja `mysqli_fetch_array()` pobiera też opcjonalny parametr określający rodzaj zwracanej tablicy (asocjacyjna, indeksowana lub dwuformatowa). Tablica asocjacyjna pozwala odwoływać się do wartości przechowywanych w kolumnach po ich nazwach, podczas gdy tablica indeksowana wymaga stosowania oznaczeń liczbowych (gdzie 0 oznacza pierwszą kolumnę). Każdy parametr jest definiowany przez jedną ze stałych

wymienionych w tabeli 9.1, przy czym domyślnie używany jest `MYSQLI_BOTH`. Zazwyczaj korzystam z ustawienia `MYSQLI_NUM`, które jest nieco szybsze (i wymaga mniej pamięci) od pozostałych. Inni programiści wolą `MYSQLI_ASSOC`, ponieważ jest ono bardziej precyzyjne i działa nawet po zmianie struktury tabeli (`$rekord['kolumna']` zamiast `$rekord[3]`).

Gdy korzystasz z funkcji `mysqli_fetch_array()`, możesz zwolnić zasoby, które nie są Ci już potrzebne:

```
mysqli_free_result($r);
```

To wyrażenie zwalnia pamięć zajmowaną przez zmienną `$r`. Podobnie jak `mysqli_close()` jest ono opcjonalne, ponieważ PHP automatycznie zwalnia wszystkie zasoby po dojściu do końca skryptu. Jednak stosowanie go jest dobrą praktyką programistyczną.

Aby pokazać Ci, jak należy obsługiwać wyniki zapytania, utworzę prosty skrypt wyświetlający listę aktualnie zarejestrowanych użytkowników.

Aby odczytać wyniki zapytania:

- I. W edytorze tekstów lub IDE utwórz nowy dokument PHP (listing 9.4).

```
<?php # Skrypt 9.4 - pokaz_uzytkownikow.php  
$page_title = 'Wykaz obecnych  
    uzytkownikow';  
include ('includes/naglowek.html');  
echo '<h1>Zarejestrowani uzytkownicy</h1>'
```

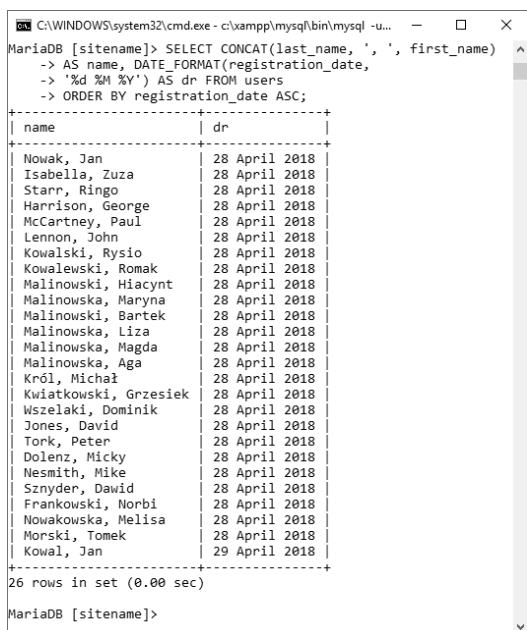
Tabela 9.1. Stałe używane w funkcji `mysqli_fetch_array()`

Stała	Przykład
MYSQL_ASSOC	<code>\$rzad['kolumna']</code>
MYSQL_NUM	<code>\$rzad[0]</code>
MYSQL_BOTH	<code>\$rzad[0]</code> lub <code>\$rzad['kolumna']</code>

2. Połącz się z bazą danych i wykonaj na niej zapytanie.

```
require('../mysqli_connect.php');
$q = "SELECT CONCAT(last_name, ' ',
↳first_name) AS name,
↳DATE_FORMAT(registration_date, '%d %M %Y')
↳AS dr FROM users ORDER BY
↳registration_date ASC";
$r = @mysqli_query ($dbc, $q);
```

Pokazane tu zapytanie zwróci dwie kolumny (patrz **A**): nazwiska użytkowników (w formacie *Nazwisko, Imię*) i daty ich rejestracji (w formacie *Miesiąc DD, RRRR*). Ponieważ obie kolumny zostały sformatowane przy użyciu funkcji MySQL-a, to zwróconym wynikiom nadałem aliasy (*name* i *dr*). Jeśli składnia wydaje Ci się skomplikowana, to wróć do lektury rozdziału 5.



```
MariaDB [sitename]> SELECT CONCAT(last_name, ' ', first_name)
-> AS name, DATE_FORMAT(registration_date,
-> '%d %M %Y') AS dr FROM users
-> ORDER BY registration_date ASC;
```

name	dr
Nowak, Jan	28 April 2018
Isabella, Zuza	28 April 2018
Starr, Ringo	28 April 2018
Harrison, George	28 April 2018
McCartney, Paul	28 April 2018
Lennon, John	28 April 2018
Kowalski, Rysio	28 April 2018
Kowalewski, Romak	28 April 2018
Malinowski, Hiacynt	28 April 2018
Malinowska, Maryna	28 April 2018
Malinowski, Bartek	28 April 2018
Malinowska, Liza	28 April 2018
Malinowska, Magda	28 April 2018
Malinowska, Aga	28 April 2018
Król, Michał	28 April 2018
Kwiatkowski, Grzesiek	28 April 2018
Wszelaki, Dominik	28 April 2018
Jones, David	28 April 2018
Tork, Peter	28 April 2018
Dolenz, Micky	28 April 2018
Nesmith, Mike	28 April 2018
Szynder, Dawid	28 April 2018
Frankowski, Norbi	28 April 2018
Nowakowska, Melisa	28 April 2018
Morski, Tomek	28 April 2018
Kowal, Jan	29 April 2018

26 rows in set (0.00 sec)

MariaDB [sitename]>

A Wyniki zapytania wykonanego za pomocą klienta mysql

3. Wyświetl wyniki zapytania.

```
if ($r) {
    echo '<table width="60%">
    <thead>
    <tr>
        <th align="left">Nazwisko</th>
        <th align="left">Data
        rejestracji</th>
    </tr>
    </thead>
    <tbody>
    ';
```

Jeśli zmienna *\$r* będzie zawierać wartość **TRUE**, będzie to oznaczać, że zapytanie zostało wykonane poprawnie i można wyświetlić wyniki. Wyświetlanie wyników zaczniemy od rozpoczęcia tabeli HTML i wyświetlenia jej nagłówka.

4. Pobierz i wyświetl każdy z pobranych rekordów:

```
while ($row =
    mysqli_fetch_array($r,
    MYSQLI_ASSOC)) {
    echo '<tr><td align="left">'
    . $row['name'] . '</td><td
    align="left">' . $row['dr']
    . '</td></tr>
    ';
```

Następnie w pętli pobierzemy każdy z rekordów przy użyciu funkcji `mysqli_fetch_array()` i wyświetlimy go na stronie. Zwróć uwagę, że wewnątrz pętli `while` odwołuję się do każdej zwróconej wartości, używając właściwego aliasu: `$row['name']` i `$row['dr']`. Nie mógłbym odwołać się do nich jako `$row['first_name']` lub `$row['date_registered']`, ponieważ nie zostały zwrócone pola o takich nazwach (patrz **A**).

Ciąg dalszy na stronie 313.

Listing 9.4. Skrypt pokaz_uzytkownikow.php wykonuje statyczne zapytanie na bazie danych i wyświetla wszystkie zwrócone rekordy

```
1 <?php # Listing 9.4 - pokaz_uzytkownikow.php
2 // Ten listing odczytuje wszystkie rekordy z tabeli users.
3
4 $page_title = 'Zobacz zarejestrowanych użytkowników';
5 include ('includes/naglowek.html');
6
7 //wyświetlamy nagłówek strony
8 echo '<h1>Zarejestrowani użytkownicy</h1>';
9
10 require_once ('../mysqli_connect.php'); //nawiazujemy połączenie z bazą danych
11
12 // tworzymy zapytanie
13 $q = "SELECT CONCAT(last_name, ' ', first_name) AS name, DATE_FORMAT(registration_date,
14     '%d %M %Y') AS dr FROM users ORDER BY registration_date ASC";
15 $r = @mysqli_query ($dbc, $q); // wykonujemy zapytanie
16
17 if ($r) { // jeżeli zapytanie zostało wykonane poprawnie, pokazujemy każdy rekord
18     // nagłówek tabeli
19     echo '<table width="60%">
20     <thead>
21     <tr>
22         <th align="left">Nazwisko</th>
23         <th align="left">Data rejestracji</th>
24     </tr>
25     </thead>
26     <tbody>
27 ';
28
29     // Pobieramy i wyświetlamy wszystkie rekordy
30     while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
31         echo '<tr><td align="left">' . $row['name'] . '</td><td align="left">' .
32             $row['dr'] . '</td></tr>
33     ';
34     }
35
36     echo '</tbody></table>'; //zamykamy tabelę
37
38     mysqli_free_result ($r); //zwalniamy zasoby
39 } else { // jeżeli zapytanie nie zostało wykonane pomyślnie...
40
41     // wyświetlamy komunikat dla użytkownika
42     echo '<p class="error">Nie możemy wyświetlić listy użytkowników
43         ze względu na awarię systemu. Przepraszamy za kłopot.</p>';
44
45     // wyświetlamy uruchomieniowy
46     echo '<p>' . mysqli_error($dbc) . '<br><br>Zapytanie: ' . $q . '</p>';
47 } // koniec instrukcji if ($r).
48
49 mysqli_close($dbc); //zamykamy połączenie z bazą danych
50
51 include ('includes/stopka.html');
52 ?>
```

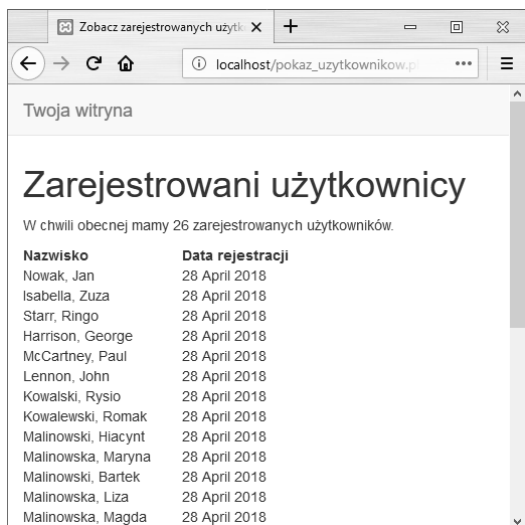
5. Zakończ tabelę HTML i zwolnij zasoby zajęte przez zapytanie.

```
echo '</tbody></table>';  
mysqli_free_result($r);
```

Jeszcze raz powtórzę, że wywoływanie tej funkcji jest opcjonalne, ale warto się na to zdecydować.

6. Dokończ główną instrukcję warunkową.

```
} else {  
    echo '<p class="error">Nie możemy  
        wyświetlić listy użytkowników ze  
        względu na awarię systemu.  
        Przepraszamy za kłopot.</p>';  
    echo '<p>' . mysqli_error($dbc)  
        . '<br><br>Zapytanie: ' . $q  
        . '</p>';  
} // koniec instrukcji if ($r).
```



B Wszystkie rekordy z danymi użytkowników są odczytywane z bazy danych

Podobnie jak w poprzednim przykładzie mamy tutaj dwa komunikaty. Pierwszy jest ogólnym komunikatem, takim jakie wyświetla się użytkownikom działającej witryny. Natomiast drugi jest bardziej szczegółowy, zawiera zarówno informacje o błędzie MySQL jak i tekst zapytania. Ten komunikat przewidziany jest jako pomoc dla programisty w procesie usuwania błędów.

7. Zamknij połączenie z bazą danych i dokończ stronę.

```
mysqli_close($dbc);  
include ('includes/stopka.html');
```

8. Zapisz plik pod nazwą *pokaz_uzytkownikow.php*, wgraj go na serwer i przetestuj w przeglądarce internetowej (patrz **B**).

Wskazówki

- ◆ Funkcja `mysqli_fetch_row()` jest odpowiednikiem `mysqli_fetch_array($r, MYSQLI_NUM)`.
- ◆ Funkcja `mysqli_fetch_assoc()` jest odpowiednikiem `mysqli_fetch_array($r, MYSQLI_ASSOC)`.
- ◆ Gdy odczytujesz rekordy z bazy danych, stosując tablice asocjacyjne, musisz wprowadzać nazwy kolumn lub aliasy dokładnie tak, jak zostały one zdefiniowane w bazie lub zapytaniu. Innymi słowy, wielkie i małe litery w kluczach są rozróżniane.
- ◆ Jeżeli znajdziesz się w sytuacji, w której będziesz musiał wykonać drugie zapytanie w ramach pętli `while`, pamiętaj, aby użyć w nim innych nazw zmiennych (`$r2` i `$row2` zamiast `$r` i `$row`). W przeciwnym wypadku powstaną błędy logiczne.
- ◆ Początkujący programiści PHP często mają problemy z pobieraniem wyników zapytania. Zapamiętaj, że najpierw należy wykonać zapytanie, używając funkcji `mysqli_query()`, a następnie zastosować funkcję `mysqli_fetch_array()` do pobrania pojedynczego rekordu. Jeśli wynik zapytania zawiera wiele rekordów, to użyj pętli `while`.

Bezpieczeństwo zapytań

Zapewnienie bazom danych bezpieczeństwa sprowadza się w PHP do trzech zagadnień:

- Ochrony informacji potrzebnych do uzyskania dostępu do bazy.
- Nieujawniania zbyt wielu informacji o bazie danych.
- Zachowania ostrożności przy wstawianiu danych do bazy, zwłaszcza wprowadzonych przez użytkownika.

Pierwszy z tych celów możesz osiągnąć, umieszczając skrypt odpowiedzialny za połączenie z serwerem MySQL poza katalogiem przechowującym strony WWW (patrz **C** w podrozdziale „Nawiązywanie połączenia z serwerem MySQL” na początku tego rozdziału). W ten sposób nie będzie można podejrzec jego zawartości za pomocą przeglądarki. Zresztą już o tym wspominałem. Drugi cel możemy osiągnąć, nie pokazując użytkownikowi komunikatów o błędach PHP ani treści zapytań (w dotychczasowych przykładach wyświetlałem te informacje w celach uruchomieniowych; nie praktykuj tego nigdy na witrynach produkcyjnych).

Jeżeli zaś chodzi o trzeci cel, to istnieje kilka kroków, które warto podjąć, wychodząc z założenia, że nie należy ufać danym wprowadzonym przez użytkownika. Po pierwsze, sprawdź, czy zostały wprowadzone dane i czy są one odpowiedniego typu (liczba, łańcuch itp.). Po drugie, możesz korzystać z rozszerzenia Filter (przedstawionego

w rozdziale 13.) lub wyrażeń regularnych (przedstawionych w rozdziale 14., „Wyrażenia regularne Perl”), aby upewnić się, że dane wprowadzone przez użytkownika mają taką postać, jak powinny. Po trzecie, możesz rzutować wartości na określone typy, aby zagwarantować, że są liczbami. Czwarty krok polega na stosowaniu funkcji `mysqli_real_escape_string()`, która poprzedza ukośnikami problematyczne znaki:

```
$dane = mysqli_real_escape_string($dbc, $dane);
```

Aby przekonać się, dlaczego stosowanie tej funkcji jest niezbędne, spójrz na rysunek **E** w podrozdziale „Wykonywanie prostych zapytań” wcześniej w tym rozdziale. Umieszczenie znaku apostrofu w polu nazwiska użytkownika sprawiło, że polecenie SQL stało się nieprawidłowe:

```
INSERT INTO users (first_name,
last_name, email, pass,
registration_date) VALUES ('Peter',
'O'Toole', 'petey@example.net',
SHA2('aPass8', 512), NOW() )
```

W tym przypadku to prawidłowe dane wpisane przez użytkownika doprowadziły do wygenerowania nieprawidłowego polecenia SQL. Gdyby jednak skrypt PHP na to pozwalał, to wrogo nastawiony użytkownik mógłby celowo wpisywać problematyczne znaki — na przykład apostrof — aby włamać się do bazy danych lub ją uszkodzić. Dlatego, ze względów bezpieczeństwa, funkcja ta powinna być stosowana dla wszystkich wartości pochodzących z pól tekstowych formularzy. Jej zastosowanie zademonstruję teraz na przykładzie skryptu rejestracji (listing 9.3).

Aby użyć funkcji `mysqli_real_escape_string()`:

1. Jeśli jeszcze tego nie zrobiłeś, to otwórz plik *rejestracja.php* (listing 9.3) w edytorze lub IDE.
2. Przenieś wiersz dołączający plik *mysqli_connect.php* (wiersz 48. na listingu 9.3) zaraz za główną instrukcją warunkową `if` (listing 9.5).

Ponieważ funkcja `mysqli_real_escape_string()` wymaga połączenia z bazą danych, to przed jej wywołaniem musi zostać wykonany skrypt *mysqli_connect.php*.

Zmodyfikuj procedury weryfikujące dane tak, aby wykorzystywały tę funkcję. W tym celu zastąp każde wyrażenie `$var = trim($_POST['var'])` wyrażeniem `$var = mysqli_real_escape_data($dbc, trim($_POST['var']))`.

```
$fn = mysqli_real_escape_string  
($dbc, trim($_POST['first_name']));  
$ln = mysqli_real_escape_string  
($dbc, trim($_POST['last_name']));  
$e = mysqli_real_escape_string  
($dbc, trim($_POST['email']));  
$p = mysqli_real_escape_string ($dbc,  
trim($_POST['pass1']));
```

Zamiast po prostu przypisać wprowadzone wartości do poszczególnych zmiennych (`$fn`, `$ln` itd.), przepuszczam je najpierw przez funkcję `mysqli_real_escape_data()`. Nadal stosuję najpierw funkcję `trim()`, aby pozbyć się niepotrzebnych odstępów.

Ciąg dalszy na stronie 317.

Listing 9.5. Skrypt *rejestracja.php* przepuszcza teraz wszystkie wprowadzone dane przez funkcję `mysqli_real_escape_string()`, aby zapewnić, że będzie można je bezpiecznie zastosować w poleceniu SQL

```
1 <?php # Listing 9.3 - rejestracja.php #2  
2 // Ten skrypt wykonuje polecenie INSERT, by dodać wiersz do tabeli użytkowników.  
3  
4 $page_title = 'Zarejestruj się';  
5 include ('includes/naglowek.html');  
6  
7 // Sprawdzamy, czy formularz został wysłany  
8 if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
9  
10     require('../mysqli_connect.php'); // nawiązujemy połączenie z bazą danych  
11  
12     $errors = array(); // inicjalizujemy tablicę błędów  
13  
14     // sprawdzamy imię  
15     if (empty($_POST['first_name'])) {  
16         $errors[] = 'Zapomniałeś podać swoje imię!';  
17     } else {  
18         $fn = mysqli_real_escape_string($dbc, trim($_POST['first_name']));  
19     }  
20  
21     // sprawdzamy nazwisko  
22     if (empty($_POST['last_name'])) {  
23         $errors[] = 'Zapomniałeś podać swoje nazwisko!';
```

Ciąg dalszy na następnej stronie.

Listing 9.5. Ciąg dalszy

```
24 } else {
25     $ln = mysqli_real_escape_string($dbc, trim($_POST['last_name']));
26 }
27
28 // sprawdzamy adres e-mail
29 if (empty($_POST['email'])) {
30     $errors[] = 'Zapomniałeś podać swój adres e-mail!';
31 } else {
32     $e = mysqli_real_escape_string($dbc, trim($_POST['email']));
33 }
34
35 // sprawdzamy, czy użytkownik wprowadził hasło i czy w obu polach jest ono takie samo
36 if (!empty($_POST['pass1'])) {
37     if ($_POST['pass1'] != $_POST['pass2']) {
38         $errors[] = 'Za drugim razem wpisałeś inne hasło!';
39     } else {
40         $p = mysqli_real_escape_string($dbc, trim($_POST['pass1']));
41     }
42 } else {
43     $errors[] = 'Zapomniałeś wprowadzić hasło!';
44 }
45
46 if (empty($errors)) { // jeśli formularz poprawnie wypełniony...
47
48     // rejestrujemy użytkownika w bazie danych
49
50     // tworzymy zapytanie
51     $q = "INSERT INTO users (first_name, last_name, email, pass,
        registration_date) VALUES ('$fn', '$ln', '$e', SHA1('$p'), NOW() )";
52     $r = @mysqli_query ($dbc, $q); // wykonujemy zapytanie
53     if ($r) { // jeśli poprawnie wykonane...
54
55         // wyświetlamy komunikat
56         echo '<h1>Dziękujemy!</h1>
57         <p>Zostałeś zarejestrowany. W rozdziale 12. będziesz mógł się
58         zalogować!</p><p><br></p>';
59
60     } else { // jeśli zapytanie nie zostało poprawnie wykonane...
61
62         // wyświetlamy komunikat dla użytkownika
63         echo '<h1>Błąd systemu</h1>
64         <p class="error">Nie zostałeś zarejestrowany z powodu awarii
65         naszego systemu. Przepraszamy za kłopot.</p>';
66
67         // wyświetlamy komunikat uruchomieniowy
68         echo '<p>' . mysqli_error($dbc) . '<br><br>Zapytanie: ' . $q . '</p>';
69
70     } // koniec instrukcji if ($r).
```

Ciąg dalszy na następnej stronie.

3. Dodaj drugie wywołanie `mysqli_close()` przed końcem głównej instrukcji warunkowej. `mysqli_close($dbc);`
- Ponieważ pierwszą operacją tego wyrażenia jest nawiązanie połączenia z bazą danych, to ostatnią, konsekwentnie, powinno być jego zamknięcie. Nadal połączenie powinno być również zamykane przed dołączeniem stopki i zakończeniem skryptu (wiersze 73. i 74.).

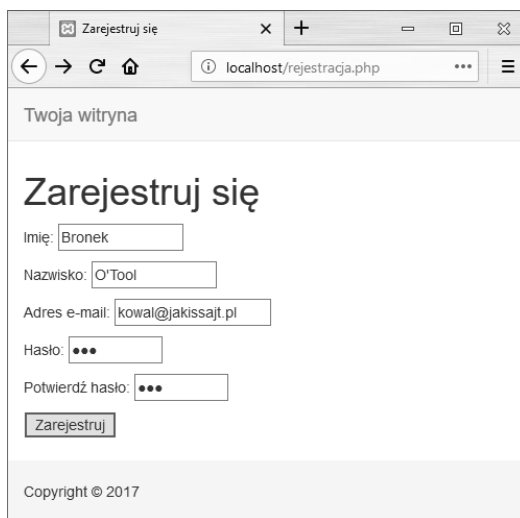
Listing 9.5. Ciąg dalszy

```
70     mysqli_close($dbc); // zamykamy połączenie z bazą danych
71
72     // dołączamy stopkę i kończymy działanie skryptu.
73     include ('includes/stopka.html');
74     exit();
75
76 } else { // wyświetlamy błędy
77
78     echo '<h1>Błąd!</h1>';
79     <p class="error">Wystąpiły następujące błędy:<br>;
80     foreach ($errors as $msg) { // wyświetlamy każdy komunikat o błędzie
81         echo " - $msg<br>\n";
82     }
83     echo '</p><p>Spróbuj jeszcze raz.</p><p><br></p>';
84
85 } // koniec instrukcji if (empty($errors))
86
87 mysqli_close($dbc); // zamykamy połączenie z bazą danych
88
89 } // koniec głównej instrukcji warunkowej (wysłania formularza)
90 ?>
91 <h1>Zarejestruj się</h1>
92 <form action="rejestracja.php" method="post">
93     <p>Imię: <input type="text" name="first_name" size="15" maxlength="20"
94         value="<?php if (isset($_POST['first_name'])) echo $_POST['first_name'];
95         ?>" ></p>
96     <p>Nazwisko: <input type="text" name="last_name" size="15" maxlength="40"
97         value="<?php if (isset($_POST['last_name'])) echo $_POST['last_name'];
98         ?>" ></p>
99     <p>Adres e-mail: <input type="text" name="email" size="20" maxlength="80"
100         value="<?php if (isset($_POST['email'])) echo $_POST['email']; ?>" ></p>
101     <p>Hasło: <input type="password" name="pass1" size="10" maxlength="20"
102         value="<?php if (isset($_POST['pass1'])) echo $_POST['pass1']; ?>" ></p>
103     <p>Potwierdź hasło: <input type="password" name="pass2" size="10"
104         maxlength="20" value="<?php if (isset($_POST['pass2']))
105         echo $_POST['pass2']; ?>" ></p>
106     <p><input type="submit" name="submit" value="Zarejestruj" ></p>
107 </form>
108 <?php include ('includes/stopka.html'); ?>
```

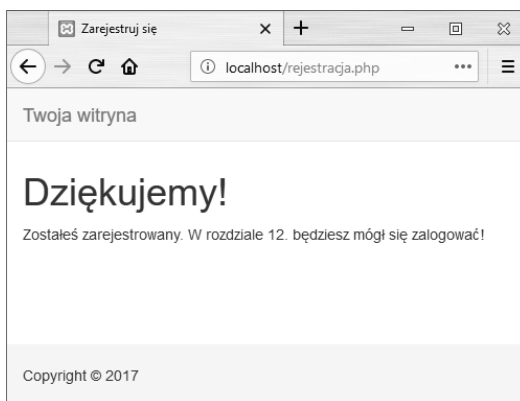
4. Zapisz plik pod nazwą *rejestracja.php*, wgraj go na serwer i przetestuj w przeglądarce (patrz **A** i **B**).

Wskazówki

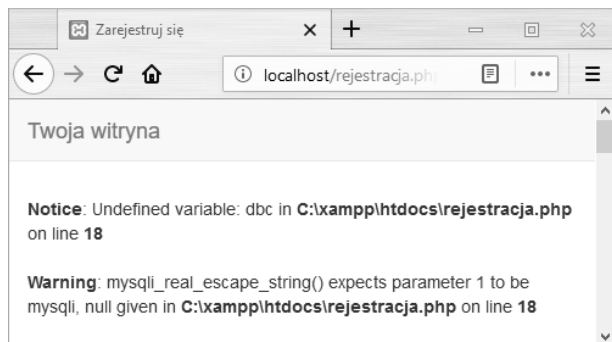
- ◆ Funkcja `mysqli_real_escape_string()` zabezpiecza łańcuchy zgodnie z regułami używanego języka (np. uporządkowaniem alfabetycznym), co daje jej niewątpliwą przewagę nad alternatywnymi rozwiązaniami.
- ◆ Jeżeli widzisz takie komunikaty jak na rysunku **C**, oznacza to, że funkcja `mysqli_real_escape_string()` nie może uzyskać dostępu do bazy danych (ze względu na brak połączenia reprezentowanego przez `$dbc`).
- ◆ Jeśli przejrzysz zawartość bazy (używając klienta mysql lub aplikacji takiej jak phpMyAdmin), to nie zobaczysz problematycznych znaków, które podczas zapisu zostały poprzedzone odwrotnymi ukośnikami. To prawidłowe działanie. Odwrotne ukośniki zabezpieczają polecenie SQL przed problematycznymi znakami, choć same nie są zapisywane w bazie danych.



A Wartości zawierające apostrofy nie spowodują błędu zapytania INSERT, jeśli zastosujesz do nich funkcję `mysqli_real_escape_string()`



B Teraz rejestracja działa również dla wartości zawierających problematyczne znaki i jest bezpieczniejsza



C Ponieważ funkcja `mysqli_real_escape_string()` potrzebuje aktywnego połączenia z bazą danych, niewłaściwe korzystanie z niej (przed dołączeniem skryptu nawiązującego połączenie) może prowadzić do wystąpienia kolejnych błędów

Zliczanie zwróconych rekordów

Następną w kolejności funkcją, którą należałoby omówić, jest `mysqli_num_rows()`. Zwraca ona liczbę rekordów odczytanych przez zapytanie `SELECT`, a pobiera zmienną przechowującą wyniki zapytania.

```
$num = mysqli_num_rows($r);
```

Funkcja ta jest wyjątkowo przydatna w wielu zastosowaniach. Jej użycie jest konieczne, jeśli chcesz wyświetlić wynik zapytania na wielu stronach (przykład znajdziesz w następnym rozdziale). Warto również, abyś zastosował ją, zanim spróbujesz pobrać wyniki zapytania w pętli `while` (ponieważ nie ma sensu pobieranie wyników, jeśli zapytanie nie zwróciło żadnych; wtedy próba pobrania wyników może spowodować błędy). Teraz zmodyfikuję skrypt `pokaz_uzytkownikow.php` w taki sposób, aby wyświetlał on całkowitą liczbę zarejestrowanych użytkowników.

Aby zmodyfikować plik `pokaz_uzytkownikow.php`:

1. Jeśli jeszcze tego nie zrobiłeś, to otwórz plik `pokaz_uzytkownikow.php` w edytorze tekstów lub IDE (patrz listing 9.4).
2. Zaraz przed instrukcją warunkową `if ($r)` wprowadź następującą instrukcję (listing 9.6):

```
$num = mysqli_num_rows($r);
```

Przypisze ona liczbę zwróconych rekordów do zmiennej `$num`.
3. Zmień pierwotne wyrażenie warunkowe `$r` na `if ($num > 0)` {
Dotychczasowe wyrażenie warunkowe bazowało na tym, czy udało się wykonać zapytanie, a nie na tym, czy zwrócono jakiegokolwiek rekordy. Teraz będzie ono bardziej precyzyjne.
4. Zanim utworzysz tabelę HTML, wyświetl liczbę zarejestrowanych użytkowników.

```
echo "<p>W chwili obecnej mamy $num  
↪zarejestrowanych użytkowników.</p>\n";
```

Listing 9.6. Skrypt `pokaz_uzytkownikow.php` będzie teraz wyświetlał całkowitą liczbę zarejestrowanych użytkowników dzięki zastosowaniu funkcji `mysqli_num_rows()`

```
1 <?php # Listing 9.6 - pokaz_uzytkownikow.php #2
2 // Ten listing odczytuje wszystkie rekordy z tabeli users.
3
4 $page_title = 'Zobacz zarejestrowanych użytkowników';
5 include ('includes/naglowek.html');
6
7 // wyświetlamy nagłówek strony
8 echo '<h1>Zarejestrowani użytkownicy</h1>';
9
10 require_once ('../mysqli_connect.php'); // nawiązujemy połączenie z bazą danych
11
12 // tworzymy zapytanie
13 $q = "SELECT CONCAT(last_name, ' ', first_name) AS name,
        DATE_FORMAT(registration_date, '%d %M %Y') AS dr FROM users ORDER BY
        registration_date ASC";
14 $r = @mysqli_query ($dbc, $q); // wykonujemy zapytanie
15
16 // określamy liczbę zwróconych rekordów
17 $num = mysqli_num_rows($r);
18
```

Ciąg dalszy na następnej stronie.

Listing 9.6. Ciąg dalszy

```
19 if ($num > 0) { // jeżeli zapytanie zwróciło jakieś rekordy, to je wyświetlamy
20
21     // wyświetlamy liczbę użytkowników
22     echo "<p>W chwili obecnej mamy $num zarejestrowanych użytkowników.</p>\n";
23
24     // wyświetlamy nagłówek tabeli
25     echo '<table width="60%">
26     <thead>
27     <tr>
28         <th align="left">Nazwisko</th>
29         <th align="left">Data rejestracji</th>
30     </tr>
31     </thead>
32     <tbody>
33 ';
34
35     // pobieramy i wyświetlamy wszystkie rekordy
36     while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
37         echo '<tr><td align="left">' . $row['name'] . '</td><td align="left">'
38             . $row['dr'] . '</td></tr>';
39     }
40
41     echo '</tbody></table>'; // zamykamy tabelę
42
43     mysqli_free_result ($r); // zwalniamy zasoby
44
45 } else { // jeżeli zapytanie nie zostało wykonane pomyślnie...
46
47     echo '<p class="error">Aktualnie nie ma żadnych zarejestrowanych
48         użytkowników.</p>';
49 }
50
51 mysqli_close($dbc); // zamykamy połączenie z bazą danych
52
53 include ('includes/stopka.html');
54 ?>
```

Modyfikacja skryptu rejestracja.php

W skrypcie *rejestracja.php* możesz zastosować funkcję `mysqli_num_rows()`, aby zapobiec wielokrotnemu zarejestrowaniu się użytkownika o tym samym adresie e-mail. Chociaż indeks `UNIQUE` założony na kolumnie przechowującej adres e-mail zapobiegnie takiej sytuacji na poziomie bazy danych, to jednak próba wstawienia kolejnego rekordu o takiej samej wartości tej kolumny spowoduje błąd MySQL-a. Aby uniknąć tego już na poziomie skryptu PHP, wystarczy najpierw wykonać dodatkowe zapytanie `SELECT`, aby sprawdzić, czy adres wprowadzony przez użytkownika nie jest już zarejestrowany w bazie. Zapytanie to będzie mieć całkiem prostą postać:

```
SELECT user_id FROM users WHERE email='$e'
```

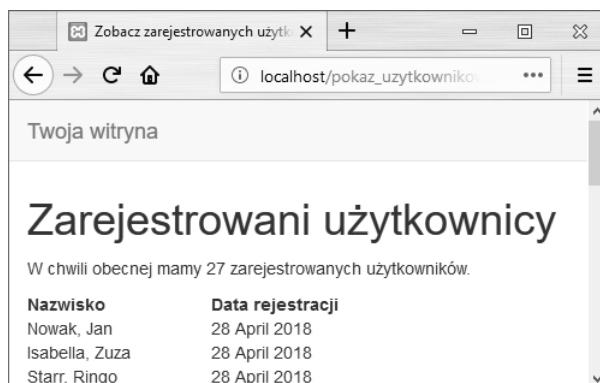
Takie zapytanie powinienś wykonać (używając w tym celu funkcji `mysqli_query()`), a następnie wywołać funkcję `mysqli_num_rows()`. Jeśli wywołanie tej drugiej funkcji zwróci wartość 0, będziesz miał pewność, że adres e-mail nie jest zarejestrowany w bazie i można wykonać polecenie `INSERT`.

5. Zmień klauzulę `else` głównej instrukcji warunkowej

```
echo '<p class="error">Aktualnie nie ma  
żadnych zarejestrowanych użytkowników.</p>';
```

Pierwotne wyrażenie warunkowe opierało się tylko na tym, czy zapytanie zostało prawidłowo wykonane, czy nie. Teraz zakładamy, że zapytanie zostało wystarczająco przetestowane i poprzedni komunikat o błędach nie jest już potrzebny. Obecny komunikat o błędzie informuje jedynie, że nie zostały zwrócone żadne rekordy.

6. Zapisz plik pod nazwą *pokaz_uzytkownikow.php*, wgraj go na serwer i przetestuj w przeglądarce internetowej (patrz **A**).



A Na górze strony jest teraz wyświetlana liczba zarejestrowanych użytkowników

Aktualizacja rekordów w PHP

Ostatnim zagadnieniem, które opiszę w tym rozdziale, jest uaktualnianie rekordów baz danych z poziomu skryptów PHP. Wymaga to użycia zapytania `UPDATE`, którego poprawne wykonanie można potwierdzić, wywołując funkcję `mysqli_affected_rows()`.

Podczas gdy funkcja `mysqli_num_rows()` zwraca liczbę rekordów wygenerowanych przez zapytanie `SELECT`, `mysqli_affected_rows()` zwraca liczbę rekordów, na które miało wpływ wykonanie zapytania `INSERT`, `UPDATE` lub `DELETE`. Używa się jej następująco:

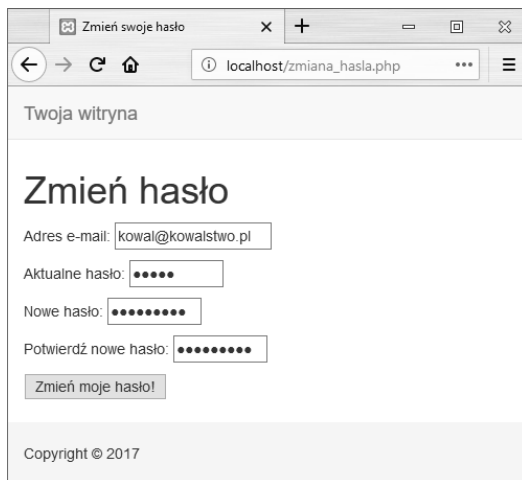
```
$num = mysqli_affected_rows($dbc);
```

Jedynym argumentem pobieranym przez tę funkcję jest identyfikator połączenia z bazą danych (`$dbc`). Nie potrzebuje ona natomiast wyników zwróconych przez ostatnie zapytanie (`$r`).

Utworzę teraz przykładowy skrypt, który pozwala zarejestrowanym użytkownikom zmienić swoje hasło. Zadeemonstruję tym samym dwie ważne techniki:

- Porównywanie nazwy użytkownika i hasła wprowadzonych przez użytkownika z wartościami przechowywanymi w bazie danych (kluczowe zagadnienie w systemie logowania).
- Uaktualnianie rekordów bazy danych przy odwoływaniu się do nich za pomocą klucza głównego.

Podobnie jak w przykładzie rejestracji użytkownika, również i ten skrypt PHP będzie wyświetlać formularz (patrz **A**) i obsługiwać go.

The screenshot shows a web browser window with the title 'Zmień swoje hasło'. The address bar shows 'localhost/zmiana_hasla.php'. The page content includes a heading 'Twoja witryna' followed by 'Zmień hasło'. There are four input fields: 'Adres e-mail:' with the value 'kowal@kowlstwo.pl', 'Aktualne hasło:', 'Nowe hasło:', and 'Potwierdź nowe hasło:'. Each password field contains seven dots. Below the fields is a button labeled 'Zmień moje hasło!'. At the bottom, it says 'Copyright © 2017'.

A Formularz służący do zmiany hasła

Aby uaktualniać rekordy w PHP:

1. W edytorze tekstów lub IDE utwórz nowy skrypt PHP, któremu nadasz nazwę *zmiana_hasla.php* (listing 9.7).

```
<?php # Skrypt 9.7 - zmiana_hasla.php
$page_title = 'Zmień swoje hasło';
include ('includes/naglowek.html');
```

2. Utwórz główną instrukcję warunkową.

```
if ($_SERVER['REQUEST_METHOD'] ==
    'POST') {
```

Ponieważ ta strona jednocześnie wyświetla i obsługuje formularz, zastosuję moje standardowe rozwiązanie do sprawdzania, czy formularz został przesłany, czy nie.

3. Dołącz skrypt nawiązujący połączenie z bazą danych i utwórz tablicę przechowującą komunikaty o błędach.

```
require('../mysqli_connect.php');
$errors = [];
```

Na początku skrypt będzie przypominał skrypt obsługujący formularz rejestracji.

4. Sprawdź wprowadzone dane.

```
if (empty($_POST['email'])) {
    $errors[] = 'Zapomniałeś
wprowadzić adres e-mail!';
} else {
    $e = mysqli_real_escape_string(
        ($dbc, trim($_POST['email'])));
}
if (empty($_POST['pass'])) {
    $errors[] = 'Zapomniałeś
wprowadzić dotychczasowe
hasło!';
} else {
    $p = mysqli_real_escape_string(
        $dbc, trim($_POST['pass']));
}
```

Wszystkie zachodzące tu procesy są dokładnie takie same jak w skrypcie *rejestracja.php*. Formularz (patrz ) będzie miał cztery pola wejściowe: adres e-mail, obecne hasło, nowe hasło i pole pozwalające potwierdzić nowe hasło. Dane, które przejdą kontrolę poprawności, zostaną pozbawione zbędnych odstępów za pomocą funkcji `trim()` i przepuszczone przez funkcję `mysqli_real_escape_string()`, co zapewni ich bezpieczne użycie w zapytaniach SQL.

Ciąg dalszy na stronie 325.

Listing 9.7. Skrypt *zmiana_hasla.php* wykonuje polecenie `UPDATE` i używa funkcji `mysqli_affected_rows()`, aby potwierdzić wprowadzenie zmian

```
1 <?php # Listing 9.7 - zmiana_hasla.php
2 // Ten skrypt pozwala użytkownikowi zmienić hasło.
3
4 $page_title = 'Zmień swoje hasło';
5 include ('includes/naglowek.html');
6
7 // Sprawdzamy, czy formularz został wysłany...
8 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
9
10     require('../mysqli_connect.php'); // nawiązujemy połączenie z bazą danych
11
12     $errors = []; // inicjujemy tablicę błędów
13
14     // sprawdzamy adres e-mail
```

Ciąg dalszy na następnej stronie.

Listing 9.7. Ciąg dalszy

```
15  if (empty($_POST['email'])) {
16      $errors[] = 'Zapomniałeś podać swój adres e-mail!';
17  } else {
18      $e = mysqli_real_escape_string($dbc, trim($_POST['email']));
19  }
20
21  // sprawdzamy, czy użytkownik wprowadził dotychczasowe hasło
22  if (empty($_POST['pass'])) {
23      $errors[] = 'Zapomniałeś wprowadzić dotychczasowe hasło!';
24  } else {
25      $p = mysqli_real_escape_string($dbc, trim($_POST['pass']));
26  }
27
28  // sprawdzamy, czy użytkownik wprowadził w obu polach takie samo hasło
29  if (!empty($_POST['pass1'])) {
30      if ($_POST['pass1'] != $_POST['pass2']) {
31          $errors[] = 'Za drugim razem wpisałeś inne hasło!';
32      } else {
33          $np = mysqli_real_escape_string($dbc, trim($_POST['pass1']));
34      }
35  } else {
36      $errors[] = 'Zapomniałeś wprowadzić nowe hasło!';
37  }
38
39  if (empty($errors)) { // jeśli w formularzu nie było błędów...
40
41      // sprawdzamy, czy została wprowadzona poprawna kombinacja e-mail/hasło
42      $q = "SELECT user_id FROM users WHERE (email='$e' AND pass=SHA2('$p',
43          512) )";
44      $r = @mysqli_query($dbc, $q);
45      $num = @mysqli_num_rows($r);
46      if ($num == 1) { // poprawna
47
48          // pobieramy id użytkownika
49          $row = mysqli_fetch_array($r, MYSQLI_NUM);
50
51          // tworzymy polecenie UPDATE
52          $q = "UPDATE users SET pass=SHA2('$np', 512) WHERE user_id=$row[0]";
53          $r = @mysqli_query($dbc, $q);
54
55          if (mysqli_affected_rows($dbc) == 1) { // jeżeli polecenie zostało wykonane poprawnie...
56
57              // wyświetlamy komunikat
58              echo '<h1>Dziękujemy!</h1>
59              <p>Hasło zostało zmienione. W rozdziale 12. będziesz mógł się
60              zalogować!</p><p><br></p>';
61          } else { // jeżeli polecenie nie zostało wykonane poprawnie...
```

Ciąg dalszy na następnej stronie.

5. Sprawdź nowe hasło.

```
if (empty($_POST['pass1'])) {
    if ($_POST['pass1'] !=
        $_POST['pass2']) {
        $errors[] = 'Za drugim razem
        wpisałeś inne hasło!';
    }
    else {
        $np = mysqli_real_escape_string
        ($dbc, trim($_POST['pass1']));
    }
} else {
    $errors[] = 'Zapomniałeś wprowadzić
    nowe hasło!';
}
```

Również tutaj wszystko przebiega podobnie jak w skrypcie rejestracji z tą różnicą, że nowe hasło zostaje przypisane zmiennej `$np` (ponieważ `$p` reprezentuje bieżące hasło).

6. Jeżeli wszystkie dane przeszły pomyślnie weryfikację, odczytaj identyfikator użytkownika.

```
if (empty($errors)) {
    $q = "SELECT user_id FROM users
    WHERE (email='$e' AND pass=SHA2
    ('$p', 512) )";
    $r = @mysqli_query ($dbc, $q);
    $num = @mysqli_num_rows ($r);
```

Pierwsze zapytanie zwraca tylko zawartość pola `user_id` rekordu pasującego do podanego adresu e-mail i hasła (patrz **B**). Aby porównać wprowadzone hasło z tym przechowywanym w bazie danych, szyfruję je za pomocą funkcji `SHA2()`. Jeżeli użytkownik jest zarejestrowany w bazie i wpisał poprawny adres e-mail i hasło, zostanie zwrócona dokładnie jedna kolumna z jednego rekordu (ponieważ adres e-mail jest unikatowy). Na zakończenie rekord w postaci tablicy (jednoelementowej) zostanie przypisany do zmiennej `$row`.

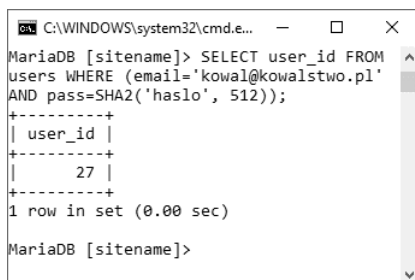
Jeśli ta część skryptu nie działa u Ciebie poprawnie, zastosuj standardowe metody uruchamiania i usuwania błędów: usuń operator `@`, aby sprawdzić, czy wystąpiły jakieś błędy; użyj funkcji `mysqli_error()` do raportowania błędów MySQL-a; wykonaj zapytania, używając innego interfejsu MySQL-a (jak na rysunku **B**).

7. Uaktualnij bazę danych.

```
$q = "UPDATE users
    SET pass=SHA2('$np', 512)
    WHERE user_id=$row[0]";
$re = @mysqli_query ($dbc, $q);
```

To zapytanie zmieni istniejące hasło na nowe w tym rekordzie, w którym zawartość kolumny `user_id` jest równa liczbie odczytanej w poprzednim zapytaniu.

Ciąg dalszy na stronie 327.



```
C:\WINDOWS\system32\cmd.e...  -  □  X
MariaDB [sitename]> SELECT user_id FROM
users WHERE (email='kowl@kowlstwo.pl'
AND pass=SHA2('haslo', 512));
+-----+
| user_id |
+-----+
|      27 |
+-----+
1 row in set (0.00 sec)

MariaDB [sitename]>
```

B Wynik wykonania takiego samego zapytania SELECT jak w skrypcie, ale za pomocą klienta mysql

Listing 9.7. Ciąg dalszy

```

62      // wyświetlamy komunikat dla użytkownika
63      echo '<h1>Błąd systemu</h1>'
64      <p class="error">Hasło nie zostało zmienione z powodu awarii naszego
        systemu. Przepraszamy za kłopot.</p>';
65
66      // wyświetlamy komunikat uruchomieniowy
67      echo '<p>' . mysqli_error($dbc) . '<br><br>Zapytanie: ' . $q
        . '</p>';
68
69    }
70
71    mysqli_close($dbc); // zamykamy połączenie z bazą danych
72
73    // dołączamy stopkę i kończymy skrypt (aby nie wyświetlił formularza)
74    include ('includes/stopka.html');
75    exit();
76
77  } else { // użytkownik wpisał nieprawidłową kombinację e-mail/hasło.
78    echo '<h1>Błąd!</h1>'
79    <p class="error">Podałeś niewłaściwy adres e-mail bądź hasło.</p>';
80  }
81
82  } else { // wyświetlamy komunikaty o błędach
83
84    echo '<h1>Błąd!</h1>'
85    <p class="error">Wystąpiły następujące błędy:<br>';
86    foreach ($errors as $msg) { // wyświetlamy każdy komunikat o błędzie
87      echo " - $msg<br>\n";
88    }
89    echo '</p><p>Spróbuj jeszcze raz.</p><p><br></p>';
90
91  } // koniec instrukcji warunkowej if (empty($errors))
92
93  mysqli_close($dbc); // zamykamy połączenie z bazą danych
94
95  } // koniec głównej instrukcji warunkowej (wysłania formularza)
96  ?>
97  <h1>Zmień hasło</h1>
98  <form action="zmiana_hasla.php" method="post">
99    <p>Adres e-mail: <input type="email" name="email" size="20" maxlength="60"
        value="<?php if (isset($_POST['email'])) echo $_POST['email']; ?>" ></p>
100    <p>Aktualne hasło: <input type="password" name="pass" size="10"
        maxlength="20" value="<?php if (isset($_POST['pass']))
        echo $_POST['pass']; ?>" ></p>
101    <p>Nowe hasło: <input type="password" name="pass1" size="10" maxlength="20"
        value="<?php if (isset($_POST['pass1'])) echo $_POST['pass1']; ?>" ></p>
102    <p>Potwierdź nowe hasło: <input type="password" name="pass2" size="10"
        maxlength="20" value="<?php if (isset($_POST['pass2']))
        echo $_POST['pass2']; ?>" ></p>
103    <p><input type="submit" name="submit" value="Zmień moje hasło!"></p>
104  </form>
105  <?php include ('includes/stopka.html'); ?>

```

8. Sprawdź wyniki zapytania.

```
if (mysqli_affected_rows($dbc) == 1) {  
    echo '<h1>Dziękujemy!</h1>  
<p>Hasło zostało zmienione.  
W rozdziale 12. będziesz mógł się  
zalogować!</p><p><br /></p>';  
} else {  
    echo '<h1>Błąd systemu</h1>  
<p class="error">Hasło nie zostało  
zmienione z powodu awarii naszego  
systemu. Przepraszamy za kłopot.  
</p>';  
    echo '<p>' . mysqli_error($dbc)  
        . '<br /><br />Zapytanie: ' . $q  
        . '</p>';  
}
```

Ta część skryptu również przypomina skrypt *rejestracja.php*. W przypadku gdy funkcja `mysqli_affected_rows()` zwraca liczbę 1, rekord jest uaktualniany i wyświetlany jest komunikat. Jeżeli zwróci ona inną wartość, zostanie wyświetlony komunikat o błędzie.

9. Dołącz stopkę i zakończ działanie skryptu.

```
include ('includes/stopka.html');  
exit();
```

W tym momencie zapytanie `UPDATE` zostało już wykonane i zakończyło się sukcesem lub nie (ze względu na błąd systemu). W obu przypadkach nie trzeba ponownie wyświetlać formularza i dlatego dołączona zostaje stopka, a wywołanie funkcji `exit()` kończy działanie skryptu. Wcześniej jednak, dla porządku, zamykamy połączenie z bazą danych.

10. Dokończ instrukcję warunkową `if ($num == 1)`:

```
} else {  
    echo '<h1>Błąd!</h1>  
<p class="error">Podałeś  
niewłaściwy adres e-mail  
bądź hasło.</p>';  
}
```

Jeśli funkcja `mysqli_num_rows()` zwróci wartość różną od 1, oznacza to, że podany adres e-mail i hasło nie zgadzają się z zapisanymi w bazie. W takiej sytuacji zostanie wyświetlony stosowny komunikat o błędzie oraz formularz, tak by użytkownik mógł wpisać poprawne informacje.

11. Wyświetl ewentualne komunikaty o błędach.

```
} else {  
    echo '<h1>Błąd!</h1>  
<p class="error">Pojawiły się  
następujące błędy:<br />';  
    foreach ($errors as $msg) {  
        echo " - $msg<br />\n";  
    }  
    echo '</p><p>Spróbuj jeszcze  
raz.</p><p><br></p>';  
}
```

Powyższa klauzula `else` zostanie wykonana, gdy tablica `$errors` nie będzie pusta (co oznacza, że dane formularza nie przeszły pomyślnie wszystkich testów). Podobnie jak w przypadku strony rejestracji, błędy zostają wyświetlone.

12. Zamknij połączenie z bazą danych i zakończ kod PHP.

```
mysqli_close($dbc);  
}  
?>
```

Ciąg dalszy na następnej stronie.

13. Wyświetl formularz.

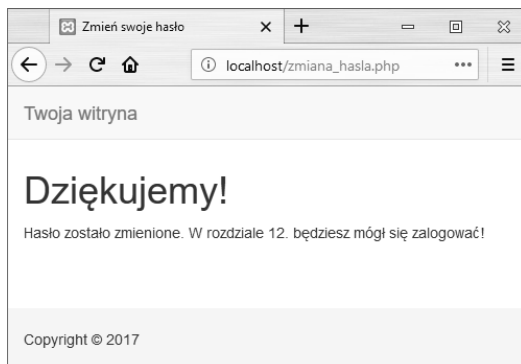
```
<h1>Zmień hasło</h1>
<form action="zmiana_hasla.php"
method="post">
  <p>Adres e-mail: <input
    type="email" name="email"
    size="20" maxlength="60"
    value="<?php if
      (isset($_POST['email']))
      echo $_POST['email']; ?>" >
  </p>
  <p>Aktualne hasło: <input
    type="password" name="pass"
    size="10" maxlength="20"
    value="<?php if
      (isset($_POST['pass']))
      echo $_POST['pass']; ?>" ></p>
  <p>Nowe hasło: <input
    type="password" name="pass1"
    size="10" maxlength="20"
    value="<?php if
      (isset($_POST['pass1']))
      echo $_POST['pass1']; ?>" >
  </p>
  <p>Potwierdź nowe hasło: <input
    type="password" name="pass2"
    size="10" maxlength="20"
    value="<?php if
      (isset($_POST['pass2']))
      echo $_POST['pass2']; ?>" >
  </p>
  <p><input type="submit"
    name="submit" value="Zmień
    moje hasło!"></p>
</form>
```

Formularz zawiera trzy pola służące do podawania hasła — aktualnego, nowego i jego potwierdzenia — a także adresu e-mail. Wprowadzony adres e-mail jest zachowywany w formularzu, co oczywiście nie jest wskazane w przypadku haseł. Wszystkie wartości są przekazywane i wyświetlane po ponownym przesłaniu formularza.

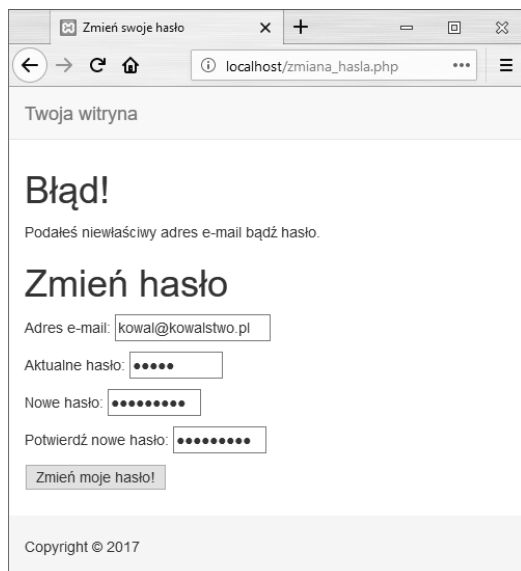
14. Dołącz plik stopki.

```
<?php
include ('includes/stopka.html');
?>
```

15. Zapisz plik pod nazwą `zmiana_hasla.php`, wgraj go na serwer i przetestuj w przeglądarce internetowej (patrz **C** i **D**).



C Hasło w bazie danych zostało zmienione



D Jeżeli wprowadzona nazwa użytkownika i hasło nie zgadzają się z informacjami przechowywanymi w bazie danych, hasło nie zostanie uaktualnione

Wskazówki

- ◆ Jeżeli usuniesz wszystkie rekordy występujące w tabeli za pomocą polecenia `TRUNCATE nazwatabeli`, funkcja `mysqli_affected_rows()` zwróci wartość 0. Nawet wtedy, gdy zapytanie zostało wykonane prawidłowo i rekordy zostały usunięte.
- ◆ Jeżeli wykonujesz zapytanie `UPDATE`, które nie zmienia faktycznej wartości przechowywanej w kolumnie (na przykład zastępujesz bieżące hasło jego wierną kopią), `mysqli_affected_row()` zwróci wartość 0.
- ◆ Użyte tu wyrażenie warunkowe `mysqli_affected_rows()` można by też zastosować (i pewnie wypadaloby) w skrypcie *rejestracja.php*, ponieważ jest ono dokładniejsze niż `if ($r)`.

Podsumowanie i kontynuacja

Podsumowanie

- Której wersji PHP używasz? Której wersji MySQL-a używasz? Czy używane wersje PHP i MySQL-a pozwalają na korzystanie z ulepszonej wersji rozszerzenia MySQL?
- Jaka jest najważniejsza sekwencja czynności służących do debugowania problemów PHP-MySQL (opisana szczegółowo w rozdziale 8., „Obsługa i usuwanie błędów”).
- Jakiej nazwy komputera, nazwy użytkownika i hasła używasz do nawiązywania połączenia z serwerem MySQL?
- Jaki kod PHP pozwala na nawiązanie połączenia z serwerem MySQL, wybranie bazy danych i ustawienie kodowania?
- Jakiego kodowania używasz? Dlaczego skrypt PHP podczas zapisywania tekstu w bazie musi używać tego samego kodowania, które jest używane do interakcji z serwerem MySQL?
- Dlaczego preferowane jest umieszczanie skryptu *mysql_connect.php* poza drzewem katalogu głównego serwera WWW? Co to jest za katalog?
- Dlaczego na witrynach produkcyjnych nie należy wyświetlać komunikatów o błędach MySQL ani o wykonywanych zapytaniach?
- Jaki kod jest niemal zawsze używany do obsługi wyników zwracanych przez polecenia SELECT? Jakiej składni można by użyć, jeśli polecenie SELECT zwróci tylko jeden wiersz wyników?
- Dlaczego ważne jest stosowanie funkcji `mysqli_real_escape_string()`?
- Po jakich poleceniach SQL można używać funkcji `mysqli_num_rows()`?
- Po jakich poleceniach SQL można używać funkcji `mysqli_affected_rows()`?

Kontynuacja

- Jeśli nie pamiętasz, jak działa stosowany przeze mnie system szablonów lub jak należy używać funkcji `include()`, to zajrzyj do rozdziału 3.
- Skorzystaj z informacji podanych w rozdziale 8., aby dodać do przedstawionych tu skryptów własną funkcję obsługi błędów.
- Zmień sposób użycia funkcji `mysqli_num_rows()` w pliku *pokaz_uzytkownikow.php*, tak by była ona wywoływana wyłącznie w przypadku, gdy wykonanie zapytania zwróci wartość TRUE.
- Użyj funkcji `mysqli_num_rows()` w skrypcie *rejestracja.php*, zgodnie z sugestiami podanymi w ramce „Modyfikacja skryptu rejestracja.php”.
- Użyj funkcji `mysqli_affected_rows()` w skrypcie *rejestracja.php*, aby upewnić się, że polecenie INSERT zostało prawidłowo wykonane.
- Jeśli chcesz, to napisz skrypty korzystające z bazy danych *banking*. Prosty projekt mógłbyś zacząć od przygotowania skryptów umożliwiających wyświetlenie listy klientów, wyświetlenie wszystkich kont (użyj złączenia, by wyświetlać także personalia klientów) oraz formularza pozwalającego na modyfikowanie stanu konta.

Skorowidz

1NF, *Patrz:* postać normalna pierwsza
2NF, *Patrz:* postać normalna druga
3NF, *Patrz:* postać normalna trzecia

A

adres

e-mail, 364, 469, 469

weryfikacja, 445

wzorzec, 490

IP, 441, 475

URL, 14, 59

bezwzględny, 411

dołączanie zmiennych, 332, 335

przepisywanie, 686

Ajax, 531

algorytm, MD5, 475, 653

alias, 183, 233, 595

aplikacja

phpMyAdmin, *Patrz:* phpMyAdmin

tworzenie, 363

atak

siłowy, 475

słownikowy, 475, 635

tabela tęczowa, 265

wstrzykiwanie SQL, 468

XSS, 461, 462, 463

B

baza danych, 140

aktywacja konta, 637, 640, 641

bezpieczeństwo, 210

element nazwa, 140, 141

hasło, *Patrz:* hasło baza danych

kolumna, 140

atomowa, 197

AUTO_INCREMENT, 146, 147

dodawanie, 250

indeksowanie, *Patrz:* indeks

tekstowa, 148

typ, 142, 143, 144, 145, 147, 250, 265,

Patrz też: typ

usuwanie, 250

modelowanie, *Patrz:* baza danych

projektowanie

normalizacja, 146, 194, 197, 198, 199, 200, 201,

203, 204*Patrz też:* postać normalna

pomijanie, 204

NoSQL, 18

obiekтова, 18

optymalizacja, 258

pole, 142

postać normalna, *Patrz:* postać normalna

projektowanie, 193, 197, 204, 205, 574

rekord

aktualizacja, 177, 322, 342, 343, 348, 477

blokowanie, 210

dopasowany, 233, 236

niedopasowany, 236

usuwanie, 179, 180, 223, 224, 225, 229

wstawianie, 161, 162, 265

relacyjna, 18, 194, 197, 232

schemat, *Patrz:* baza danych struktura

struktura, 194

szyfrowanie, *Patrz:* szyfrowanie, funkcja

szyfrująca, funkcja deszyfrująca

tabela, 140

InnoDB, 210, 211, 215, 223, 262

MEMORY, 211

MyISAM, 210, 211

sortowanie, 357

tworzenie, 158

typ, 160, 210, 211

tworzenie, 140, 158, 159, 207, 210, 576, 577

wybijanie danych, *Patrz:* zapytanie

zależności, *Patrz:* zależność

bezpieczeństwo, 76, 104, 296, 361, 443, 445, 475, 476
 atak, *Patrz:* atak
 dostęp do katalogów, 371
 e-mail, *Patrz:* e-mail, spam
 przesyłanie plików, 371
 sesja, 430, 438, 439, 440, 441
 szyfrowanie, *Patrz:* szyfrowanie, funkcja szyfrująca, funkcja deszyfrująca
 weryfikacja danych, 451
 zapytań, 314
 biblioteka
 jQuery, *Patrz:* jQuery
 SPL, *Patrz:* SPL
 błąd
 Access denied for user, 291
 headers already sent, 420
 HTML-a, 270
 usuwanie, 275
 komunikat, *Patrz:* komunikat o błędach
 logiczny, 270, 271
 MySQL, 270, 271, 273
 brak dostępu do bazy danych, 271, 291
 dziennik, 291
 kod, 291
 usuwanie, 290, 291
 parsowania, 286
 PHP, 270, 272, 286
 usuwanie, 286, 287, 288, 289
 własna funkcja obsługi, 281, 282, 283, 284, 285
 wyświetlanie, 276, 277, 278, 279, 280, 281
 składni, 270, 286, 287, 456
 SQL, 270, 271
 usuwanie, 290, 291
 usuwanie, *Patrz też:* debugowanie
 użytkowania, 274
 wykonania, 270
 związany
 z ciasteczkami, *Patrz:* ciasteczko błąd
 z sesją, *Patrz:* sesja błąd

C

CAPTCHA, 450
 CDN, 513
 ciasteczko, 407, 416, 420, 422, 430, 476, 537
 błąd, 416, 420
 Internet Explorer, 429
 dostęp, 420, 421, 422

parametr, *Patrz:* funkcja setcookie parametr
 PHPSESSID, 430, 433
 testowanie, 417
 tworzenie, 416, 417, 418
 usuwanie, 425, 426, 429
 wygasanie, 424
 wysłanie, 418, 419, 420
 Content Delivery Network, *Patrz:* CDN
 cookies, *Patrz:* ciasteczko
 CSS, 23, 63
 framework Bootstrap, *Patrz:* framework Bootstrap
 klasa, *Patrz:* klasa CSS
 czas
 GMT, 217
 strefa czasowa, 217, 218, 565, 592
 UTC, 217, 218, 219, 221, 222
 Zulu, 222

D

dane
 baza, *Patrz:* baza danych
 binarne, 145
 poufne, 110, 296, 298, 299, 341, 476
 przesyłanie do przeglądarki internetowej, *Patrz:* przeglądarka przesyłanie danych
 sanityzacja, 464
 szyfrowanie, 163
 walidacja, *Patrz:* dane weryfikacja
 weryfikacja, 451
 HTML5, 452
 rozszerzenie Filter, 464
 data, 143, 187, 188, 190, 191, 564
 DBMS, 18
 debugowanie, 31, 57, 59, 272, 273, 274
 JavaScript, 511
 deszyfrowanie, 265, 266, *Patrz też:* funkcja deszyfrująca
 dokument
 HTML, 518
 model obiektowy, *Patrz:* DOM
 dokumentacja PHP, 48
 DOM, 513, 515, 525
 element, 525, 526, 530
 dziennik, 285, 291

E

edytor tekstów, 29, 272, 489
e-mail, 364
 aktywacja konta, 637, 640, 641
 ciało, *Patrz:* e-mail treść
 hasło, *Patrz:* hasło poczty elektronicznej
 nagłówek, 365, 366, 444, 445
 odzyskiwanie hasła, 650, 654
 treść, 365, 444
 tworzenie, 364
 wysyłanie, 365
encja, 197, 461

F

Ferrare Anthony, 477
Filter, 464, 465, 466, 467
 FILTER_SANITIZE_, 464, 465
 FILTER_VALIDATE_, 464
foreign key constraints, *Patrz:* klucz obcy
 ograniczenia
format JSON, 543
formularz, 62, 91, 365, 366, 444
 HTML, 518, 519, 520, 533, 534, 537
 obsługa, 67, 68, 111, 112, 522, 606, 608, 609
 pole ukryte, 336, 341
 tworzenie, 63, 115, 602, 603
 weryfikacja danych, 75, 76, 114
 z pamięcią, 117, 118, 119, 120
forum, 573, 574
 administrowanie, 611
 baza danych, 574, 575
 tworzenie, 576, 577
 formularz
 obsługa, 606, 607
 tworzenie, 602, 603
 strona
 forum, 591
 główna, 590
 wątku, 597, 598
 wiadomość, 602
framework
 Bootstrap, 104, 116, 584
 JavaScript, 511
 jQuery, *Patrz:* jQuery
funkcja
 ABS, 185
 addClass, 525
 ADDDATE, 189

addslashes, 465
ADDDATE, 189
AES_DECRYPT, 265, 267
AES_ENCRYPT, 265, 267
ajax, 537
anonimowa, 516, 522
append, 530
argument, 123, 124
 wartość domyślna, 127, 130
array_map, 449, 450
arsort, 92
asort, 92
attr, 530
AVG, 242, 243
CASE, 246
CEILING, 185
check_login, 483
checkdate, 396
COALESCE, 245, 249
CONCAT, 182, 184
CONCAT_WS, 182, 184
CONVERT_TZ, 218, 222, 593
count, 88
COUNT, 242, 243, 244, 245, 595
create_window, 384, 390
CURDATE, 187, 188
CURTIME, 187, 188
DATE, 187, 188
date, 396, 398
date_default_timezone_set, 396, 398
DATE_FORMAT, 190, 200
DATEDIFF, 189
daty i czasu, 396, 397, 398, 564
DAYNAME, 187, 188
DAYOFMONTH, 187
debug_print_backtrace, 284
define, 52
deszyfrująca, 265
die, 289, 625
dokumentacja, 48
echo, 32, 34, 35, 36
empty, 75
empty, 77
error_log, 284, 285
error_reporting, 279
et_error_handler, 282
exit, 289, 411
explode, 92, 568
faden, 525
fadeOut, 525

funkcja

filesize, 394, 398
filter_has_var, 467
filter_input_array, 467
filter_var, 464
finfo_close, 457, 459
finfo_file, 457, 459
finfo_open, 457
FLOOR, 185
FORMAT, 185
getdate, 396
getimagesize, 386
GREATEST, 245
GROUP_CONCAT, 242, 245
grupująca, 242, 243, 244, 245
header, 391, 392, 395, 411, 415, 584, 615
headers_sent, 395
hide, 525
HOUR, 187
html, 525
htmlentities, 461, 462, 463, 489, 609
htmlspecialchars, 461, 463, 465, 609
IFNULL, 249
implode, 92
include, 102, 103, 110, 136
include_once, 102, 103
ini_set, 276, 277, 437, 438
intdiv, 51
is_array, 88, 451
is_bool, 451
is_float, 451
is_int, 451
is_null, 451
is_numeric, 80, 451
is_resource, 451
is_scalar, 451
is_string, 451
isset, 71, 75, 350, 552
krsort, 92
ksort, 92
LEFT, 182
LENGTH, 182, 184
list, 135
logowania, 411, 412, 415
LOWER, 182
mail, 364, 365, 369, 444, 676
MAX, 242, 595
md5, 653
MIN, 242, 595
MINUTE, 187

mktime, 396
MOD, 185, 186
MONTH, 187
MONTHNAME, 187
my_error_handler, 282, 623, 624
mysqli_affected_rows, 322, 323, 329
mysqli_close, 301, 307, 317
mysqli_connect, 296, 297, 298, 625
mysqli_connect_error, 296
mysqli_error, 301, 306
mysqli_fetch_array, 310, 311, 557
mysqli_fetch_assoc, 313
mysqli_fetch_row, 313
mysqli_free_result, 310
mysqli_multi_query, 309
mysqli_num_rows, 319, 321, 322
mysqli_prepare, 469
mysqli_query, 301, 306, 309
mysqli_real_escape_string, 314, 315, 318, 468, 481, 609, 610, 636
mysqli_set_charset, 296
mysqli_stmt_bind_param, 469
natsort, 95
nazwa, 121
NOT REGEXP, 184
NOW, 163
NOW, 187
number_format, 49, 114
numeryczna, 185, 186
ob_clean, 619
ob_end_clean, 615
ob_end_flush, 615, 619
ob_start, 615
obsługi błędów, 281, 282, 283, 284, 285
operująca na danych, 187, 188
parametr, 123
password_hash, 477, 650
password_verify, 477, 483
phpinfo, 273, 372
POW, 185
preg_match, 486, 488, 498, 504
preg_match_all, 498, 504
preg_replace, 504, 505, 506, 507
preg_split, 501
prepend, 530
print, 32, 34, 35, 36
print_r, 284
RAND, 185
rand, 636, 656
random_bytes, 653

range, 88
ready, 515, 517
REGEXP, 184
remove, 530
removeClass, 525
REPLACE, 182
require, 102, 103, 110
require_once, 102, 103
RIGHT, 182
round, 49
ROUND, 185
rsort, 92
rtrim, 412
SECOND, 187
session_id, 435
session_name, 438
session_regenerate_id, 441
session_set_cookie_params, 438
session_start, 430, 431, 433, 435, 437, 438, 615, 619
session_status, 435
setcookie, 417, 420, 425, 438, 615
 parametr, 422, 423, 424, 425, 429
SHA2, 163, 265, 266, 267, 308, 477
show, 525
shuffle, 95
sleep, 475
sort, 92
SQRT, 185
str_replace, 448
strip_tags, 461, 463, 465, 609
stripos, 446
strlen, 48, 123
strstr, 492
strtolower, 48, 392
strtoupper, 48
SUBDATE, 189
substr, 392
SUBSTRING, 182
SUBTIME, 189
SUM, 242, 243
szyfrująca, 265, 266
tekstowa, 182
text, 525
time, 422
TIME_FORMAT, 190, 200
TODAY, 53
toggleClass, 525
trigger_error, 278, 280, 624

TRIM, 182
trim, 315, 323, 344, 412, 488, 489
tworzenie, 121, 122, 123, 124
ucfirst, 48
ucwords, 48
UNHEX, 266
uniqid, 636, 653
UNIX_TIMESTAMP, 187
unset, 435, 550, 556
UPPER, 182
usort, 95
UTC_DATE, 217
UTC_TIME, 217
UTC_TIMESTAMP, 187, 217, 221, 610
val, 523, 525
w PHP 7, 137
wordwrap, 365
YEAR, 187
zwracanie wartości, 131, 137

G

garbage collection, *Patrz:* sesja procedura oczyszczająca

H

hasło, 141, 475, 635
 baza danych, 150, 151, 154, 265, 267, 291, 296, 299, 305
 zmiana, 323, 348, 329, 348
 odzyskiwanie, 650, 651, 654
 poczty elektronicznej, 194, 208, 410, 414, 469
 szyfrowanie, 162, 170, 305, 308
 zarządzanie, 650, 655
 zmiana, 655, 659
host wirtualny, 679, 680, 681
HTML, 23, 461
 znacznik, *Patrz:* znacznik HTML
HTML5
 pole tekstowe, 115
 weryfikacja
 danych, 452
 formularza, 100
HTTP, 391
 nagłówek, *Patrz:* nagłówek HTTP
 protokół, *Patrz:* protokół HTTP

I

- IDE, 21
- indeks, 146, 207, 229
 - dodawanie, 250, 251
 - FULLTEXT, 207, 208, 210, 211, 250, 251, 252, 253, 254
 - tryb Boolean, 255
 - WITH QUERY EXPANSION, 257
 - INDEX, 207, 208
 - PRIMARY KEY, 207, 208
 - tworzenie, 207, 208
 - UNIQUE, 207, 208, 209
 - usuwanie, 250
 - wielokolumnowy, 207
- instrukcja, *Patrz też:* funkcja
 - break, 74
 - return, 131, 132, 135
- warunkowa
 - else, 71, 115
 - elseif, 71
 - formatowanie, 74
 - if, 71
 - stosowanie, 72
 - switch, 74

J

- JavaScript, 382, 510
 - framework, *Patrz:* framework
 - obiekt
 - ogólny, 537
 - XMLHttpRequest, *Patrz:* XHR
- język
 - obiektyowy, 383, 511, 545
 - funkcja, 517
 - składnia, 546, 547
 - skryptowy, 14, 27, 28, 407
 - SQL, *Patrz:* SQL
- jQuery, 510, 516
 - dołączanie, 512, 513
 - kod, 515
 - selektor, 520
 - wersja, 512, 513
 - wtyczka, *Patrz:* wtyczka
- jQuery Mobile, 510
- jQuery UI, 510

K

- catalog, 110
 - htdocs, 42, 298, 299
 - includes, 104
 - templates, 104
 - uploads, 371
- klasa, 546
 - atrybut, *Patrz:* klasa właściwość
 - CSS, 76
 - dodawanie, 525
 - error, 519, 526
 - errorMessage, 519, 526, 538
 - usuwanie, 525
 - DateInterval, 566, 568
 - DateTime, 547, 548, 564, 565, 567, 571
 - instancja, 549, *Patrz też:* obiekt tworzenie
 - konstruktor, 547
 - metoda, 546, 548, *Patrz też:* metoda
 - MySQLi, 547, 548, 549, 551
 - MySQLi_Result, 549
 - MySQLi_Stmt, 549, 560
 - nazwa, 546
 - składowa, 546
 - właściwość, 546
- klauzula
 - ADD COLUMN, 250
 - ADD INDEX, 250
 - CHANGE COLUMN, 250
 - DROP COLUMN, 250
 - DROP INDEX, 250
 - GROUP BY, 243, 245, 595
 - JOIN, 323
 - LIMIT, 175, 243, 349
 - ORDER BY, 173, 174, 175, 184, 207, 208, 232, 243
 - RENAME AS, 250
 - WHERE, *Patrz:* wyrażenie warunkowe WHERE
- klient
 - FTP, 21
 - MySQL, *Patrz:* MySQL klient
- klucz, 81, 146, 196
 - główny, 146, 160, 195, 200, 208, 215, 223, 224
 - łańcuchowy, 81, 82, 84
 - numeryczny, 81, 82, 84
 - obcy, 146, 160, 195, 223, 224
 - ograniczenia, 223, 224, 225, 229, 611
 - porównywanie, 224

- kod
 - PHP, 29, *Patrz też:* skrypt
 - formatowanie, 74
 - szablon, *Patrz:* szablon
- kodowanie, 28, 31, 212, 213, 214, 296, 549, 576
 - UTF-8, 28, 213, 219
- komentarz
 - aktualizowanie, 39
 - HTML, 36
 - JavaScript, 384
 - PHP, 36, 37
 - #, 36, 37
 - /*, 36, 37
 - //, 36, 37
 - zagnieżdżanie, 39
- komunikat o błędach, 59, 270, 313
 - dziennik, 285
 - jQuery, 526, 528
 - numeracja wierszy, 287
 - PHP, 276, 277, 278, 279, 281, 286
 - ukrywanie, 278, 279, 280, 314

L

- Lerdorf Rasmus, 13
- liczba, 49, 451
 - całkowita, 40, 51, 143
 - zmiennoprzecinkowa, 40, 51, 143
- lista rozwijana, 65
- logowanie, 408, 409, 410, 435, 469, 643, 644
 - atak siłowy, *Patrz:* atak siłowy
 - wylogowanie, *Patrz:* wylogowanie
 - zmiana, 481

Ł

- łańcuch, 44, 182, 183, 451, 461, 486, 492
 - długość, 48
 - konwersja na tablicę, 92
 - łączenie, 47, 48
 - pusty, 71
 - tworzenie, 45

M

- mechanizm składowania danych, 210, *Patrz też:*
 - baza danych tabela typ
- metaznak
 - \$, 490, 493, 502
 - (, 490
 -), 490

- *, 493
- ., 490, 495
- ?, 499
- ?, 493
- [, 490, 495
- \, 490, 492, 495
- \A, 503
- \b, 497
- \B, 497
- \d, 495
- \D, 495
- \E, 492
- \Q, 492
- \s, 495
- \S, 495
- \w, 495
- \W, 495
- \Z, 503
- \z, 503
-], 490, 495
- ^, 490, 492, 493, 495, 502
- {, 490, 493
- |, 490, 491
- }, 490, 493
- +, 493
- A, 502
- i, 502
- modyfikator, 502
- s, 502
- U, 502
- x, 502
- metoda
 - add, 565, 568
 - character_set_name, 552
 - diff, 566
 - fetch_array, 557
 - fetch_object, 560
 - format, 566
 - GET, 62, 70, 111
 - get_charset, 552
 - getTimestamp, 571
 - POST, 62, 70, 83, 111, 112
 - prepare, 560, 561
 - query, 552, 557, 558
 - set_charset, 550
 - setDate, 564
 - setTime, 564
 - setTimeZone, 565
 - strtotime, 565
 - sub, 566

model obiektowy dokumentu, *Patrz*: DOM
modyfikator, 265
MySQL, 18, 19, 23, 139, 231, 293, 549
 aplikacja, 149
 instalacja, 661, 662, 665
 testowanie, 672, 673, 674
klient, 19, 149
 argument, 150
serwer, 19, 149, 150, 549
 hasło, 664, 667, 668
 interfejs, 153
 połączenie, 296, 297, 298, 549, 550, 551, 624
szyfrowanie, *Patrz*: szyfrowanie, funkcja
 szyfrująca, funkcja deszyfrująca
uruchamianie, 149, 150, 151, 152, 153
użytkownik, 667, 669, 670, 671
wersja, 20
wydajność, 20
mysql, *Patrz*: MySQL klient
mysqld, *Patrz*: MySQL server

N

nagłówek HTTP, 390, 391
NF, *Patrz*: postać normalna
Node, 511
Normal Form, *Patrz*: postać normalna
NOT NULL, 146, 165
notacja
 daty i czasu, 565
 z kropką, 232, 383
NULL, 40, 71, 75, 146

O

obiekt, 40
 DateTime, 564, 567
 metoda, 547, 548, *Patrz też*: metoda
 MySQLi, 550, 552
 tworzenie, 547, 550
 właściwość, 547, 548
obszar tekstowy, 66
odstęp biały, 36
operator
 @, 278
 Boolean, *Patrz*: operator logiczny
 dekrementacji, 49
 dodawania, 49
 dzielenia, 49

dzielenia modulo, 49
hierarchia, 51
inkrementacji, 49
konkatenacji, 52, 385, *Patrz też*: znak ., znak +
logiczny, 71, 255, 256, 257
matematyczny, 49
mnożenia, 49
negacji, 71, *Patrz też*: znak !
odejmowania, 49
porównania, 71
przypisania, 51, *Patrz też*: znak +=
przypisania i konkatenacji, 48, *Patrz też*:
 znak .=
trójargumentowy, 350

P

para klucz-wartość, 81
pętla
 do...while, 98
 for, 96, 97, 98
 foreach, 85, 86, 88
 while, 96, 97, 98, 310
PHP, 13, 293, 382, 510
 buforowanie wyjścia, 615
 dokumentacja, *Patrz*: dokumentacja PHP
 instalacja, 661, 662, 665
 testowanie, 672
 konfigurowanie, 675, 676, 677, 687
 display_errors, 59, 276, 675
 rozszerzenie, 675
 Filter, *Patrz*: Filter
 mod_rewrite, 687
 sterowanie wyjściem, 615
 wersja, 53, 59, 273
PHP 5.3, 12, 14
PHP 6, 12
PHP 7, 12, 14
 funkcje, 137
 sugerowanie typów, 137
phpMyAdmin, 149
 konfiguracja, 154, 155
 transakcje, 262, 263
 uruchamianie, 153, 154, 155
plik
 .htaccess, 371, 683, 685
 .html, 102, 104
 .inc, 104
 .php, 102, 104
 FileInfo, 457, 459, 460

- httpd.conf, 678, 680, 683
- JavaScript, 383, 384, 385
- mysqli_connect.php, 296, 298, 299
- nagłówkowy, 294, 410
- przesyłanie, 370, 376, 377, 378, 381
- stopki, 116
- walidacja według typu, *Patrz*: typ walidacja
- zewnętrzny, 102, 103, 104, 136, 515
 - rozszerzenie, 102, 104, 107
 - ścieżka, *Patrz*: ścieżka
- poczta elektroniczna, *Patrz*: e-mail
- pole
 - tekstowe, 62, 65, 66, 143
 - HTML5, 115
 - ukryte, 332, 336, 341
 - wejściowe HTML, 332
 - wyboru, 62, 65
- polecenie
 - @charset, 31
 - ALTER, 250, 251, 258, 310, 478
 - CREATE, 158
 - DELETE, 179, 180, 301, 310, 469, 553
 - DESCRIBE, 160
 - FLUSH PRIVILEGES, 291
 - INNER, 232, 233, 235
 - INSERT, 161, 165, 210, 258, 301, 310, 469, 553
 - MODIFY, 478
 - OPTIMIZE, 258
 - przygotowane, 468, 469, 470, 471, 560, 561
 - dowiązywanie parametrów, 471
 - efektywność, 468
 - REPLACE, 165
 - SELECT, 166, 167, 181, 207, 210, 242, 301, 310, 469
 - COLLATE, 216
 - SIMPLE, 259
 - WHERE, *Patrz*: wyrażenie warunkowe WHERE
 - zliczanie zwróconych rekordów, 319
 - SHOW, 160
 - CHARACTER SET, 212
 - COLLATION, 213
 - ENGINES, 211
 - TRUNCATE, 180, 329
 - UNION, 237, 259
 - UPDATE, 177, 178, 258, 301, 310, 322, 469, 553
 - wstrzykiwanie, *Patrz*: atak wstrzykiwanie SQL
- postać normalna, 194
 - druga, 200, 201, 202
 - pierwsza, 197, 198
 - trzecia, 203, 204

- procedura obsługi zdarzeń, 521
- program MySQL Workbench, 149, 197
- programowanie
 - obiektowe, 545, 546, *Patrz też*: język
 - obiektowy
 - proceduralne, 546
- protokół
 - HTTP, 407, 423
 - HTTPS, 423
 - SSL, 476
- przeglądarka przesyłanie danych, 32, 33, 34
- przestrzeń nazw, 12
- przycisk opcji, 65, 115, 124, 125, 126

R

- RDBMS, 18
- regular expression, *Patrz*: wyrażenie regularne
- relacja, 197
- rozszerzenie Filter, *Patrz*: Filter

S

- salt, *Patrz*: modyfikator
- samołączenie, 238, 239
- selektor CSS, 518
- self-joins, *Patrz*: samołączenie
- serwer
 - Apache, 678
 - domyślna strony katalogu, 684
 - wirtualny host, 679, 680, 681
 - MySQL, *Patrz*: MySQL serwer
 - obsługa przesyłania plików, 370, 372, 373, 374, 375
 - poczty elektronicznej, 364, 365
 - SMTP, 364, 676
 - WWW, 21, 31, 364
 - instalacja, 661, 662, 665
 - zdalny, 30, 31
- sesja, 407, 430, 537
 - bezpieczeństwo, *Patrz*: bezpieczeństwo sesja
 - błąd, 437
 - dane, *Patrz*: sesja zmienne
 - identyfikator, 437, 438, 441
 - konfigurowanie, 430
 - procedura oczyszczająca, 435
 - przejęcie, *Patrz*: bezpieczeństwo sesja
 - rozpoczynanie, 431, 432, 585
 - status, 435
 - utrwalanie, 441

- sesja
 - zmienne, 437, 438, 439, 441
 - dostęp, 433, 434, 435
 - usuwanie, 435, 436, 437
- session fixation, *Patrz:* sesja utrwalanie
- sieć dostarczania treści, *Patrz:* CDN
- single-access token, *Patrz:* token
- skrót, 163, 475, 476, 650
 - tworzenie, 477
- skrypt
 - działający na serwerze, 535, 537
 - JSON, 543
 - generujący odwołania do funkcji JavaScript, 386, 387, 388, 389
 - konfiguracyjny, 620, 624, 627
 - pośredniczący, 371, 390
 - przekazywanie wartości, 332
 - przetwarzanie, 30, 31
 - tworzenie, 29
- słowo kluczowe, *Patrz też:* klauzula, polecenie, wyrażenie warunkowe
 - AS, 183
 - DISTINCT, 243
 - else, *Patrz:* instrukcja warunkowa else
 - elseif, *Patrz:* instrukcja warunkowa elseif
 - EXPLAIN, 258, 259
 - EXPLAIN EXTENDED, 261
 - FALSE, 170
 - if, *Patrz:* instrukcja warunkowa if
 - OUTER, 239
 - SIMPLE, 259
 - TRUE, 170
 - var, 522
- spam, 365
 - sygnałatory, 445, 446
 - zapobieganie, 444, 445, 446, 450, 452
 - czarna lista, 451
- SPL, 548
- SQL, 157, 231
 - polecenie, *Patrz:* polecenie
 - wstrzykiwanie, *Patrz:* atak wstrzykiwanie SQL
- stała
 - E_ALL, 279, 280
 - FILEINFO_MIME_TYPE, 457
 - MYSQL_ASSOC, 310
 - MYSQL_BOTH, 310
 - MYSQL_NUM, 310
 - PASSWORD_DEFAULT, 477
 - PHP_OS, 53
 - PHP_VERSION, 53, 59
 - predefiniowana, 53
 - tworzenie, 52, 53
- Standard PHP Library, *Patrz:* SPL
- strona, *Patrz też:* witryna
 - dynamiczna, 12, 15, 101, 461
 - logowania, *Patrz:* logowanie
 - statyczna, 17, 101
 - tworzenie, 591, 592, 597, 598
- system
 - dopasowywania wzorców, 485
 - operacyjny, 53, 54
 - zarządzania relacyjnymi bazami danych, *Patrz:* RDBMS
- szablon, 103, 109, 293, 583, 584, 614
- szyfrowanie, 265, 266
 - modyfikator, *Patrz:* modyfikator

Ś

- ścieżka, 102
- środowisko programistyczne zintegrowane, 21

T

- tablica, 40, 81, 451, *Patrz też:* zmienna
 - \$GLOBALS, 136
 - asocjacyjna, 89
 - element, 82, 83
 - dostęp, 85, 86
 - liczba, 88
 - wyświetlanie, 84, 135
 - indeks, *Patrz:* klucz
 - indeksowana, 81
 - klucz, *Patrz:* klucz
 - konwersja na łańcuch, 92
 - skojarzeniowa, 81
 - sortowanie, 92, 93, 95
 - superglobalna, 82
 - tworzenie, 85
 - wielowymiarowa, 88, 89, 90
 - sortowanie, 95
 - tworzenie, 89, 90, 91
- technologia
 - Ajax, *Patrz:* Ajax
 - działająca po stronie serwera, 14, 17
 - przenośna, 14
- token, 483

transakcja, 210, 211, 264, 400
bazodanowa, 262
ograniczenia, 262
punkt zapisu, 264
wykonywanie, 262, 263, 400, 405

trigger, *Patrz:* wyzwalacz

typ

BIGINT, 143
BINARY, 145
CHAR, 143, 144
DATE, 142, 143, 163
DATETIME, 143, 144, 198
DECIMAL, 143
DOUBLE, 143
ENUM, 142, 143, 147, 174
FileInfo, 457, 459, 460
FLOAT, 143
INT, 143
logiczny, 40
LONGBLOB, 145
LONGTEXT, 143
łańcuchowy, *Patrz:* łańcuch
MEDIUMBLOB, 145
MEDIUMINT, 143
MEDIUMTEXT, 143, 144
MIME, 391, 457
nieskalarny, 40, 81
NULL, *Patrz:* NULL
numeryczny, 147
Object, 537
rzutowanie, 452, 453, 456, 464
SET, 142, 143
skalarny, 40
SMALLINT, 143, 145
TEXT, 143, 148
TIME, 142, 143, 163
TIMESTAMP, 143, 145, 148, 163
TINYBLOB, 145
TINYINT, 143, 145
TINYTEXT, 143, 215
UNSIGNED, 147, 148
VARBINARY, 145, 265
VARCHAR, 143, 144, 215
walidacja, 451, 456, 457
ZEROFILL, 147

U

unia, 237, 259
Unicode, 28

W

wartość

interpretowana, 55
zmiennej, *Patrz:* zmienna wartość

witryna, *Patrz też:* strona

struktura, 104, 299

wtyczka, 510, 523

wylogowanie, 426, 428, 429, 435, 437, 643, 648

wyrażenie

regularne, 184, 445, 456, 485, 486, 686, 633, 634
leniwe, 499
ogranicznik, 486, 490
PCRE, 686, *Patrz też:* wyrażenie regularne
Perl, 485, 686, *Patrz też:* wyrażenie regularne
POSIX, 489
rozkład łańcucha, 501
wzorzec, *Patrz:* wzorzec
zachłanność, 499, 502
warunkowe, *Patrz też:* instrukcja warunkowa
LIKE, 171, 172, 250
maska, 171, 255
NOT LIKE, 171, 172
WHERE, 168, 169, 170, 183, 185, 207, 208, 243, 250

wyzwalacz, 229

wzorzec

adresu e-mail, 490, 496, 497
alternacja, 491
dopasowywanie, *Patrz:* funkcja preg_match, funkcja preg_match_all
dopasowywanie i zastępowanie, *Patrz:* preg_replace
granica, 497
klasa znaków, 495, 496, 497
negatywna, 499
koniec, 490
kwantyfikatory, 493
modyfikator, 502, *Patrz też:* metaznak
odwołanie wstecz, 504, 507
operator negacji, 495
początek, 490
rozgałęzienie, *Patrz:* wzorzec alternacja
słowo, 497
tworzenie, 486, 487, 490, 491, 492
literał, 490
metaznak, *Patrz:* metaznak
zastępowanie, 504

X

XAMPP, 662, 878
instalacja, 663, 664, 665
XHR, 532

Z

zależność
jeden do jednego, 196, 202
jeden do wielu, 196, 202
przerwanie, 223
wiele do wielu, 196, 202
zapora sieciowa, 662
zapytanie, 166, 301, 302, 456, 552, 553, 554, 557, 593, 599, *Patrz też*: polecenie, indeks
bezpieczeństwo, 314
dotyczące większej liczby tabel, *Patrz*: złączenie
klauzula, *Patrz*: klauzula, wyrażenie
warunkowe
ograniczanie wyników, 175
optymalizacja, 258
parametr
wchodzący, 564
wychodzący, 564
sortowanie, 173, 207, 208
wynik, 310, 311, 312, 558
stronicowanie, 349, 350, 356
zaawansowane, 245, 246, 247, 248, 249
zliczanie zwróconych rekordów, 319
zdarzenie
change, 521
click, 521
focus, 521
mouseover, 521
obsługa, 521, 522
onload, 517
ready, 515, 520
select, 521
submit, 521
złączenie, 208, 232
notacja z kropką, *Patrz*: notacja z kropką
składnia, 232
tabeli z nią samą, *Patrz*: samozłączenie
trzech lub większej liczby tabel, 239, 240
wewnętrzne, 232, 233, 234, 235
wynik
grupowanie, 242, 243, 244, 245
tabela wirtualna, 232, 239, 243
zewnętrzne, 232, 236, 237, 239

lewe, 236, 237
pełne, 236, 237
prawe, 236

zmienna, 40
\$_COOKIE, 82
\$_COOKIE, 420, 437
\$_ENV, 82
\$_FILE, 457
\$_FILES, 376
\$_GET, 136, 332
\$_POST, 82, 136, 332, 450
\$_REQUEST, 70, 75, 82, 136, *Patrz też*: znak ??
\$_SERVER, 82, 111
\$_SESSION, 82
\$_SESSION, 431, 435, 437
deklarowanie, 522
globalna, 136
nazwa, 42, 43
superglobalna, 136
tworzenie, 42, 43
typ, *Patrz*: typ
wartość, 46, 55, 56, 57
zasięg, 123, 136
znacznik
fieldset, 64
form, 62, 64
HTML, 36, 584
legend, 64
option, 65
PHP, 29, 30
zamykający, 298
script, 383, 513, 514
select, 65
usuwanie, 461, 463
w stylu XML, *Patrz*: znacznik PHP
znak, *Patrz też*: metaznak
-, 49, 255
--, 49
!, 71, 486, *Patrz też*: operator negacji
!=, 71, 168
"" "", 255
%, 49
&&, 71, 74, 168
(), 255
*, 49, 166, 170, 244, 245, 255, 256
., *Patrz*: operator konkatencji
.=, *Patrz*: operator przypisania i konkatencji
/, 486
?, 350
??, 75, *Patrz też*: zmienna \$_REQUEST

?>, 298	=, 71
@, 278	>=, 71, 168
\', 55	apostrof, 32, 34, 38, 44, 55, 56, 57, 161, 162
"", 55	odwrotny, 165
\\$, 55	biały, <i>Patrz</i> : odstęp biały
\\, 55	cudzysłów, 32, 34, 38, 44, 55, 56, 57
\n, <i>Patrz</i> : znak nowego wiersza	nowego wiersza (\n), 36, 42, 55
\r, 55	specjalny, 32, 55
\t, 55	Unicode, 12, <i>Patrz też</i> : Unicode, kodowanie
, 486	UTF-8
ll, 71, 74, 168	
+, 49, 255, 385	
++, 49	
+-, <i>Patrz też</i> : operator przypisania	
<=, 71, 168	

Ż

żądanie HTTP, 531, 532
 obsługa, 531, 532, 536, 538, 543
 żeton jednokrotnego dostępu, 483

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion



Szybki start ▶

PHP i MySQL — wydajne, niezawodne, stabilne działanie!

Język PHP i serwer bazy danych MySQL stały się nieformalnym standardem tworzenia dynamicznych witryn, które korzystają z baz danych. Mimo że istnieje wiele innych konkurencyjnych technologii, rzesza programistów aplikacji internetowych decyduje się właśnie na to rozwiązanie. Otwarte źródła, dostępność, coraz szersze możliwości, elastyczność i szybkość, konsekwentny rozwój — to tylko kilka zalet PHP i MySQL. Projektanci dynamicznych stron WWW wysoce sobie cenią te zalety, o czym świadczy choćby stale wzrastająca liczba witryn WWW, które napisano z wykorzystaniem tych technologii.

Ta książka jest doskonałym poradnikiem dla projektantów dynamicznych stron WWW. Szczególnie przydatna okaże się dla osób dopiero nabierających wprawy w programowaniu. W przejrzysty sposób opisano podstawy języków PHP oraz SQL. Przedstawiono zasady poprawnego konfigurowania serwerów PHP i MySQL. Uwzględniono zagadnienia dotyczące obsługi i usuwania błędów, a także przeprowadzania testów aplikacji. Szeroko potraktowano kwestie bezpieczeństwa aplikacji. Znalazło się tu również sporo informacji o integracji PHP z takimi technologiami jak JavaScript, jQuery, Perl czy Ajax. Przy tym wszystkim książka jest napisana w sposób bardzo przystępny, a liczne przykłady znakomicie ułatwiają zrozumienie prezentowanych treści.

Najważniejsze zagadnienia przedstawione w książce:

- Podstawy PHP i MySQL
- Zaawansowane zagadnienia i tworzenie zoptymalizowanego kodu
- Ukryte pola, stronicowanie wyników i inne techniki programistyczne
- Korzystanie z ciasteczek i sesje
- Zabezpieczenia aplikacji

Larry Ullman jest prezesem spółki Digital Media and Communications, firmy specjalizującej się w technologiach informacyjnych. Jest też autorem świetnych książek, programistą, trenerem, mówcą i konsultantem. Programuje dynamiczne strony WWW i zajmuje się bazami danych. Posiada rzadki dar tłumaczenia złożonych tajników programowania w prosty, zrozumiały i bardzo użyteczny sposób. Mimo to uparcie twierdzi, że z całą pewnością nie jest geekiem.

Helion	Sprawdź nasze szkolenia!	KOD KORZYŚCI Ściągnij po więcej! ▶	
helion.pl		ISBN 978-83-283-4466-2	
0 801 339900	AKADEMIA IT & BUSINESS		
0 601 339900	WWW.SZKOLENIA.HELION.PL	9 788328 344662	
INFORMATYKA W NAJLEPSZYM WYDANIU			Cena: 99,00 zł