


O'REILLY®

Zawiera informacje na temat HTML5, CSS3 i jQuery

Wydanie IV



# PHP, MySQL i JavaScript

## Wprowadzenie

PRZEWODNIK TWÓRCY STRON I APLIKACJI SIECIOWYCH!



Helion

Robin Nixon

Tytuł oryginału: Learning PHP, MySQL & JavaScript, 4th Edition

Tłumaczenie: Piotr Cieślak

ISBN: 978-83-283-0842-8

© 2015 Helion S.A.

Authorized Polish translation of the English edition of Learning PHP, MySQL & JavaScript, 4th Edition, ISBN 9781491918661 © 2015 Robin Nixon.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

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 the Publisher.

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 Wydawnictwo HELION 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.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/phmyj4>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/phmyj4.zip>

Printed in Poland.

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

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

---

# Spis treści

<b>Przedmowa .....</b>	<b>21</b>
<b>1. Wstęp do dynamicznych stron internetowych .....</b>	<b>25</b>
HTTP i HTML: podstawy wynalazku Bernersa-Lee	26
Procedura żądanie/odpowiedź	26
Zalety PHP, MySQL, JavaScriptu, CSS i HTML5	28
Zastosowanie PHP	29
Zastosowanie MySQL	30
Zastosowanie JavaScriptu	31
Zastosowanie CSS	32
I HTML5 na dokładkę	33
Serwer WWW Apache	34
Kilka słów o Open Source	35
Zgrany zespół	35
Pytania	37
<b>2. Konfigurowanie serwera .....</b>	<b>39</b>
WAMP, MAMP, LAMP — a cóż to takiego?	39
Instalowanie pakietu XAMPP w systemie Windows	40
Testowanie instalacji	47
Instalowanie pakietu XAMPP w Mac OS X	49
Dostęp do głównego foldera	49
Instalowanie pakietu LAMP pod Linuxem	50
Praca zdalna	50
Logowanie	50
Obsługa FTP	51
Obsługa edytora kodu	51
Obsługa środowiska IDE	52
Pytania	54
<b>3. Wstęp do PHP .....</b>	<b>55</b>
Dodawanie elementów PHP do kodu HTML	55
Przykłady z tej książki	56

Składnia PHP	57
Zastosowanie komentarzy	57
Podstawowa składnia	58
Zmienne	59
Operatory	63
Przypisywanie wartości zmiennym	66
Instrukcje wielowierszowe	68
Deklaracja typu zmiennych	70
Stałe	71
Stałe predefiniowane	71
Różnica między instrukcjami echo i print	72
Funkcje	73
Zasięg zmiennych	74
Pytania	78
<b>4. Wyrażenia i sterowanie działaniem programu w PHP .....</b>	<b>81</b>
Wyrażenia	81
Prawda czy fałsz?	81
Literały i zmienne	83
Operatory	84
Priorytet operatorów	84
Asocjacyjność	86
Operatory relacji	87
Wyrażenia warunkowe	91
Instrukcja if	91
Instrukcja else	92
Instrukcja elseif	93
Instrukcja switch	95
Operator ?	97
Pętle	98
Pętla while	98
Pętla do ... while	100
Pętla for	101
Przerywanie pętli	102
Instrukcja continue	103
Rzutowanie jawne i niejawne	104
Dynamiczne linkowanie w PHP	105
Dynamiczne linkowanie w praktyce	105
Pytania	106
<b>5. Funkcje i obiekty w PHP .....</b>	<b>107</b>
Funkcje PHP	107
Definiowanie funkcji	109
Zwracanie wartości	110
Zwracanie tablicy	111
Nie przekazuj argumentów przez referencję	111
Zwracanie zmiennych globalnych	113
Przypomnienie informacji o zasięgu zmiennych	114

Dołączanie i wymaganie plików	114
Instrukcja include	114
Zastosowanie instrukcji include_once	114
Zastosowanie instrukcji require i require_once	115
Sprawdzanie zgodności wersji PHP	115
Obiekty w PHP	116
Terminologia	116
Deklarowanie klasy	118
Tworzenie obiektu	118
Odwoływanie się do obiektów	119
Klonowanie obiektów	120
Konstruktory	121
Destruktry w PHP 5	122
Tworzenie metod	122
Metody statyczne w PHP 5	123
Deklarowanie właściwości	123
Deklarowanie stałych	124
Zasięg właściwości i metod w PHP 5	124
Właściwości i metody statyczne	125
Dziedziczenie	126
Pytania	130
<b>6. Tablice w PHP .....</b>	<b>131</b>
Prosty dostęp	131
Tablice indeksowane numerycznie	131
Tablice asocjacyjne	133
Dodawanie pozycji do tablicy przy użyciu słowa kluczowego array	133
Pętla foreach ... as	134
Tablice wielowymiarowe	136
Zastosowanie funkcji do obsługi tablic	139
is_array	139
count	139
sort	139
shuffle	140
explode	140
extract	141
compact	142
reset	143
end	143
Pytania	143
<b>7. PHP w praktyce .....</b>	<b>145</b>
Zastosowanie funkcji printf	145
Określanie precyzji	146
Dopełnianie łańcuchów tekstowych	148
Zastosowanie funkcji sprintf	149
Funkcje do obsługi daty i czasu	149
Stałe związane z datą	150
Zastosowanie funkcji checkdate	150

Obsługa plików	152
Sprawdzanie istnienia pliku	152
Tworzenie pliku	152
Odczytywanie zawartości plików	153
Kopiowanie plików	155
Przenoszenie pliku	155
Kasowanie pliku	155
Aktualizowanie plików	156
Ochrona plików przed wielokrotnym otwarciem	157
Odczytywanie całego pliku	158
Wysyłanie plików	159
Wywołania systemowe	163
XHTML czy HTML5?	165
Pytania	166
<b>8. Wstęp do MySQL .....</b>	<b>167</b>
Podstawy MySQL	167
Podsumowanie pojęć dotyczących baz danych	168
Dostęp do MySQL z poziomu wiersza poleceń	168
Uruchamianie wiersza poleceń	168
Obsługa serwera z poziomu wiersza poleceń	172
Instrukcje MySQL	173
Typy danych	177
Indeksy	185
Tworzenie indeksu	185
Tworzenie zapytań do bazy MySQL	190
Łączenie tabel	198
Zastosowanie operatorów logicznych	201
Funkcje MySQL	201
Dostęp do MySQL za pośrednictwem aplikacji phpMyAdmin	201
Pytania	203
<b>9. Zaawansowana obsługa MySQL .....</b>	<b>205</b>
Projektowanie bazy	205
Klucze główne, czyli kluczowy element relacyjnych baz danych	206
Normalizacja	207
Pierwsza postać normalna	207
Druga postać normalna	209
Trzecia postać normalna	212
Kiedy nie stosować normalizacji	214
Relacje	214
Jeden do jednego	214
Jeden do wielu	215
Wiele do wielu	216
Bazy danych i anonimowość	217
Transakcje	217
Mechanizmy składowania danych z obsługą transakcji	218
Instrukcja BEGIN	219

Instrukcja COMMIT	219
Instrukcja ROLLBACK	219
Instrukcja EXPLAIN	220
Archiwizacja i przywracanie danych	221
Instrukcja mysqldump	221
Tworzenie pliku z kopią zapasową	222
Odtwarzanie danych z pliku kopii zapasowej	224
Zapisywanie danych w formacie CSV	225
Planowanie tworzenia kopii zapasowych	225
Pytania	226
<b>10. Korzystanie z MySQL za pośrednictwem PHP .....</b>	<b>227</b>
Tworzenie zapytań do bazy MySQL za pośrednictwem PHP	227
Proces	227
Tworzenie pliku logowania	228
Nawiązywanie połączenia z MySQL	229
Praktyczny przykład	233
Tablica \$_POST	235
Usuwanie rekordu	236
Wyświetlanie formularza	237
Wysyłanie zapytań do bazy danych	237
Działanie programu	238
MySQL w praktyce	239
Tworzenie tabeli	239
Wyświetlanie informacji o tabeli	240
Usuwanie tabeli	241
Dodawanie danych	241
Odczytywanie danych	242
Aktualizowanie danych	242
Usuwanie danych	243
Zastosowanie opcji AUTO_INCREMENT	243
Wykonywanie zapytań pomocniczych	245
Zapobieganie próbom ataków	246
Działania prewencyjne	247
Zastosowanie elementów zastępczych	248
Zapobieganie przekazywaniu niepożądanych danych przez HTML	249
Proceduralny wariant zastosowania mysqli	251
Pytania	252
<b>11. Obsługa formularzy .....</b>	<b>253</b>
Tworzenie formularzy	253
Odczytywanie przesłanych danych	254
Opcja register_globals — rozwiązanie przestarzałe, ale wciąż spotykane	256
Wartości domyślne	257
Rodzaje pól	258
Oczyszczanie danych wejściowych	264
Przykładowy program	266

Co nowego w HTML5?	268
Atrybut autocomplete	269
Atrybut autofocus	269
Atrybut placeholder	269
Atrybut required	269
Atrybuty nadpisania	270
Atrybuty width i height	270
Funkcje oczekujące na pełną implementację	270
Atrybut form	270
Atrybut list	271
Atrybuty min oraz max	271
Atrybut step	271
Pole wejściowe typu color	272
Pola wejściowe typu number i range	272
Selektory daty i czasu	272
Pytania	272
<b>12. Ciasteczka, sesje i autoryzacja .....</b>	<b>275</b>
Zastosowanie ciasteczek w PHP	275
Tworzenie ciasteczka	276
Dostęp do ciasteczka	277
Usuwanie ciasteczek	277
Autoryzacja HTTP	278
Przechowywanie loginów i haseł	281
„Solenie”	281
Obsługa sesji	285
Inicjowanie sesji	285
Kończenie sesji	288
Określanie czasu trwania sesji	289
Bezpieczeństwo sesji	289
Pytania	292
<b>13. Zapoznanie z JavaScriptem .....</b>	<b>293</b>
JavaScript i tekst w HTML	293
Zastosowanie skryptów w nagłówku dokumentu	295
Starsze i niestandardowe przeglądarki	295
Dołączanie plików JavaScript	296
Debugowanie kodu JavaScript	297
Zastosowanie komentarzy	299
Średniki	299
Zmienne	299
Zmienne znakowe	300
Zmienne numeryczne	300
Tablice	300
Operatory	301
Operatory arytmetyczne	301
Operatory przypisania	302
Operatory porównania	302



Operatory logiczne	302
Inkrementacja i dekrementacja zmiennych	303
Konkatenacja łańcuchów znaków	303
Znaki modyfikujące	303
Typowanie zmiennych	304
Funkcje	305
Zmienne globalne	305
Zmienne lokalne	306
Obiektowy model dokumentu	307
Ale to nie takie proste...	308
Kolejne zastosowanie symbolu \$	309
Zastosowanie obiektowego modelu dokumentu	309
Kilka słów o document.write	310
Zastosowanie funkcji console.log	310
Zastosowanie funkcji alert	311
Umieszczanie tekstu w elementach HTML	311
Zastosowanie funkcji document.write	311
Pytania	312
<b>14. Wyrażenia i sterowanie działaniem programu w JavaScriptcie .....</b>	<b>313</b>
Wyrażenia	313
Literały i zmienne	314
Operatory	315
Priorytet operatorów	315
Asocjacyjność	316
Operatory relacji	316
Instrukcja with	319
Zdarzenie onerror	320
Konstrukcja try ... catch	321
Wyrażenia warunkowe	322
Instrukcja if	322
Instrukcja else	322
Instrukcja switch	323
Operator ?	324
Pętle	325
Pętla while	325
Pętla do ... while	326
Pętla for	326
Przerywanie pętli	327
Instrukcja continue	327
Typowanie jawne	328
Pytania	329
<b>15. Funkcje, obiekty i tablice w JavaScriptcie .....</b>	<b>331</b>
Funkcje w JavaScriptcie	331
Definiowanie funkcji	331
Tablica arguments	332

Zwracanie wartości	333
Zwracanie tablicy	334
Obiekty w JavaScriptcie	335
Deklarowanie klasy	335
Tworzenie obiektu	337
Dostęp do obiektów	337
Słowo kluczowe prototype	337
Tablice w JavaScriptcie	339
Tablice numeryczne	340
Tablice asocjacyjne	341
Tablice wielowymiarowe	341
Zastosowanie metod do obsługi tablic	342
Pytania	346
<b>16. Weryfikacja danych i obsługa błędów w JavaScriptcie i PHP .....</b>	<b>349</b>
Weryfikowanie wprowadzonych danych przy użyciu JavaScriptu	349
Dokument validate.html (część pierwsza)	350
Dokument validate.html (część druga)	352
Wyrażenia regularne	355
Dopasowywanie za pomocą metaznaków	355
Dopasowanie „rozmyte”	356
Grupowanie przy użyciu nawiasów	357
Klasy znaków	357
Określanie zakresu	358
Zaprzeczenie	358
Kilka bardziej skomplikowanych przykładów	358
Podsumowanie metaznaków	361
Modyfikatory ogólne	362
Zastosowanie wyrażen regularnych w JavaScriptcie	362
Zastosowanie wyrażen regularnych w PHP	363
Ponowne wyświetlenie formularza po weryfikacji w PHP	364
Pytania	369
<b>17. Zastosowanie technologii Ajax .....</b>	<b>371</b>
Czym jest Ajax?	372
Zastosowanie obiektu XMLHttpRequest	372
Twój pierwszy program Ajax	374
Zastosowanie metody GET zamiast POST	378
Przesyłanie żądań XML	380
Zastosowanie platform Ajax	384
Pytania	385
<b>18. Wstęp do CSS .....</b>	<b>387</b>
Importowanie arkusza stylów	388
Importowanie stylów CSS z poziomu HTML	388
Style zagnieżdżone	389
Zastosowanie identyfikatorów ID	389

Zastosowanie klas	389
Zastosowanie średników	389
Reguły CSS	390
Wiele deklaracji	390
Zastosowanie komentarzy	391
Rodzaje stylów	391
Style domyślne	392
Style użytkownika	392
Zewnętrzne arkusze stylów	393
Style wewnętrzne	393
Style bezpośrednie	393
Selektory CSS	393
Selektor typu	393
Selektor potomka	393
Selektor dziecka	394
Selektor identyfikatora	395
Selektor klasy	396
Selektor atrybutu	396
Selektor uniwersalny	397
Selekcja grupowa	398
Dziedziczenie kaskadowe	398
Źródła stylów	398
Metody definiowania reguł	399
Selektory arkuszy stylów	399
Obliczanie specyficzności	400
Różnica między elementami div i span	401
Jednostki miar	403
Fonty i typografia	405
font-family	405
font-style	406
font-size	406
font-weight	407
Zarządzanie stylami tekstu	407
Efekty tekstowe	407
Odstępy	408
Wyrównanie	408
Wielkość znaków	408
Wcięcia	408
Kolory w CSS	408
Skrócone określenia kolorów	409
Gradients	410
Rozmieszczanie elementów	411
Położenie bezwzględne	411
Położenie względne	412
Położenie stałe	412
Pseudoklasy	413
Skracanie reguł	415

Model pudełkowy i układ strony	416
Definiowanie marginesów	416
Definiowanie ramek	418
Definiowanie odstępu	419
Zawartość obiektu	420
Pytania	420
<b>19. Zaawansowane reguły CSS w CSS3 .....</b>	<b>423</b>
Selektory atrybutów	423
Dopasowywanie fragmentów łańcuchów	423
Właściwość box-sizing	425
Tła w CSS3	425
Właściwość background-clip	425
Właściwość background-origin	427
Właściwość background-size	427
Zastosowanie właściwości auto	428
Wiele obrazów w tle	428
Ramki w CSS3	430
Właściwość border-color	430
Właściwość border-radius	430
Cienie	433
Właściwość overflow	434
Układ wielokolumnowy	434
Kolory i przezroczystość	435
Kolory HSL	436
Kolory HSLA	436
Kolory RGB	437
Kolory RGBA	437
Właściwość opacity	437
Efekty tekstowe	438
Właściwość text-shadow	438
Właściwość text-overflow	438
Właściwość word-wrap	439
Fonty internetowe	439
Fonty Google	440
Przekształcenia	441
Przekształcenia 3D	442
Przejścia	443
Właściwości przejść	443
Czas trwania przejścia	444
Opóźnienie przejścia	444
Dynamika przejścia	444
Skrócona składnia	445
Pytania	446
<b>20. Dostęp do CSS z poziomu JavaScriptu .....</b>	<b>449</b>
Ponowne spotkanie z funkcją getElementById	449
Funkcja O	449
Funkcja S	450

Funkcja C	451
Dołączanie opisanych funkcji	451
Dostęp do właściwości CSS z poziomu JavaScriptu	452
Niektóre typowe właściwości	452
Inne właściwości	453
JavaScript w kodzie HTML	455
Słowo kluczowe this	455
Łączenie zdarzeń i obiektów w skrypcie	456
Odwoływanie się do innych zdarzeń	456
Dodawanie nowych elementów	457
Usuwanie elementów	459
Inne sposoby na dodawanie i usuwanie elementów	459
Zastosowanie przerw	460
Zastosowanie przerwania setTimeout	460
Anulowanie opóźnienia	461
Zastosowanie przerwania setInterval	461
Animacje na bazie przerw	463
Pytania	464
<b>21. Wprowadzenie do jQuery .....</b>	<b>467</b>
Dlaczego jQuery?	467
Dołączanie jQuery	468
Wybór odpowiedniej wersji	468
Pobieranie	469
Zastosowanie sieci dostarczania treści (CDN)	469
Zawsze najnowsza wersja	470
Dostosowywanie jQuery	471
Składnia jQuery	471
Prosty przykład	471
Unikanie konfliktów między bibliotekami	472
Selektory	473
Metoda css	473
Selektor elementów	474
Selektor identyfikatorów	474
Selektor klas	474
Łączenie selektorów	474
Obsługa zdarzeń	475
Oczekiwanie na gotowość dokumentu	476
Funkcje i właściwości związane ze zdarzeniami	477
Zdarzenia blur i focus	477
Słowo kluczowe this	478
Zdarzenia click i dblclick	479
Zdarzenie keypress	480
Przemysłane programowanie	481
Zdarzenie mousemove	482
Inne zdarzenia myszy	484
Inne metody związane z obsługą myszy	485
Zdarzenie submit	486

Efekty specjalne	487
Ukrywanie i wyświetlanie	488
Metoda toggle	489
Stopniowe zanikanie i wyświetlanie	489
Przesuwanie elementów w górę i w dół	490
Animacje	491
Zatrzymywanie animacji	494
Manipulowanie drzewem DOM	494
Różnica między metodami text i html	495
Metody val i attr	496
Dodawanie i usuwanie elementów	496
Dynamiczne stosowanie klas	499
Modyfikowanie wymiarów	499
Metody width i height	499
Metody innerWidth i innerHeight	502
Metody outerWidth i outerHeight	502
Nawigowanie w obrębie drzewa DOM	502
Elementy nadrzędne	503
Elementy potomne	506
Elementy siostrzane	507
Wybieranie poprzedzających i kolejnych elementów	508
Przetwarzanie selekcji w jQuery	509
Metoda is	511
Użycie jQuery bez selektorów	512
Metoda \$.each	512
Metoda \$.map	513
Zastosowanie technologii Ajax	514
Zastosowanie metody post	514
Zastosowanie metody get	514
Rozszerzenia	515
jQuery User Interface	515
Inne rozszerzenia	516
jQuery Mobile	516
Pytania	516
<b>22. Wstęp do HTML5 .....</b>	<b>519</b>
Obiekt canvas	520
Geolokacja	521
Dźwięk i filmy	523
Formularze	524
Magazyn danych	524
Web workers	525
Aplikacje sieciowe	525
Mikrodane	525
Podsumowanie	525
Pytania	526

<b>23. Obiekt canvas w HTML5 .....</b>	<b>527</b>
Tworzenie elementu canvas i dostęp do niego	527
Funkcja toDataURL	529
Określanie formatu obrazu	530
Metoda fillRect	530
Metoda clearRect	531
Metoda strokeRect	531
Łączenie wymienionych instrukcji	531
Metoda createLinearGradient	532
Szczegółowe informacje o metodzie addColorStop	534
Metoda createRadialGradient	535
Wypełnianie wzorkami	536
Umieszczanie napisów na elemencie canvas	538
Metoda strokeText	538
Własność textBaseLine	539
Własność font	539
Własność textAlign	539
Metoda fillText	540
Metoda measureText	541
Rysowanie linii	541
Własność lineWidth	541
Własności lineCap i lineJoin	541
Własność miterLimit	543
Kreślenie ścieżek	543
Metody moveTo i lineTo	544
Metoda stroke	544
Metoda rect	544
Wypełnianie obszarów	545
Metoda clip	546
Metoda isPointInPath	548
Zastosowanie krzywych	549
Metoda arc	550
Metoda arcTo	552
Metoda quadraticCurveTo	552
Metoda bezierCurveTo	554
Obsługa obrazków	555
Metoda drawImage	555
Skalowanie obrazu	555
Wybieranie fragmentu obrazu	556
Kopiowanie z elementu canvas	557
Tworzenie cieni	557
Przetwarzanie obrazu na poziomie pikseli	558
Metoda getImageData	559
Tablica data	560
Metoda putImageData	561
Metoda createImageData	562

Zaawansowane efekty graficzne	562
Własność globalCompositeOperation	562
Własność globalAlpha	564
Przekształcenia	564
Metoda scale	564
Metody save i restore	566
Metoda rotate	566
Metoda translate	567
Metoda transform	568
Metoda setTransform	570
Podsumowanie	570
Pytania	571
<b>24. Filmy i dźwięk w HTML5 .....</b>	<b>573</b>
O kodekach	574
Element <audio>	575
Wsparcie dla przeglądarek nieobsługujących HTML5	577
Element <video>	578
Kodeki wideo	578
Obsługa starszych przeglądarek	581
Podsumowanie	583
Pytania	583
<b>25. Inne funkcje HTML5 .....</b>	<b>585</b>
Geolokacja i usługi GPS	585
Inne sposoby lokalizacji	586
Geolokacja i HTML5	586
Magazyn lokalny	590
Zastosowanie magazynu lokalnego	590
Obiekt localStorage	591
Web workers	593
Aplikacje offline	594
Technologia przeciągnij i upuść	596
Komunikacja między dokumentami	598
Mikrodane	601
Inne znaczniki HTML5	603
Podsumowanie	603
Pytania	604
<b>26. Zastosowanie wszystkich omówionych technologii .....</b>	<b>605</b>
Projektowanie serwisu społecznościowego	605
Strona WWW z przykładami	606
functions.php	606
Funkcje	606
header.php	608
setup.php	609
index.php	611
signup.php	611



Sprawdzanie dostępności nazwy użytkownika	613
Logowanie	614
checkuser.php	614
login.php	615
profile.php	616
Dodawanie tekstu O mnie	617
Dodawanie zdjęcia profilowego	617
Przetwarzanie obrazu	618
Wyświetlanie bieżącego profilu	618
members.php	621
Wyświetlanie profilu użytkownika	622
Dodawanie i usuwanie znajomych	622
Wyświetlanie listy wszystkich użytkowników	622
friends.php	623
messages.php	626
logout.php	629
styles.css	629
javascript.js	632
<b>A Odpowiedzi na pytania kontrolne .....</b>	<b>635</b>
<b>B Zasoby internetowe .....</b>	<b>653</b>
Informacje na temat PHP	653
Informacje na temat MySQL	653
Informacje na temat JavaScriptu	654
Informacje na temat CSS	654
Informacje na temat HTML5	654
Informacje na temat technologii AJAX	654
Inne ciekawe strony WWW	655
Serwisy informacyjne wydawnictwa O'Reilly	655
<b>C Słowa z grupy stopwords w MySQL .....</b>	<b>657</b>
<b>D Funkcje MySQL .....</b>	<b>661</b>
Funkcje do obsługi łańcuchów znaków	661
Funkcje do obsługi daty	663
Funkcje do obsługi czasu	668
<b>E Selektory, obiekty i metody jQuery .....</b>	<b>671</b>
Selektory jQuery	671
Obiekty jQuery	674
Metody jQuery	676
<b>Skorowidz .....</b>	<b>689</b>



# Korzystanie z MySQL za pośrednictwem PHP

Jeśli zapoznałeś się z poprzednimi rozdziałami, to powinieneś już sprawnie posługiwać się MySQL i PHP. W tym rozdziale dowiesz się, w jaki sposób połączyć te dwie technologie za pomocą wbudowanych funkcji PHP umożliwiających bezpośredni dostęp do MySQL.

## Tworzenie zapytań do bazy MySQL za pośrednictwem PHP

Zasadniczym powodem stosowania PHP jako interfejsu dla MySQL jest możliwość sformatowania wyników zwróconych przez zapytania SQL w sposób pozwalający na wyświetlenie ich na stronie internetowej. Jeśli możesz się zalogować do MySQL za pomocą nazwy użytkownika i hasła, to możesz to zrobić także poprzez PHP.

Różnica polega na tym, że zamiast korzystać z wiersza poleceń MySQL do wprowadzania instrukcji i wyświetlania wyników, będziesz tworzył zapytania w postaci łańcuchów znaków, przekazywane do MySQL. Wynik zwrócony przez bazę nie zostanie sformatowany tak jak w przypadku pracy z wierszem poleceń, lecz będzie miał postać pewnej struktury, którą PHP potrafi zinterpretować. Za pomocą instrukcji PHP można odczytywać dane z tej struktury i wyświetlać je na stronie WWW.



W tym rozdziale z poprzednich wydań książki opisywałem stary sposób dostępu do bazy danych MySQL, wykorzystujący funkcję `mysql`; dopiero w kolejnym rozdziale omawiałem nowsze rozszerzenie o nazwie `mysqli`. Ale jak to mówią, czas nie stoi w miejscu; liczba systemów ze starszymi wersjami kodu i oprogramowania powinna być już relatywnie mała, w tym wydaniu postanowiłem więc od razu przystąpić do omawiania nowszego rozszerzenia — które obecnie stało się już praktycznie standardem.

## Proces

Proces komunikacji z MySQL za pomocą PHP wygląda tak:

1. nawiązanie połączenia z MySQL i wybór bazy danych,

2. utworzenie zapytania,
3. wykonanie zapytania,
4. pozyskanie rezultatów i wyświetlenie ich na stronie internetowej,
5. powtórzenie kroków od 2. do 4. aż do uzyskania wszystkich potrzebnych danych,
6. rozłączenie z MySQL.

Już za chwilę zajmiemy się tymi punktami krok po kroku, przede wszystkim jednak powinieneś skonfigurować procedurę logowania w bezpieczny sposób, by utrudnić postronnym osobom przechwycenie danych dostępowych do bazy.

## Tworzenie pliku logowania

Większość stron internetowych napisanych w PHP składa się z wielu skryptów wymagających dostępu do MySQL, a tym samym — loginu oraz hasła. Mając to na uwadze, dobrze jest utworzyć osobny plik zawierający te informacje i dołączać go do plików strony za każdym razem, gdy będzie to konieczne. Przykład 10.1 przedstawia tego rodzaju plik, który nazwałem *login.php*.

Przykład 10.1. Plik *login.php*

```
<?php //login.php
$hn = 'localhost';
$db = 'publications';
$un = 'uzytkownik';
$pw = 'haslo';
?>
```

Przepisz podany przykład, zastępując tymczasowe wartości, takie jak *uzytkownik* i *haslo*, rzeczywistymi parametrami dostępowymi do bazy MySQL, a następnie zapisz plik w folderze z projektem, który skonfigurowałeś w rozdziale 2. Już za chwilę z niego skorzystamy.

Nazwa hosta *localhost* powinna być poprawna, jeśli pracujesz na MySQL zainstalowanym na lokalnym komputerze, a jeżeli wykonywałeś ćwiczenia opisane w poprzednich rozdziałach, to na serwerze powinna się znajdować baza danych o nazwie *publications*.

Znaczniki `<?php` oraz `?>` są bardzo ważne zwłaszcza w przypadku pliku *login.php* przedstawionego w przykładzie 10.1, bo wiersze pomiędzy nimi mogą być zinterpretowane *wyłącznie* jako kod PHP. Jeśli byś je pominął, a ktoś odwołałby się do tego pliku bezpośrednio na Twojej stronie internetowej, to zostałby on po prostu wyświetlony w przeglądarce i zdradził Twoje sekrety. Dzięki zastosowaniu znaczników intruz zobaczy tylko pustą stronę. Tak przygotowany plik będzie można poprawnie dołączać do innych plików PHP.

Zmienna `$hn` informuje PHP, do jakiego serwera ma się odwołać przy połączeniu z bazą danych. To konieczne, gdyż za pośrednictwem PHP możesz nawiązać połączenie z dowolną bazą danych MySQL dostępną z poziomu Twojego systemu, w tym także z serwerami w internecie. Na potrzeby przykładów opisanych w tym rozdziale użyjemy jednak lokalnego serwera i zamiast podawać pełną nazwę domeny, taką jak *mysql.mojserwer.com*, wpiszemy po prostu *localhost* (lub adres IP *127.0.0.1*).

W zmiennej `$db` podajemy nazwę bazy, której będziemy używać — w tym przypadku będzie to baza *publications*, którą zapewne utworzyłeś w rozdziale 8., bądź baza udostępniona przez administratora serwera (wówczas musisz oczywiście odpowiednio zmodyfikować plik *login.php*).



Kolejną zaletą umieszczenia danych logowania w jednym miejscu jest łatwość aktualizacji projektu przy zmianie hasła: możesz to robić tak często, jak tylko masz ochotę, bo niezależnie od liczby plików PHP odwołujących się do MySQL będziesz musiał wprowadzić poprawkę tylko w jednym miejscu.

## Nawiązywanie połączenia z MySQL

Po zapisaniu pliku *login.php* można go dołączyć za pomocą dyrektywy `require_once` do dowolnego pliku z kodem PHP, który będzie wymagał dostępu do bazy danych. Takie rozwiązanie jest o tyle lepsze od użycia instrukcji `include`, że nieodnalezienie pliku z danymi logowania spowoduje wyświetlenie komunikatu o krytycznym błędzie. A wierz mi, brak pliku z danymi logowania do bazy *jest* krytycznym błędem.

Ponadto zastosowanie dyrektywy `require_once` zamiast `require` oznacza, że plik zostanie wczytany tylko wtedy, jeśli nie został dołączony już wcześniej, co pozwala zapobiec marnowaniu zasobów na niepotrzebne odwołania do dysku twardego. Przykład 10.2 przedstawia niezbędny fragment kodu.

*Przykład 10.2. Nawiązywanie połączenia z serwerem MySQL za pośrednictwem `mysqli`*

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
?>
```

W tym przykładzie został utworzony nowy obiekt o nazwie `$conn` poprzez wywołanie metody `mysqli`, do której zostały przekazane argumenty zaczerpnięte z pliku *login.php*. Za obsługę błędów odpowiada odwołanie do właściwości `$conn->connect_error`.



Funkcja `die` oddaje nieocenione usługi podczas programowania w PHP, ale na serwerze produkcyjnym warto przygotować specjalne komunikaty o błędach, które będą znacznie przyjaźniejsze w odbiorze. W takich przypadkach nie przerywa się działania programu PHP, ale odpowiednio formatuje komunikat i wyświetla go po zwykłym zakończeniu pracy programu, na przykład tak:

```
function mysql_fatal_error($msg)
{
    $msg2 = mysql_error();
    echo <<< _END
    Niestety nie udało się zrealizować zadania.
    Komunikat błędu:

    <p>$msg: $msg2</p>

    Kliknij przycisk Wstecz w przeglądarce i spróbuj ponownie.
    W razie dalszych problemów prosimy o wysłanie
    <a href="mailto:admin@serwer.com">maila do administratora</a>.
    Dziękujemy.
    _END;
}
```

Operator `->` oznacza, że element po jego prawej stronie jest właściwością albo metodą obiektu znajdującego się po stronie lewej. Gdyby w tym przypadku właściwość `connect_error` miała

jakąś wartość, oznaczałoby to wystąpienie błędu. Wywołujemy więc funkcję `die`, która wyświetli tę wartość — zawiera ona informacje o napotkanym błędzie.

Obiekt `$conn` będzie używany w kolejnych przykładach do uzyskania dostępu do bazy MySQL.

## Konstruowanie i wykonywanie zapytania

Aby wysłać zapytanie do MySQL przy użyciu PHP, wystarczy użyć metody `query` obiektu połączenia. Przykład 10.3 ilustruje, jak to zrobić.

*Przykład 10.3. Wysyłanie zapytania do bazy za pośrednictwem mysqli*

```
<?php
$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die($conn->error);
?>
```

W tym przypadku zmiennej `$query` został przypisany łańcuch tekstowy zawierający zapytanie do wykonania. Zmienna ta została następnie przekazana do metody `query` obiektu `$conn`. Z kolei metoda ta zwraca rezultat, który trafia do obiektu `$result`. Jeśli wartość `$result` wynosi `FALSE`, to znaczy, że wystąpił jakiś błąd, którego opis został zapisany we właściwości `error` obiektu połączenia. W takim przypadku wywołana zostaje funkcja `die`, która ten błąd wyświetli.

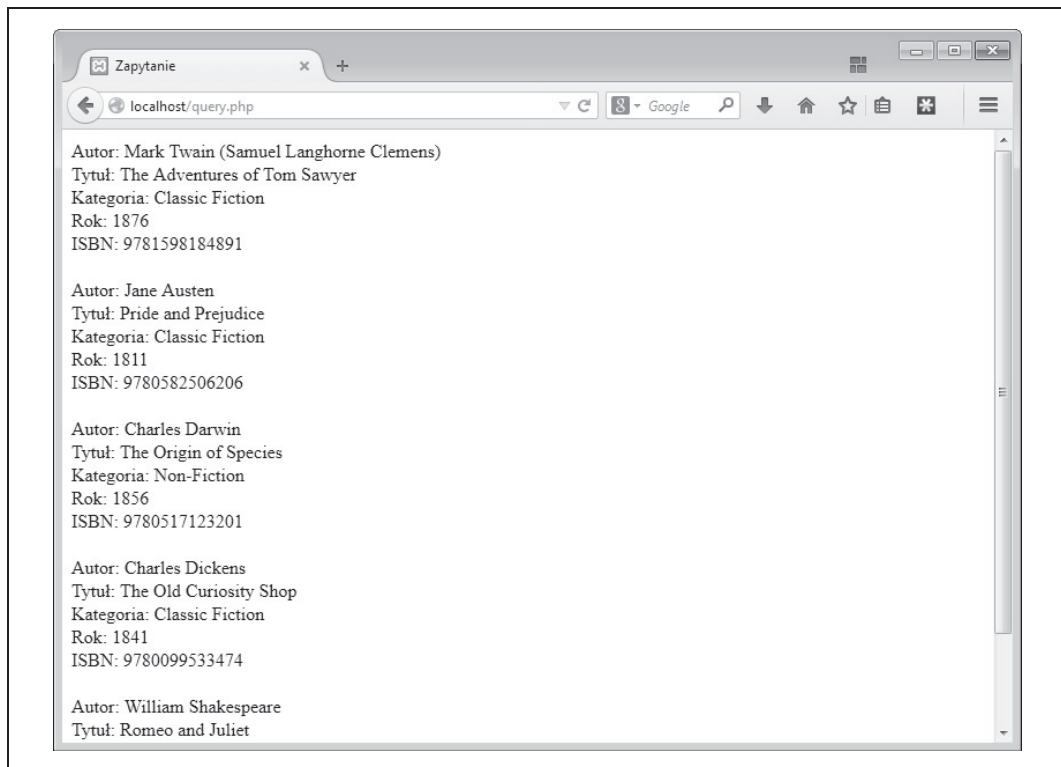
Wszystkie dane zwrócone przez MySQL trafią do obiektu `$result` w postaci, w której można je łatwo przetwarzać.

## Pobieranie rezultatu

Po zwróceniu rezultatu do obiektu `$result` możesz pobrać z niego potrzebne dane po kolei przy użyciu metody `fetch_assoc` tego obiektu. Przykład 10.4 stanowi połączenie i rozszerzenie dotychczasowych przykładów. Jest to już kompletny program, który możesz przepisać i uruchomić, aby pozyskać z bazy potrzebne dane (rysunek 10.1). Zalecam zapisanie go pod nazwą `query.php` w tym samym folderze co plik `login.php` (albo wykorzystanie pliku pobranego z darmowego archiwum przykładów, dostępnego na stronie [lpmj.net](http://lpmj.net)).

*Przykład 10.4. Zwracanie zawartości poszczególnych komórek*

```
<?php // query.php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die($conn->error);
$rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)
{
    $result->data_seek($j);
    echo 'Autor: ' . $result->fetch_assoc()['author'] . '<br>';
    $result->data_seek($j);
    echo 'Tytuł: ' . $result->fetch_assoc()['title'] . '<br>';
    $result->data_seek($j);
    echo 'Kategoria: ' . $result->fetch_assoc()['category'] . '<br>';
    $result->data_seek($j);
    echo 'Rok: ' . $result->fetch_assoc()['year'] . '<br>';
}
```



Rysunek 10.1. Rezultat działania programu `query.php` z przykładu 10.4

```

$result->data_seek($j);
echo 'ISBN: ' . $result->fetch_assoc()['isbn'] . '<br><br>';
}
$result->close();
$conn->close();
?>

```

W opisanym przykładzie w celu odwołania się do odpowiedniego wiersza danych przy każdej iteracji pętli użyliśmy metody `data_seek` obiektu `$result`. Następnie wykorzystaliśmy metodę `fetch_assoc` do pobrania wartości przechowywanych w poszczególnych komórkach i wreszcie wartości te wyświetliliśmy przy użyciu instrukcji `echo`.

Zapewne zgodzisz się ze mną, że takie wyszukiwanie danych jest dość uciążliwe i powinna być jakaś skuteczniejsza metoda na osiągnięcie podobnego efektu. Taki sposób rzeczywiście istnieje i polega na pobieraniu całych wierszy danych naraz.



W rozdziale 9. przeczytałeś o pierwszej, drugiej i trzeciej postaci normalnej bazy danych i być może zwróciłeś uwagę na fakt, że tabela `classics` nie spełnia warunków tych postaci, ponieważ zawiera ona informacje o autorze i o książce. Wynika to z faktu, że tabela `classics` została utworzona jeszcze przed omówieniem kwestii normalizacji. Jednak na potrzeby zilustrowania metod dostępu do MySQL z poziomu PHP ta tabela zupełnie wystarczy i pozwoli nam uniknąć żmudnego wpisywania kolejnych danych — proponuję więc się nią posłużyć.

## Pobieranie wiersza danych

Aby pobrać cały wiersz danych naraz, należy zastąpić pętlę `for` z przykładu 10.4 taką, jaka została wyróżniona pogrubieniem w przykładzie 10.5. Po wykonaniu tej zmiany powinieneś uzyskać taki sam efekt, jaki został pokazany wcześniej na rysunku 10.1. Tak zmodyfikowany plik możesz zapisać pod nazwą `fetchrow.php`.

*Przykład 10.5. Odczytywanie zawartości kolejnych wierszy*

```
<?php //fetchrow.php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die($conn->error);
$rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)
{
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_ASSOC);
    echo 'Autor: ' . $row['author'] . '<br>';
    echo 'Tytuł: ' . $row['title'] . '<br>';
    echo 'Kategoria: ' . $row['category'] . '<br>';
    echo 'Rok: ' . $row['year'] . '<br>';
    echo 'ISBN: ' . $row['isbn'] . '<br><br>';
}
$result->close();
$conn->close();
?>
```

W tak zmodyfikowanym przykładzie mamy do czynienia z pięciokrotnie mniejszą liczbą odwołań do obiektu `$result` (w porównaniu z poprzednim kodem), a w ramach każdej iteracji pętli następuje tylko jedno takie odwołanie, ponieważ przy użyciu metody `fetch_array` są pobierane całe wiersze kodu. Metoda ta zwraca cały wiersz danych w postaci tablicy, która w naszym programie jest następnie przypisywana do zmiennej `$row`.

Zależnie od przekazanych do niej wartości metoda `fetch_array` może zwrócić trzy rodzaje tablic:

### MYSQLI\_NUM

Tablica numeryczna. Poszczególne kolumny pojawiają się w tablicy zgodnie z kolejnością, w jakiej zostały utworzone w tabeli (z uwzględnieniem późniejszych zmian). W naszym przypadku na zerowej pozycji tablicy znajduje się kolumna *Author*, na pierwszej kolumna *Title* i tak dalej.

### MYSQLI\_ASSOC

Tablica asocjacyjna. Każdy klucz stanowi nazwę kolumny. Ponieważ w tym przypadku do danych trzeba się odwoływać za pośrednictwem nazwy kolumny (a nie numeru indeksu), warto korzystać z tego wariantu zawsze, gdy to możliwe, aby ułatwić sobie debugowanie programu, a innym programistom — interpretację kodu.

### MYSQLI\_BOTH

Tablica asocjacyjna i numeryczna.

Tablice asocjacyjne są zwykle bardziej praktyczne od numerycznych, gdyż umożliwiają odwoływanie się do kolumn za pomocą nazw, na przykład `$row['author']`, dzięki czemu nie trzeba



zapamiętywać, na którym miejscu w tabeli znajduje się potrzebna kolumna. W przykładowym skrypcie została użyta tablica asocjacyjna, o czym świadczy argument `MYSQLI_ASSOC`.

## Zamykanie połączenia

Po zakończeniu wykonywania skryptu PHP zwolni pamięć zaalokowaną na potrzeby obiektów, więc w przypadku niewielkich programów na ogół nie trzeba się troszczyć o samodzielne zarządzanie pamięcią. Jednak w przypadku większej liczby rezultatów albo dużych porcji danych dobrze jest zwolnić pamięć, której program już nie potrzebuje, aby uniknąć problemów z jego dalszym działaniem.

Jest to szczególnie istotne na często odwiedzanych stronach, gdyż ilość pamięci zajmowanej w trakcie sesji może raptownie rosnąć. Zwróć uwagę na metody `close` dla obiektów `$result` oraz `$conn`, które w poprzednich skryptach były wywoływane w sytuacji, gdy dany obiekt przestał już być potrzebny.

```
$result->close();  
$conn->close();
```



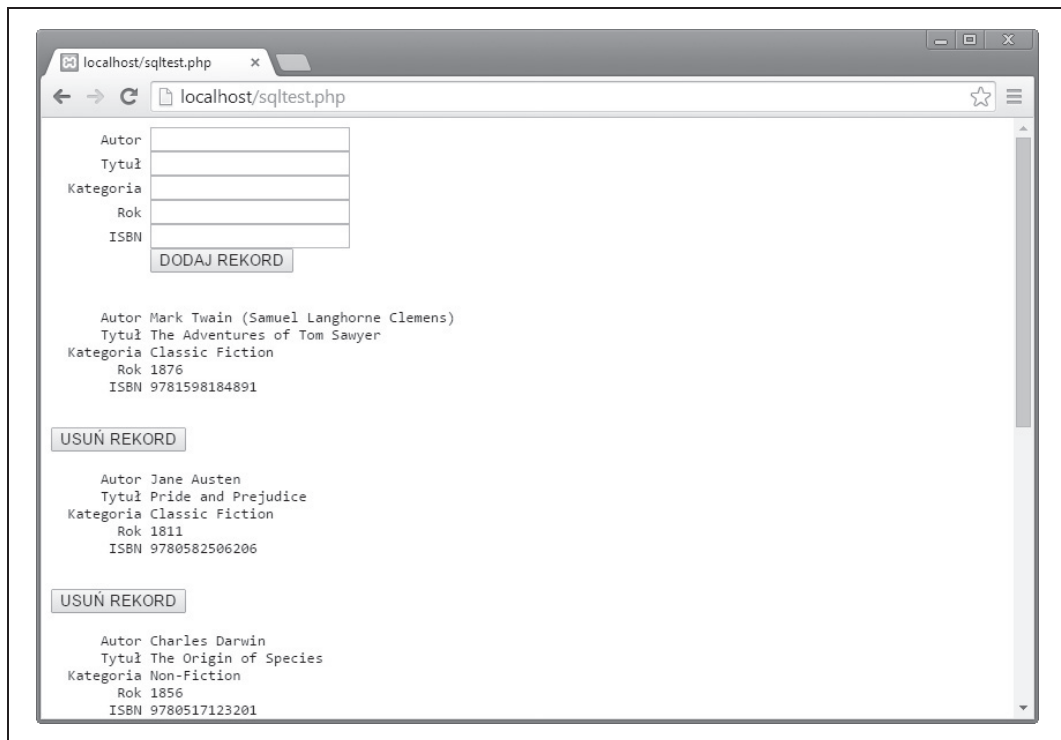
Najlepiej byłoby zamknąć każdy obiekt z wynikami zapytania, gdy przestanie być używany, a następnie zamknąć obiekt połączenia w chwili, gdy komunikacja z serwerem MySQL stanie się zbędna. Takie rozwiązanie gwarantuje zwolnienie zasobów do systemu tak szybko, jak to możliwe, to zaś przekłada się na płynne działanie MySQL i eliminuje wszelkie wątpliwości co do tego, czy PHP sam zwróci zajęta pamięć, zanim będzie ponownie potrzebna.

## Praktyczny przykład

Pora na napisanie pierwszego programu, który za pomocą PHP będzie umieszczał dane w tabeli MySQL i usuwał je stamtąd. Sugeruję, abyś przykład 10.6 zapisał w roboczym folderze z dokumentami WWW pod nazwą `sqltest.php`. Przykład działania programu został zilustrowany na rysunku 10.2.

*Przykład 10.6. Wstawianie i usuwanie danych przy użyciu programu `sqltest.php`*

```
<?php //sqltest.php  
require_once 'login.php';  
$conn = new mysqli($hn, $un, $pw, $db);  
if ($conn->connect_error) die($conn->connect_error);  
if (isset($_POST['delete']) && isset($_POST['isbn']))  
{  
    $isbn = get_post($conn, 'isbn');  
    $query = "DELETE FROM classics WHERE isbn='$isbn'";  
    $result = $conn->query($query);  
    if (!$result) echo "Instrukcja DELETE nie powiodła się: $query<br>" .  
        $conn->error . "<br><br>";  
}  
if (isset($_POST['author']) &&  
    isset($_POST['title']) &&  
    isset($_POST['category']) &&  
    isset($_POST['year']) &&  
    isset($_POST['isbn']))  
{  
    $author = get_post($conn, 'author');
```



Rysunek 10.2. Rezultat działania programu `sqltest.php` z przykładu 10.8

```

$title = get_post($conn, 'title');
$category = get_post($conn, 'category');
$year = get_post($conn, 'year');
$isbn = get_post($conn, 'isbn');
$query = "INSERT INTO classics VALUES" .
    "('$author', '$title', '$category', '$year', '$isbn')";
$result = $conn->query($query);
if (!$result) echo "Instrukcja INSERT nie powiodła się: $query<br>" .
    $conn->error . "<br><br>";
}
echo <<< END
<form action="sqltest.php" method="post"><pre>
    Autor <input type="text" name="author">
    Tytuł <input type="text" name="title">
    Kategoria <input type="text" name="category">
    Rok <input type="text" name="year">
    ISBN <input type="text" name="isbn">
    <input type="submit" value="DODAJ REKORD">
</pre></form>
END;
$query = "SELECT * FROM classics"; $result = $conn->query($query);
if (!$result) die ("Brak dostępu do bazy danych: " . $conn->error);
$rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)
{
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);

```

```

echo <<<_END
<pre>
    Autor $row[0]
    Tytuł $row[1]
    Kategoria $row[2]
    Rok $row[3]
    ISBN $row[4]
</pre>
<form action="sqltest.php" method="post">
<input type="hidden" name="delete" value="yes">
<input type="hidden" name="isbn" value="$row[4]">
<input type="submit" value="USUŃ REKORD"></form>
_END;
}
$result->close();
$conn->close();
function get_post($conn, $var)
{
    return $conn->real_escape_string($_POST[$var]);
}
?>

```



W przykładzie 10.6 został użyty typowy formularz HTML. W rozdziale 11. zapoznasz się ze szczegółowymi informacjami na temat formularzy — tymczasem przyjmij, że uznałem ich znajomość za coś oczywistego, i skup się przede wszystkim na komunikacji z bazą danych.

Składający się z ponad 80 linii program może sprawiać wrażenie skomplikowanego, ale nie obawiaj się — większość kodu poznałeś już w przykładzie 10.5, a jego działanie jest stosunkowo proste.

Najpierw program sprawdza, czy użytkownik podjął jakieś działania, a następnie — zgodnie z nimi — umieszcza w tabeli *classics* w bazie *publications* nowy rekord albo usuwa jeden z istniejących. Niezależnie od akcji podjętej przez użytkownika program wyświetla w przeglądarce wszystkie aktualne wiersze tabeli. Przeanalizujmy działanie programu.

Pierwsza część nowego kodu rozpoczyna się od wywołania funkcji `isset`, która sprawdza, czy wypełnione zostały wszystkie pola umożliwiające dodanie nowego rekordu lub usunięcie go. Jeśli tak, to w każdym z wierszy w ramach instrukcji `if` jest wywoływana funkcja `get_post`, która została zdefiniowana na końcu programu. Funkcja ta pełni prostą, ale bardzo ważną funkcję: zbiera i przesyła dane z przeglądarki.

## Tablica `$_POST`

W jednym z poprzednich rozdziałów wspomniałem, że dane wprowadzone przez użytkownika w przeglądarce są wysyłane za pośrednictwem metody GET albo POST. Na ogół lepsza jest metoda POST, której tutaj użyliśmy (ponieważ pozwala uniknąć wyświetlania niepożądanych danych na pasku adresu przeglądarki). Serwer WWW gromadzi wszystkie dane podane przez użytkownika (nawet jeśli mowa o formularzu składającym się ze stu pól) i umieszcza je w tablicy o nazwie `$_POST`.

Tablica `$_POST` ma charakter asocjacyjny — z tego rodzaju tablicami zetknąłeś się już w rozdziale 6. W zależności od tego, czy formularz został skonfigurowany z użyciem metody POST,

czy GET, dane z niego trafiają do tablicy asocjacyjnej \$\_POST albo \$\_GET. Informacje zawarte w obydwu mogą być odczytywane w identyczny sposób.

Każdemu polu formularza jest przypisywany element tablicy o nazwie zgodnej z nazwą tego pola. To oznacza, że jeśli w formularzu znajduje się pole o nazwie isbn, w tablicy \$\_POST pojawi się element, którego klucz będzie nosił nazwę isbn. Program w PHP może odczytać zawartość takiego pola poprzez odwołanie w postaci \$\_POST['isbn'] albo \$\_POST["isbn"] (w tym przypadku pojedynczy i podwójny cudzysłów będzie miał ten sam efekt).

Jeśli składnia \$\_POST wydaje Ci się skomplikowana, możesz po prostu skorzystać z rozwiązania przedstawionego w przykładzie 10.6: skopiuj dane wprowadzone przez użytkownika do innych zmiennych, a potem możesz zapomnieć o tablicy \$\_POST. To rozwiązanie jest często stosowane w programach PHP — na samym początku pobiera się dane z wszystkich pól tablicy \$\_POST, a potem ją ignoruje.



Nie ma powodu, by modyfikować elementy tablicy \$\_POST. Jej rola ogranicza się do wymiany informacji między przeglądarką a programem i z tego względu lepiej najpierw przenieść zawarte w niej dane do własnych zmiennych.

Wróćmy do wspomnianej funkcji get\_post: zauważ, że każda informacja jest najpierw poddawana działaniu funkcji real\_escape\_string w celu usunięcia dowolnych znaków, które haker mógłby wykorzystać do włamania się do bazy lub zmodyfikowania jej zawartości.

```
function get_post($conn, $var)
{
    return $conn->real_escape_string($_POST[$var]);
}
```

## Usuwanie rekordu

Przed sprawdzeniem, czy użytkownik wpisał nowe dane do wprowadzenia do bazy, program weryfikuje wartość zmiennej \$\_POST['delete']. Jeśli taka wartość istnieje, to znaczy, że użytkownik kliknął przycisk *USUŃ REKORD*, aby usunąć jeden z istniejących rekordów z bazy. W takim przypadku przekazywana jest ponadto wartość zmiennej \$isbn.

Jak zapewne pamiętasz, numer ISBN jednoznacznie identyfikuje każdy rekord. Formularz HTML uwzględni ten numer w zawartym w zmiennej \$query zapytaniu DELETE FROM, które jest następnie przekazywane do metody query obiektu \$conn, skąd trafia do MySQL.

Jeśli zmienna \$\_POST['delete'] nie ma wartości (co oznacza, że nie trzeba usuwać danych z bazy), sprawdzana jest zawartość zmiennej \$\_POST['author'] i pozostałych. Jeśli wszystkim została przypisana wartość, to do zmiennej \$query trafia zapytanie w postaci INSERT INTO z pięcioma danymi do umieszczenia w bazie. Zmienna ta jest następnie przekazywana do metody query, która może zwrócić wartość TRUE albo FALSE. W przypadku wartości FALSE generowany jest komunikat błędu, który trafia do właściwości error obiektu \$conn i może zostać wyświetlony na przykład tak:

```
if (!$result) echo "Instrukcja INSERT nie powiodła się: $query<br>" .
    $conn->error . "<br><br>";
```

## Wyświetlanie formularza

Następna część kodu odpowiada za wyświetlenie niewielkiego formularza, widocznego w górnej części rysunku 10.2. Zapewne pamiętasz z poprzednich rozdziałów konstrukcję echo <<<\_END...\_END, która powoduje wyświetlenie w przeglądarce wszystkiego, co znajduje się między znacznikami \_END.



Zamiast używać instrukcji echo, można byłoby przerwać program PHP za pomocą znacznika `?`, umieścić w pliku niezbędny fragment dokumentu HTML, a potem wstawić znacznik `<?`, aby wrócić do kodu PHP. Obrona metoda zależy tylko od preferencji programisty, ja jednak zawsze zalecam zrealizowanie całej operacji w kodzie PHP. Powody ku temu mam następujące:

- Umieszczanie tylko kodu PHP w plikach `.php` jest korzystne podczas debugowania (także przez innych programistów). Nie ma wówczas potrzeby szukać wydzielonych fragmentów HTML.
- Jeśli chciałbyś wyświetlić wartość zmiennej PHP w obrębie kodu HTML, możesz po prostu wpisać jej nazwę. Jeśli zamknąłbyś blok kodu PHP, musiałbyś najpierw zainicjować nowy blok, wyświetlić zmienną, a potem wrócić do HTML.

Fragment kodu HTML z formularzem po prostu przekierowuje jego działanie na plik `sqltest.php`. To oznacza, że po wysłaniu formularza zawartość jego pól trafia z powrotem do pliku `sqltest.php`, czyli naszego programu. Formularz został ponadto skonfigurowany tak, by pola były wysyłane metodą POST, a nie GET. Ten wybór jest podyktowany faktem, że w przypadku metody GET dane są dołączane do przesyłanego adresu URL, co może powodować mały bałagan w pasku adresu przeglądarki. Poza tym metoda GET umożliwia łatwą ingerencję w przesyłane dane, a tym samym pozwala na podejmowanie prób włamania się na serwer. Z tego względu zawsze gdy tylko jest to możliwe, lepiej używać metody POST, która ma zarazem tę zaletę, że ukrywa wysyłane dane.

Poniżej pól formularza HTML znajduje się przycisk wysyłania danych z napisem *DODAJ REKORD*. Zwróć uwagę na zastosowanie znaczników `<pre>` i `</pre>`, które zostały użyte w celu wymuszenia fontu o stałej szerokości znaków, co pozwala na eleganckie wyrównanie poszczególnych wpisów w formularzu. Ponadto w obrębie znaczników `<pre>` są uwzględniane znaki końca linii.

## Wysyłanie zapytań do bazy danych

Następnie kod wraca do znajomego „terytorium” z przykładu 10.5 — do bazy MySQL jest wysyłane zapytanie o wszystkie rekordy z tabeli `classics`:

```
$query = "SELECT * FROM classics";  
$result = $conn->query($query);
```

W dalszej kolejności zmiennej `$rows` jest przypisywana wartość odzwierciedlająca liczbę wierszy w tabeli:

```
$rows = $result->num_rows;
```

Na podstawie wartości zmiennej `$rows` jest inicjalizowana pętla `for`, która służy do wyświetlania zawartości poszczególnych wierszy. W każdej iteracji tej pętli wywoływana jest metoda `data_seek` obiektu `$result`, która odczytuje porcję interesujących nas danych:

```
$result->data_seek($j);
```

Następnie tablica `$row` jest wypełniana poszczególnymi wierszami danych przy użyciu metody `fetch_array` obiektu `$result`. Do metody jest przekazywana stała `MYSQLI_NUM`, która wymusza zwrócenie tablicy numerycznej (a nie asocjacyjnej):

```
$row = $result->fetch_array(MYSQLI_NUM);
```

Po umieszczeniu danych w tablicy `$row` ich wyświetlenie jest już proste. Użyłem w tym celu konstrukcji `heredoc` z instrukcjami `echo` i z zastosowaniem znaczników `<pre>`, które umożliwiają estetyczne wyrównanie poszczególnych wierszy.

Po każdym wyświetlonym rekordzie mamy do czynienia z kolejnym formularzem, który również przekierowuje użytkownika do pliku `sqltest.php` (czyli do tego samego programu). Ten formularz zawiera dwa ukryte pola: `delete` oraz `isbn`. Pole `delete` ma wartość `yes`, zaś wartość pola `isbn` jest określana na podstawie zawartości pola tablicy `$row[4]`, w którym jest przechowywany numer ISBN dla danego rekordu.

Formularz kończy przycisk wysyłania danych z napisem `USUŃ REKORD`. Następnie nawias klamrowy zamyka pętlę, która będzie powtarzana aż do wyświetlenia wszystkich rekordów. Gdy tak się stanie, wywoływane są metody `close` obiektów `$result` oraz `$conn` w celu zamknięcia połączenia i zwolnienia zasobów dla PHP.

```
$result->close();  
$conn->close();
```

Wreszcie na końcu kodu znajduje się definicja funkcji `get_post`, której już wcześniej się przyglądaliśmy. I to już koniec struktury naszego pierwszego programu PHP umożliwiającego komunikację z bazą danych. Przekonajmy się, co on potrafi.

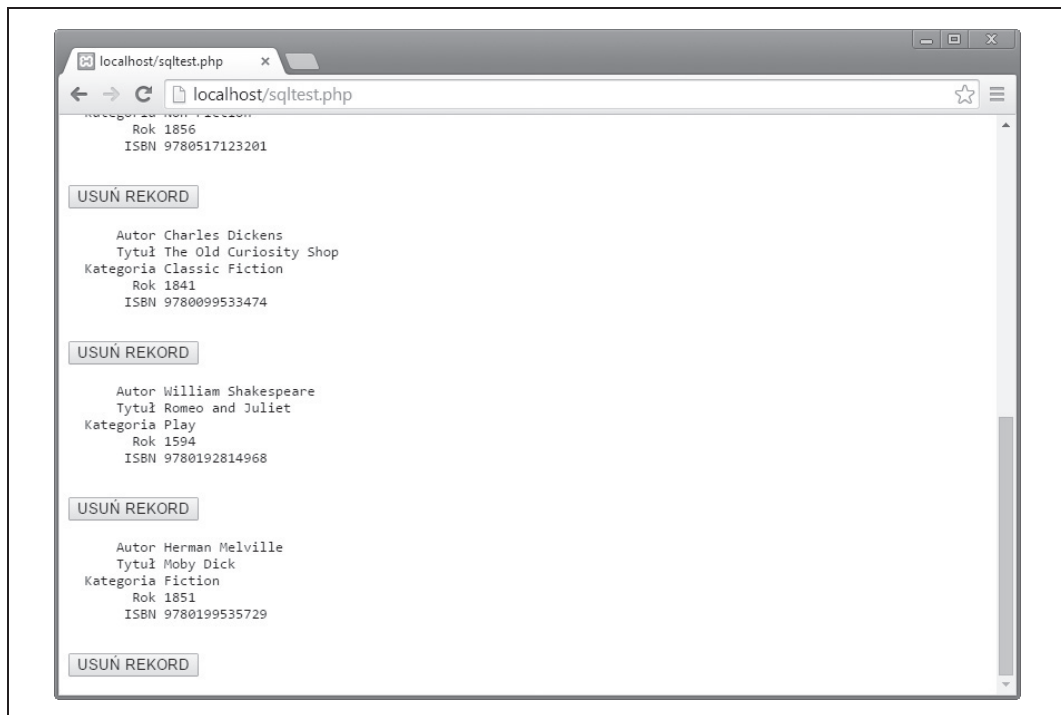
Po wprowadzeniu kodu (i poprawieniu ewentualnych pomyłek) spróbuj wprowadzić następujące informacje do kolejnych pól formularza, aby dodać do bazy nowy rekord poświęcony książce *Moby Dick*:

```
Herman Melville  
Moby Dick  
Fiction  
1851  
9780199535729
```

## Działanie programu

Po wysłaniu powyższych danych do bazy za pomocą przycisku `DODAJ REKORD` przewiń zawartość strony w dół, aby wyświetlić nowy rekord. Efekt powinien wyglądać podobnie jak na rysunku 10.3.

Przekonajmy się teraz, jak działa usuwanie danych z bazy, na podstawie przykładowego rekordu, który wpisujemy tylko w tym celu. Wprowadź cyfrę 1 w każdym z pięciu pól z informacjami o nowej książce i kliknij przycisk `DODAJ REKORD`. Jeśli teraz przewiniesz zawartość okna w dół, zobaczysz nowy wpis składający się z samych jedynek. Oczywiście taki rekord nie jest nam w tabeli do niczego potrzebny, kliknij więc znajdujący się pod nim przycisk `USUŃ`



Rysunek 10.3. Po dodaniu do bazy książki „Moby Dick”

REKORD i ponownie przewiń zawartość okna w dół, aby się przekonać, że rekord rzeczywiście został usunięty.



Przy założeniu, że wszystko przebiegło zgodnie z planem, za pomocą opisanego programu możesz dodawać i usuwać rekordy z bazy. Wykonaj kilka tego typu operacji, ale zostaw w tabeli dotychczasowe informacje o książkach (w tym rekord z książką *Moby Dick*), bo będziemy ich jeszcze potrzebować. Spróbuj dwukrotnie dodać rekord zawierający same jedyńki — przy drugiej próbie pojawi się komunikat błędu wynikający z faktu, że w bazie istnieje już książka o ISBN wynoszącym 1.

## MySQL w praktyce

Masz już wszystkie wiadomości niezbędne do zapoznania się z kilkoma praktycznymi metodami komunikacji z bazą danych za pośrednictwem PHP, umożliwiającymi między innymi tworzenie i usuwanie tabel, wstawianie, aktualizowanie i kasowanie rekordów oraz zabezpieczanie bazy danych oraz strony internetowej przed złośliwymi atakami. W kolejnych przykładach przyjąłem, że dysponujesz plikiem *login.php*, o którym wspominałem wcześniej w tym rozdziale.

## Tworzenie tabeli

Przypuśćmy, że pracujesz w ogrodzie zoologicznym i chciałbyś utworzyć bazę danych z informacjami o wszystkich gatunkach kotowatych, które w nim zamieszkują. Wiesz, że istnieje

dziewięć *rodzin* kotów — lwy, tygrysy, jaguary, pantery, pумы, gepardy, rysie, karakale i koty domowe. Aby je ująć w bazie, potrzebna będzie osobna kolumna. Każdy kot ma jakieś *imię*, które powinno trafić do kolejnej kolumny; powinieneś też zebrać informacje o ich *wieku* — czyli potrzebna będzie jeszcze jedna. Oczywiście na dalszym etapie rozwoju bazy niezbędne okazałyby się jeszcze inne kolumny, na przykład z informacjami dotyczącymi wymogów żywieniowych, szczepień itp., ale na razie nie są nam potrzebne. Ponadto każde zwierzę będzie wymagało unikatowego identyfikatora, to zaś oznacza, że trzeba będzie utworzyć jeszcze jedną kolumnę o nazwie *id*.

Przykład 10.7 przedstawia kod tabeli MySQL, w której można przechowywać tego rodzaju dane. Główne zapytanie zostało wyróżnione pogrubieniem.

*Przykład 10.7. Tworzenie tabeli o nazwie koty*

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);

$query = "CREATE TABLE koty (
    id SMALLINT NOT NULL AUTO_INCREMENT,
    rodzina VARCHAR(32) NOT NULL,
    imie VARCHAR(32) NOT NULL,
    wiek TINYINT NOT NULL,
    PRIMARY KEY (id)
)";

$result = $conn->query($query);
if (!$result) die ("Brak dostępu do bazy danych: " . $conn->error);
?>
```

Jak widać, zapytanie do bazy MySQL w PHP wygląda bardzo podobnie jak wówczas, gdy wpisuje się je z poziomu wiersza poleceń; różnica polega przede wszystkim na braku kończącego je średnika — ten znak jest bowiem zbędny w przypadku wysyłania zapytań za pośrednictwem PHP.

## Wyświetlanie informacji o tabeli

Oto przydatny program, który umożliwia weryfikację poprawności tabeli utworzonej za pomocą przeglądarki bez uciekania się do wiersza poleceń MySQL. Jego działanie opiera się na instrukcji `DESCRIBE koty` i polega na wyświetleniu tabeli HTML z czterema nagłówkami — *Kolumna*, *Typ*, *Null* oraz *Klucz*. Pod nagłówkami są wymienione wszystkie kolumny tabeli. Aby użyć tego programu w odniesieniu do innej tabeli, wystarczy zastąpić nazwę *koty* w zapytaniu nazwą żądanej tabeli (przykład 10.8).

*Przykład 10.8. Wyświetlanie informacji o tabeli koty*

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "DESCRIBE koty";
$result = $conn->query($query);
if (!$result) die ("Brak dostępu do bazy danych: " . $conn->error);
$rows = $result->num_rows;
```



```

echo "<table><tr><th>Kolumna</th><th>Typ</th><th>Null</th><th>Klucz</th></tr>";
for ($j = 0 ; $j < $rows ; ++$j)
{
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);
    echo "<tr>";
    for ($k = 0 ; $k < 4 ; ++$k) echo "<td>$row[$k]</td>";
    echo "</tr>";
}
echo "</table>";
?>

```

Efekt działania powyższego programu powinien wyglądać tak:

Kolumna	Typ	Null	Klucz
id	smallint(6)	NO	PRI
rodzina	varchar(32)	NO	
imie	varchar(32)	NO	
wiek	tinyint(4)	NO	

## Usuwanie tabeli

Usuwanie tabeli jest bardzo proste, a co za tym idzie — bardzo niebezpieczne, trzeba więc zachować ostrożność. Przykład 10.9 przedstawia kod, który to umożliwia. Proponuję jednak, żebyś chwilę poczekał z jego wypróbowaniem, aż zapoznasz się z kolejnymi przykładami — uruchomienie tego programu spowoduje bowiem usunięcie tabeli *koty* i trzeba będzie ją ponownie utworzyć za pomocą kodu z przykładu 10.7.

*Przykład 10.9. Usuwanie tabeli koty*

```

<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "DROP TABLE koty";
$result = $conn->query($query);
if (!$result) die("Brak dostępu do bazy danych: " . $conn->error);
?>

```

## Dodawanie danych

Dodajmy trochę danych do naszej tabeli, korzystając z kodu przykładu 10.10.

*Przykład 10.10. Dodawanie danych do tabeli koty*

```

<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "INSERT INTO koty VALUES(NULL, 'Lew', 'Leon', 4)";
$result = $conn->query($query);
if (!$result) die("Brak dostępu do bazy danych: " . $conn->error);
?>

```

W analogiczny sposób można dodać informacje o dwóch innych kotach. Aby to zrobić, zmień zawartość zmiennej `$query` zgodnie z poniższymi propozycjami i za każdym razem ponownie uruchom program w przeglądarce:

```
$query = "INSERT INTO koty VALUES(NULL, 'Puma', 'Pazur', 2)";
$query = "INSERT INTO koty VALUES(NULL, 'Gepard', 'Splinter', 3)";
```

Zwróciłeś uwagę na parametr NULL przekazywany na początku listy danych? Jego obecność wynika z faktu, że kolumna *id* jest typu AUTO\_INCREMENT, więc to MySQL decyduje, jaką wartość w niej wpisać, zgodnie z kolejnym dostępnym numerem. Jeśli prześlemy w zapytaniu wartość NULL, parametr ten zostanie zignorowany, co załatwi sprawę.

Oczywiście najskuteczniejszym sposobem na umieszczenie danych w bazie MySQL jest utworzenie tablicy i wstawienie danych w postaci jednego zapytania.

## Odczytywanie danych

Ponieważ w tabeli *koty* powinny już być jakieś dane, za pomocą kodu z przykładu 10.11 możemy sprawdzić, czy zostały one poprawnie wprowadzone.

*Przykład 10.11. Odczytywanie wierszy z tabeli koty*

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "SELECT * FROM koty";
$result = $conn->query($query);
if (!$result) die ("Brak dostępu do bazy danych: " . $conn->error);
$rows = $result->num_rows;
echo "<table><tr><th>Id</th><th>Rodzina</th><th>Imię</th><th>Wiek</th></tr>";
for ($j = 0 ; $j < $rows ; ++$j)
{
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);
    echo "<tr>";
    for ($k = 0 ; $k < 4 ; ++$k) echo "<td>$row[$k]</td>";
    echo "</tr>";
}
echo "</table>";
?>
```

Ten kod powoduje po prostu wysłanie zapytania MySQL w postaci SELECT \* FROM koty i wyświetlenia wszystkie zwrócone przez nie wiersze. Rezultat jest następujący:

Id	Rodzina	Imię	Wiek
1	Lew	Leon	4
2	Puma	Pazur	2
3	Gepard	Splinter	3

Jak widać, wartość w kolumnie *id* była automatycznie zwiększana.

## Aktualizowanie danych

Zmiana danych znajdujących się w tabeli również jest dość prosta. Być może zwróciłeś uwagę na pisownię imienia geparda? Zamiast Splinter powinno być Sprinter. Wprowadźmy więc odpowiednią poprawkę, jak w przykładzie 10.12.

Przykład 10.12. Zmiana imienia geparda ze Splinter na Sprinter

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "UPDATE koty SET imie='Sprinter' WHERE imie='Splinter'";
$result = $conn->query($query);
if (!$result) die ("Brak dostępu do bazy danych: " . $conn->error);
?>
```

Jeśli teraz ponownie uruchomiłbyś przykład 10.11, to okazałoby się, że rezultat jest następujący:

Id	Rodzina	Imię	Wiek
1	Lew	Leon	4
2	Puma	Pazur	2
3	Gepard	Sprinter	3

## Usuwanie danych

Puma Pazur została przeniesiona do innego zoo, powinniśmy więc usunąć ją z bazy danych (przykład 10.13).

Przykład 10.13. Usuwanie pumy o imieniu Pazur z tabeli koty

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "DELETE FROM koty WHERE imie='Pazur'";
$result = $conn->query($query);
if (!$result) die ("Brak dostępu do bazy danych: " . $conn->error);
?>
```

Ten przykład bazuje na standardowym zapytaniu DELETE FROM, a gdy po jego wykonaniu raz jeszcze uruchomisz program z przykładu 10.11, okaże się, że jeden wiersz tabeli rzeczywiście został usunięty:

Id	Rodzina	Imię	Wiek
1	Lew	Leon	4
3	Gepard	Sprinter	3

## Zastosowanie opcji AUTO\_INCREMENT

W przypadku opcji AUTO\_INCREMENT nie wiadomo, jaka konkretnie wartość została umieszczona w kolumnie po dodaniu do tabeli nowego wiersza. Jeśli chcesz się dowiedzieć, musisz zapytać o to MySQL za pomocą funkcji `mysql_insert_id` albo użyć właściwości `insert_id` obiektu połączenia. Takie sytuacje jak powyższa zdarzają się często, na przykład przy przetwarzaniu zamówienia, po dodaniu nowego klienta do tabeli *Klienci*; można potem odwołać się do nowo utworzonego identyfikatora *IdKlienta* podczas umieszczania danych o jego zakupach w tabeli *Zakupy*.

Przykład 10.14 opiera się na zmodyfikowanym kodzie przykładu 10.10. Zmiany powodują wyświetlenie omawianej wartości po każdym wstawieniu nowego wiersza.

Przykład 10.14. Dodawanie danych do tabeli *koty* i wyświetlanie identyfikatora nowego wiersza

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$query = "INSERT INTO koty VALUES(NULL, 'Ryś', 'Rysiek', 5)";
$result = $conn->query($query);
if (!$result) die("Brak dostępu do bazy danych: " . $conn->error);
echo "Identyfikator nowego wiersza to: " . $conn->insert_id;
?>
```

Zawartość tabeli powinna teraz wyglądać tak jak poniżej (zwróć uwagę, że użyty wcześniej identyfikator 2 *nie został* ponownie wykorzystany, gdyż w pewnych sytuacjach mogłoby to prowadzić do komplikacji):

Id	Rodzina	Imię	Wiek
1	Lew	Leon	4
3	Gepard	Sprinter	3
4	Ryś	Rysiek	5

## Zastosowanie identyfikatorów wstawionych wierszy

Bardzo często zdarza się, że dane trzeba wstawić w kilku tabelach: po dodaniu książki trzeba wprowadzić informacje o jej autorze, po utworzeniu konta nowego klienta należy dodać dane o jego zakupie itp. W takich przypadkach, jeśli użyta została kolumna z automatyczną inkrementacją, należy zachować identyfikator wstawionego wiersza, aby posłużyć się nim w innej, powiązanej tabeli.

Przypuśćmy na przykład, że koty z naszego zoo mogą być „wirtualnie adoptowane” przez osoby spoza ogrodu, które będą organizowały zbiórki wspomagające ich utrzymanie. W takim przypadku po umieszczeniu nowego kota w tabeli *koty* powinniśmy utworzyć klucz umożliwiający powiązanie zwierzęcia z jego opiekunem. Kod umożliwiający przeprowadzenie tej operacji jest podobny do kodu z przykładu 10.14. Różnicą jest to, że zwrócony identyfikator nowego wiersza trafia do zmiennej o nazwie `$insertID`, która jest następnie wykorzystywana jako element kolejnego zapytania:

```
$query = "INSERT INTO koty VALUES(NULL, 'Ryś', 'Rysiek', 5)";
$result = $conn->query($query);
$insertID = $conn->insert_id;

$query = "INSERT INTO sponsorzy VALUES($insertID, 'Anna', 'Kowalska')";
$result = $conn->query($query);
```

W rezultacie kot jest powiązany ze swoim opiekunem poprzez unikatowy identyfikator, utworzony automatycznie przy użyciu opcji `AUTO_INCREMENT`.

## Zastosowanie blokad

Bezpieczna procedura łączenia tabel za pośrednictwem identyfikatora wstawionego wiersza polega na zastosowaniu blokad (albo transakcji, zgodnie ze wskazówkami podanymi w rozdziale 9.). Takie rozwiązanie może wprawdzie wydłużyć czas operacji w przypadku dużej liczby użytkowników zapisujących dane w tej samej tabeli, ale na ogół warto się z tym pogodzić. Procedura wygląda następująco:

1. Zablokowanie pierwszej tabeli (np. *koty*).
2. Wstawienie danych do pierwszej tabeli.
3. Pobranie unikatowego identyfikatora nowego wiersza z pierwszej tabeli przy użyciu właściwości `insert_id`.
4. Odblokowanie pierwszej tabeli.
5. Wstawienie danych do drugiej tabeli.

Blokadę można bezpiecznie zdjąć przed wpisaniem danych do drugiej tabeli, gdyż identyfikator został już pobrany i zapisany w zmiennej programu. Zamiast blokowania tabel można się posłużyć transakcjami, ale to rozwiązanie jeszcze bardziej spowalnia działanie serwera MySQL.

## Wykonywanie zapytań pomocniczych

Na razie wystarczy kocich spraw. Aby przeanalizować trochę bardziej skomplikowane zapytania, musimy bowiem wrócić do tabel *customers* oraz *classics* utworzonych w rozdziale 8. W tabeli *customers* znajdują się informacje o klientach, tabela *classics* zawiera informacje o kilku książkach. Obie tabele mają wspólną kolumnę z numerami ISBN, o nazwie *isbn*, której możemy użyć do utworzenia dodatkowych zapytań.

Na przykład w celu wyświetlenia wszystkich klientów wraz z tytułami i autorami kupionych przez nich książek można użyć kodu podanego w przykładzie 10.15.

Przykład 10.15. Wykonywanie zapytań pomocniczych

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);

$query = "SELECT * FROM customers";
$result = $conn->query($query);
if (!$result) die ("Brak dostępu do bazy danych: " . $conn->error);

$rows = $result->num_rows;

for ($j = 0 ; $j < $rows ; ++$j)
{
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);
    echo "$row[0] zakupił(a) ISBN $row[1]:<br>";
    $subquery = "SELECT * FROM classics WHERE isbn='$row[1]'";
    $subresult = $conn->query($subquery);
    if (!$subresult) die ("Brak dostępu do bazy danych: " . $conn->error);
    $subrow = $subresult->fetch_array(MYSQLI_NUM);
    echo " '$subrow[1]' autora $subrow[0]<br>";
}
?>
```

Ten program wykonuje zapytanie pomocnicze do tabeli *customers* i wyszukuje wszystkich klientów, a następnie na podstawie numeru ISBN zakupionych przez nich książek wykonuje kolejne zapytanie do tabeli *classics*, aby odszukać tytuły i autorów tych pozycji. Rezultat działania powyższego kodu jest następujący:

Joe Bloggs zakupił(a) ISBN 9780099533474:  
'The Old Curiosity Shop' autora Charles Dickens  
Mary Smith zakupił(a) ISBN 9780582506206:  
'Pride and Prejudice' autora Jane Austen  
Jack Wilson zakupił(a) ISBN 9780517123201:  
'The Origin of Species' autora Charles Darwin



W tym konkretnym przypadku — choć nie byłaby to ilustracja możliwości zapytań pomocniczych — można byłoby uzyskać te same informacje za pomocą zapytania typu NATURAL JOIN (rozdział 8.), na przykład takiego:

```
SELECT name,isbn,title,author FROM customers  
NATURAL JOIN classics;
```

## Zapobieganie próbom ataków

Brak weryfikacji danych wprowadzanych przez użytkowników do bazy MySQL niesie ze sobą duże ryzyko, z którego nie zawsze zdajemy sobie sprawę. Przypuśćmy, że mamy następujący prosty kawałek kodu służący do weryfikacji użytkowników:

```
$user = $_POST['user'];  
$pass = $_POST['pass'];  
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
```

Na pierwszy rzut oka taki kod nie budzi najmniejszych zastrzeżeń. Jeśli na przykład użytkownik wprowadzi wartości jankowalski oraz haslo123, które trafią do zmiennych \$user i \$pass, to całe zapytanie przekazane do bazy MySQL będzie wyglądało następująco:

```
SELECT * FROM users WHERE user='jankowalski' AND pass='haslo123'
```

Tu akurat rzeczywiście nic złego się nie stanie, ale co, jeśli do zmiennej \$user trafiłby następujący łańcuch znaków (a zmienna \$pass pozostałaby pusta)?

```
admin' #
```

Przyjrzyjmy się zapytaniu, które w tej sytuacji zostałyby wysłane do serwera MySQL:

```
SELECT * FROM users WHERE user='admin' #' AND pass=''
```

Dostrzegasz problem? W MySQL symbol # oznacza początek komentarza. W efekcie użytkownik zostałby zalogowany jako *admin* (jeśli takie konto rzeczywiście istnieje) bez konieczności wpisywania hasła. Przyjrzyj się jeszcze raz temu zapytaniu; pogrubieniem została oznaczona tylko ta część, która rzeczywiście zostałaby wykonana; resztę serwer by pominął:

```
SELECT * FROM users WHERE user='admin' #' AND pass=''
```

Można byłoby jednak mówić o dużym szczęściu, gdyby złośliwy użytkownik ograniczył się do czegoś takiego. W tej sytuacji mógłbyś bowiem uruchomić swoją aplikację i naprawić szkody wyrządzone przez kogoś, kto zaloguje się jako *admin*. Ale co by było, gdyby kod programu powodował usunięcie użytkownika z bazy? Kod mógłby wyglądać na przykład tak:

```
$user = $_POST['user'];  
$pass = $_POST['pass'];  
$query = "DELETE FROM users WHERE user='$user' AND pass='$pass'";
```

I ponownie: taki kod nie wydaje się podejrzany, ale co by się stało, gdyby ktoś wpisał do zmiennej \$user poniższy ciąg znaków?

```
cokolwiek' OR 1=1 $
```

Do serwera MySQL trafiłoby następujące zapytanie:

```
DELETE FROM users WHERE user='cokołwiek' OR 1=1 # AND pass=''
```

Ups... to zapytanie SQL zawsze będzie zwracało wartość TRUE, a w rezultacie cała baza użytkowników zostałaby usunięta! W jaki sposób zabezpieczyć się przed tego typu atakami?

## Działania prewencyjne

Przed wszystkim nie należy polegać na wbudowanym mechanizmie PHP o nazwie *magic quotes* (dosł. magiczne cudzysłowy), który automatycznie poprzedza pojedyncze i podwójne cudzysłowy modyfikatorem —lewym ukośnikiem (\). Dlaczego? Ponieważ ten mechanizm można wyłączyć; wielu programistów postępuje w ten sposób, aby zastąpić go własnymi zabezpieczeniami. Nie ma więc żadnej gwarancji, że właśnie tak się nie stało na serwerze, na którym pracujesz. Co więcej, w PHP 5.3.0 mechanizm ten został uznany za przestarzały, a z PHP 6.0.0 już go usunięto.

Zamiast tego we wszystkich odwołaniach do MySQL powinieneś używać funkcji `real_escape_string`. Przykład 10.16 ilustruje działanie tej funkcji, która „oczyszcza” łańcuch znaków wprowadzony przez użytkownika z modyfikatorów i niepożądanych znaków.

*Przykład 10.16. Prawidłowe „oczyszczanie” danych użytkownika przed wystąpieniem do serwera MySQL*

```
<?php
function mysql_fix_string($conn, $string)
{
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
}
?>
```

Funkcja `get_magic_quotes_gpc` zwraca wartość TRUE, jeśli mechanizm *magic quotes* jest aktywny. W takim przypadku ukośniki dodane do łańcucha znaków zostają usunięte, bo w przeciwnym razie metoda `real_escape_string` spowoduje dodanie kolejnych modyfikatorów i doprowadzi do przekłamań w łańcuchu znaków. W przykładzie 10.17 została zdefiniowana funkcja `mysql_fix_string`, wykorzystująca opisane rozwiązanie. Funkcji tej można bez przeszkód użyć w kodzie innych, własnych programów.

*Przykład 10.17. Bezpieczny dostęp do MySQL z użyciem danych użytkownika*

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);
$user = mysql_fix_string($conn, $_POST['user']);
$pass = mysql_fix_string($conn, $_POST['pass']);
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";

// itd.

function mysql_fix_string($conn, $string)
{
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
}
?>
```



Rola opisanych zabezpieczeń ostatnio trochę zmalała, a to ze względu na znacznie łatwiejszą i bezpieczniejszą metodę dostępu do MySQL, która pozwala uniknąć ich stosowania. Mam na myśli używanie elementów zastępczych, o których przeczytasz za chwilę.

## Zastosowanie elementów zastępczych

Przygotowanie wyrażeń z elementami zastępczymi umożliwia przesłanie do bazy „czystych” danych, dzięki czemu można uniknąć niepożądanego interpretacji danych przesłanych przez użytkownika (lub innych) jako instrukcji MySQL (co stanowi potencjalną furtkę do włamania).

Trik polega na uprzednim zdefiniowaniu wyrażenia, które ma być wykonane przez MySQL, i zastąpieniu wszystkich jego elementów, które będą zawierały dane, zwykłymi znakami zapytania.

W czystym kodzie MySQL tak spreparowane wyrażenie wygląda podobnie jak w przykładzie 10.18.

*Przykład 10.18. Elementy zastępcze w MySQL*

```
PREPARE statement FROM "INSERT INTO classics VALUES(?,?,?,?)";

SET @author = "Emily Brontë",
    @title = "Wuthering Heights",
    @category = "Classic Fiction",
    @year = "1847",
    @isbn = "9780553212587";

EXECUTE statement USING @author,@title,@category,@year,@isbn;
DEALLOCATE PREPARE statement;
```

Obsługa tego rodzaju wyrażeń jest dość kłopotliwa, ale na szczęście w sukurs przychodzi nam rozszerzenie myśli, a konkretnie jego metoda o nazwie `prepare`, którą wywołuje się następująco:

```
$stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?)');
```

Obiekt o nazwie `$stmt` (lub dowolnej innej, jaką wybierzesz) zwracany przez tę metodę jest następnie używany do przesyłania na serwer danych w miejsce znaków zapytania. Najpierw jednak należy powiązać zmienne PHP z poszczególnymi znakami zapytania (czyli elementami zastępczymi) w kolejności ich wymienienia:

```
$stmt->bind_param('sssss', $author, $title, $category, $year, $isbn);
```

Pierwszy argument metody `bind_param` to ciąg znaków odzwierciedlający typy kolejnych argumentów. W tym przypadku składa się z pięciu znaków `s`, które odpowiadają łańcuchom tekstowym, ale można w nim podać dowolną kombinację typów według następujących zasad:

- `i` — liczba całkowita,
- `d` — liczba zmiennoprzecinkowa podwójnej precyzji,
- `s` — łańcuch znaków,
- `b` — obiekt binarny (tzw. *blob*, zostanie przesłany w pakietach).

Po powiązaniu zmiennych z elementami gotowego wyrażenia należy zapełnić te zmienne danymi, które zostaną przekazane do MySQL:



```

$author = 'Emily Brontë';
$title  = 'Wuthering Heights';
$category = 'Classic Fiction';
$year   = '1847';
$isbn   = '9780553212587';

```

Na tym etapie PHP dysponuje wszystkimi informacjami niezbędnymi do wykonania przygotowanego wyrażenia. Wobec tego możemy użyć poniższej instrukcji, która odwołuje się do metody `execute` obiektu `$stmt` utworzonego wcześniej:

```
$stmt->execute();
```

Zanim przystąpimy do dalszych operacji, sprawdźmy, czy instrukcja została wykonana poprawnie. Można tego dokonać przez zweryfikowanie właściwości `affected_rows` wyrażenia `$stmt`:

```
printf("%d Wiersz wstawiony.\n", $stmt->affected_rows);
```

W przypadku przykładu podanego niżej instrukcja ta powinna poinformować o pomyślnym wstawieniu jednego wiersza.

Po udanym wykonaniu instrukcji (albo rozprawieniu się z ewentualnymi błędami) można zamknąć obiekt `$stmt`:

```
$stmt->close();
```

Na koniec można też zamknąć obiekt `$conn` (przy założeniu, że nie będziesz go już używał):

```
$conn->close();
```

Owoce wszystkich wykonanych działań jest przykład 10.19:

*Przykład 10.19. Praktyczne zastosowanie przygotowanego wyrażenia*

```

<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die($conn->connect_error);

$stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?)');
$stmt->bind_param('sssss', $author, $title, $category, $year, $isbn);

$author = 'Emily Brontë';
$title  = 'Wuthering Heights';
$category = 'Classic Fiction';
$year   = '1847';
$isbn   = '9780553212587';

$stmt->execute();
printf("%d Wiersz wstawiony.\n", $stmt->affected_rows);
$stmt->close();
$conn->close();
?>

```

Każde użycie predefiniowanych wyrażeń zamiast zwykłych to jedna okazja dla potencjalnych hakerów mniej, warto więc poświęcić trochę czasu na opanowanie techniki ich stosowania.

## Zapobieganie przekazywaniu niepożądanych danych przez HTML

Istnieje jeszcze jedna metoda podstawiania niepożądanych danych, przed którą warto się zabezpieczyć — przy czym tym razem chodzi nie tyle o bezpieczeństwo strony internetowej,

ale o prywatność i ochronę praw jej użytkowników. Mowa o ataku typu *cross-site scripting* (w skrócie XSS).

Do tego rodzaju ataku może dojść wówczas, gdy zezwoli się użytkownikowi na wpisywanie kodu HTML (a częściej JavaScript), który zostanie następnie wyświetlony na stronie. Bardzo często wykorzystuje się w tym celu formularz komentarzy. Atak na ogół polega na podjęciu przez hakera próby skonstruowania takiego kodu, który odczytywałby zawartość ciasteczek użytkowników strony, przechwytywał zapisywane w nich loginy i ewentualnie hasła bądź inne informacje. Co gorsza, w ten sposób można też opracować atak mający na celu pobranie złośliwego oprogramowania (trojana) przez zwykłych użytkowników.

Zapobieganie takim działaniom jest na szczęście proste i polega na zastosowaniu funkcji `htmlentities`, która usuwa z kodu HTML znaki specjalne znaczników i zamienia je na łańcuchy tekstowe, tzw. encje HTML. W tej postaci są one przez przeglądarkę wyświetlane jako zwykłe znaki, ale nie będą traktowane jako elementy kodu. Weźmy na przykład następujący kod HTML:

```
<script src='http://x.com/hack.js'>
</script><script>hack();</script>
```

Załóżmy, że ten kod powoduje uruchomienie programu w JavaScriptcie, który podejmuje złośliwe działania. Jeśli jednak najpierw „przepuścimy” ten kod przez funkcję `htmlentities`, otrzymamy następujący, zupełnie nieszkodliwy łańcuch znaków:

```
&lt;script src='http://x.com/hack.js'&gt;
&lt;/script&gt;&lt;script&gt;hack();&lt;/script&gt;
```

Jeśli zamierzasz uwzględnić w programie możliwość wyświetlania informacji wprowadzonych przez użytkownika — czy to od razu, czy po zapisaniu ich w bazie danych — najpierw powinieneś oczyścić te informacje przy użyciu funkcji `htmlentities`. Najwygodniej będzie napisać w tym celu własną funkcję, taką jak w przykładzie 10.20, która oczyszcza kod SQL i zapobiega atakom XSS.

*Przykład 10.20. Funkcje zapobiegające przemycaniu złośliwego kodu w zapytaniach SQL i atakom XSS*

```
<?php
function mysql_entities_fix_string($conn, $string)
{
    return htmlentities(mysql_fix_string($conn, $string));
}
function mysql_fix_string($conn, $string)
{
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
}
?>
```

Funkcja `mysql_entities_fix_string` najpierw wywołuje funkcję `mysql_fix_string`, następnie przekazuje rezultat jej działania do funkcji `htmlentities` i wreszcie zwraca w pełni oczyszczony łańcuch znaków. Aby móc skorzystać z dowolnej z tych funkcji, musisz dysponować obiektem otwartego połączenia z bazą MySQL.

Przykład 10.21 stanowi udoskonaloną („nie do złamania”) wersję przykładu 10.17.

*Przykład 10.21. Sposób na bezpieczny dostęp do MySQL i uniknięcie ataków typu XSS*

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
```

```

if ($conn->connect_error) die($conn->connect_error);
$user = mysql_entities_fix_string($conn, $_POST['user']);
$pass = mysql_entities_fix_string($conn, $_POST['pass']);
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
// itd.
function mysql_entities_fix_string($conn, $string)
{
    return htmlentities(mysql_fix_string($conn, $string));
}
function mysql_fix_string($conn, $string)
{
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
}
?>

```

## Proceduralny wariant zastosowania mysqli

Jeśli wolisz, możesz skorzystać z alternatywnego zestawu funkcji, umożliwiającego obsługę mysqli w postaci proceduralnej (a nie obiektowej).

Zamiast tworzenia obiektu `$connection` w taki sposób:

```
$conn = new mysqli($hn, $un, $pw, $db);
```

możesz użyć następującego kodu:

```
$link = mysqli_connect($hn, $un, $pw, $db);
```

Aby sprawdzić poprawność połączenia i je obsłużyć, możesz wykorzystać następującą instrukcję:

```
if (mysqli_connect_errno()) die(mysqli_connect_error());
```

Z kolei w celu wysłania zapytania do MySQL należy użyć następującego kodu:

```
$result = mysqli_query($link, "SELECT * FROM classics");
```

Po jego wykonaniu rezultat zapytania trafi do zmiennej `$result`. Liczbę zwróconych wierszy można sprawdzić następująco:

```
$rows = mysqli_num_rows($result);
```

Do zmiennej `$rows` trafia całkowita liczba pobranych wierszy. Konkretnie dane możesz odczytać wierszami za pomocą następującej instrukcji, która zwraca tablicę numeryczną:

```
$row = mysqli_fetch_array($result, MYSQLI_NUM);
```

W tym przypadku pole `$row[0]` będzie zawierało pierwszą kolumnę danych, pole `$row[1]` drugą kolumnę i tak dalej. Zgodnie z wcześniejszym opisem poszczególne wiersze danych mogą być zwrócone w postaci tablic asocjacyjnych lub obu typów tablic — w zależności od wartości drugiego argumentu.

Jeśli chciałbyś sprawdzić identyfikator operacji wstawiania, możesz w tym celu użyć funkcji `mysqli_insert_id` w następujący sposób:

```
$insertID = mysqli_insert_id($result);
```

Zastosowanie znaków modyfikujących w przypadku proceduralnej obsługi mysqli jest bardzo proste i można to zrobić na przykład tak:

```
$escaped = mysqli_real_escape_string($link, $val);
```

Przygotowanie zapytania przedstawia się następująco:

```
$stmt = mysqli_prepare($link, 'INSERT INTO classics VALUES(?,?,?,?)');
```

Aby powiązać zmienne z konkretnymi znakami zastępczymi w zapytaniu, należy postąpić tak:

```
mysqli_stmt_bind_param($stmt, 'sssss', $author, $title, $category, $year, $isbn);
```

Z kolei aby wykonać tak przygotowane zapytanie po podstawieniu zmiennych, trzeba użyć następującej instrukcji:

```
mysqli_stmt_execute($stmt);
```

Zakończenie pracy z danym zapytaniem wygląda tak:

```
mysqli_stmt_close($stmt);
```

A tak zamyka się połączenie z bazą MySQL:

```
mysqli_close($link);
```



Szczegółowe informacje na temat stosowania elementów zastępczych (proceduralnie lub obiektowo) znajdziesz pod adresem <http://tinyurl.com/mysqlistmt>. Z kolei omówienie innych aspektów zastosowania rozszerzenia `mysqli` zostało opisane na stronie <http://tinyurl.com/usingmysqli>.

Poznałeś różne sposoby odwoływania się do baz MySQL za pomocą PHP, a to stanowi solidny fundament pod wiedzę, jaką przyswoisz w następnym rozdziale. Przeczytasz w nim bowiem o tworzeniu wygodnych w obsłudze formularzy i wykorzystywaniu danych przekazywanych za ich pośrednictwem.

## Pytania

1. Jak można połączyć się z bazą MySQL za pomocą `mysqli`?
2. Jak wysłać do bazy MySQL zapytanie za pomocą `mysqli`?
3. W jaki sposób uzyskać dostęp do łańcucha tekstowego z komunikatem błędu, jeśli taki wystąpił przy korzystaniu z `mysqli`?
4. W jaki sposób sprawdzić liczbę wierszy zwróconych przez zapytanie `mysqli`?
5. Jak odczytać konkretny wiersz danych spośród całego rezultatu zwróconego przez `mysqli`?
6. Jakiej metody `mysqli` należy użyć, aby oczyścić dane wprowadzone przez użytkownika i w ten sposób uniknąć „przemycenia” niepożądanego kodu?
7. Jakie skutki uboczne może mieć niezamknięcie obiektów utworzonych za pomocą metod `mysqli`?

Odpowiedzi na pytania znajdziesz w dodatku A, w sekcji „Odpowiedzi na pytania z rozdziału 10.”.

## A

adres  
  IP, 586  
  MAC, 586  
Ajax, 29, 32, 371, 514  
aktualizowanie  
  danych, 242  
  plików, 156  
algorytm haszujący  
  md5, 281  
  ripemd, 281  
  sha1, 281  
animacje, 463, 491  
anonimowość, 217  
anulowanie  
  opóźnienia, 461  
  polecenia, 172  
Apache, 34  
aplikacje  
  offline, 594  
  sieciowe, 525  
archiwizacja, 221  
argumenty funkcji setcookie, 277  
asocjacyjność, 86, 316  
asocjacyjność operatorów, 87  
atak  
  session fixation, 290  
  siłowy, 281  
  XSS, 250  
atrybut  
  autocomplete, 269  
  autofocus, 269  
  form, 270  
  height, 270  
  list, 271  
  max, 271  
  min, 271  
  placeholder, 269

  required, 269  
  step, 271  
  width, 270  
atrybuty  
  elementu source, 580  
  nadpisanie, 270  
automatyczna  
  inkrementacja, 181  
  konwersja liczby, 70  
autoryzacja, 280, 283  
autoryzacja HTTP, 278

## B

baza danych MySQL, 29, 167  
  kolumna, 168  
  tabela, 168  
  wiersz, 168  
BBS, Bulletin Board System, 25  
bezpieczeństwo, 78  
bezpieczeństwo sesji, 289  
bezpieczny dostęp do bazy, 250  
biblioteka  
  GD, 34  
  jQuery, 35, 467  
blokada, 244  
blokada pliku, 157  
błąd  
  dzielenia przez zero, 103  
  krytyczny, 229

## C

CGI, Common Gateway Interface, 29  
ciasteczko, 275  
  tworzenie, 276  
  usuwanie, 277  
  włączanie, 291

- cienie, 433, 557
  - CSS, Cascading Style Sheets, 29, 32, 387
    - dziedziczenie kaskadowe, 398
    - fonty i typografia, 405
    - identyfikatory ID, 389
    - importowanie, 388
    - jednostki miar, 403
    - klasy, 389
    - kolory, 408
    - model pudełkowy, 416
    - pseudoklasy, 413
    - reguły, 390
    - rodzaje stylów, 391
    - rozmieszczanie elementów, 411
    - selektory, 393
    - skracanie reguł, 415
    - układ strony, 416
    - zarządzanie, 407
  - CSS3, 32
    - cienie, 433
    - efekty tekstowe, 438
    - fonty Google, 440
    - fonty internetowe, 439
    - kolory, 435
    - obrazy w tle, 428
    - przejścia, 443
    - przekształcenia, 441
    - przekształcenia 3D, 442
    - przezroczystość, 435
    - ramki, 430
    - selektory atrybutów, 423
    - tła, 425
    - układ wielokolumnowy, 434
    - właściwość box-sizing, 425
    - właściwość overflow, 434
  - czas trwania
    - przejścia, 444
    - sesji, 289
- ## D
- debugowanie kodu JavaScript, 297
  - definiowanie
    - funkcji, 109, 331
    - marginesów, 416
    - odstępu, 419
    - ramek, 418
    - reguł, 399
    - wymiarów elementów, 500
  - deklarowanie
    - klasy, 118, 335
    - stałych, 124
    - typu zmiennych, 70
    - właściwości, 123
  - dekrementacja, 303
  - destruktor, 122
  - document.write, 310
  - dodawanie
    - danych, 241
    - elementów, 457, 459, 496
    - indeksów, 187
    - kolumny, 183
    - tekstu O mnie, 617
    - zdjęcia profilowego, 617
    - znajomych, 622
  - dokument, *Patrz także* plik
    - document.write, 310
    - urlget.html, 378
    - urlget.php, 380
    - urlpost.html, 374
    - urlpost.php, 377
    - validate.html, 350, 352
    - XML, 383
    - xmlget.html, 381
    - xmlget.php, 380
  - dołączanie
    - funkcji, 451
    - jQuery, 468
    - plików JavaScript, 296
  - DOM, Document Object Model, 293, 307, 457
    - nawigowanie, 502
  - domena Open Source, 35
  - dopasowanie rozmyte, 356
  - dopasowywanie
    - fragmentów łańcuchów, 423
    - za pomocą metaznaków, 355
  - dopełnianie łańcuchów tekstowych, 148
  - dostarczanie treści, 469
  - dostęp do
    - ciasteczka, 277
    - CSS, 449
    - głównego foldera, 49
    - katalogu root, 47
    - MySQL, 168
    - obiektów, 337
    - tablicy, 131
    - właściwości CSS, 452
    - zdalnego serwera WWW, 50
  - dostosowywanie jQuery, 471
  - druga postać normalna, 209
  - drzewo DOM, 383, 459
  - dynamiczne
    - linkowanie, 105
    - stosowanie klas, 499
    - strony internetowe, 25
  - dynamika przejścia, 444

- dyrektywa
  - GROUP BY, 198
  - include, 114
  - NATURAL JOIN, 200
  - require\_once, 229
- działania prewencyjne, 247
- działanie programu sqltest.php, 238
- dziedziczenie, 126
- dziedziczenie kaskadowe, 398
- dźwięk, 523, 573

## E

- edytor
  - kodu, 51, 53
  - tekstu, 53
- edytowanie profilu użytkownika, 620
- efekty
  - graficzne, 562
  - specjalne, 487
  - tekstowe, 407, 438
- ekran programu phpMyAdmin, 203
- element
  - audio, 575
  - canvas, 35
  - div, 401
  - span, 401
  - video, 578
- elementy
  - nadrzędne, 503
  - poprzedzające, 508
  - potomne, 506
  - siostrzane, 507
  - zastępcze, 248
- etapy procesu Ajax, 35
- etykiety, 263

## F

- file handle, 153
- film, 523, 573
- filtr, 503
- Flash, 577, 581
- font-family, 405
- font-size, 406
- font-style, 406
- font-weight, 407
- fonty, 405
  - Google, 440
  - internetowe, 439
- format CSV, 225
- formaty obrazów, 530
- formatowanie nagłówków, 387

- formularze, 237, 253, 524
  - nieudana weryfikacja, 368
  - oczyszczanie danych, 264
  - opcja register\_globals, 256
  - rodzaje pól, 258
  - wartości domyślne, 257
  - weryfikacja danych, 350, 352, 364
- fragment obrazu, 556
- FTP, 51
- functions.php, 606, 607
- funkcja, 73, 107, 305, 331
  - alert, 311
  - array\_combine, 115
  - C, 451
  - checkdate, 150
  - compact, 142
  - console.log, 310
  - copy, 155
  - count, 139
  - each, 139
  - end, 143
  - explode, 140
  - extract, 141
  - date, 149, 151
  - die, 229
  - document.write, 311
  - escapeshellcmd, 165
  - fgets, 153, 154
  - file\_exists, 152
  - file\_get\_contents, 158, 377
  - flock, 157, 158
  - fopen, 153, 154
  - fseek, 156, 157
  - function\_exists, 115
  - get\_magic\_quotes\_gpc, 247
  - getElementById, 449
  - hash, 281
  - htmlentities, 78, 250, 285
  - is\_array, 139
  - isset, 235
  - mktime, 149
  - O, 449
  - phpinfo, 108
  - print\_r, 119
  - printf, 145
    - dopełnianie łańcuchów, 148
    - modyfikator, 146
    - określanie precyzji, 146
  - real\_escape\_string, 247
  - rename, 155
  - reset, 143
  - S, 450
  - sanitizeMySQL, 265

- funkcja
  - sanitizeString, 265
  - session\_destroy, 288
  - session\_regenerate\_id, 291
  - setcookie, 277
  - shuffle, 140
  - sort, 139
  - strtoupper, 110
  - time, 149
  - toDataURL, 529
  - unlink, 155
- funkcje
  - definiowanie, 109
  - do obsługi
    - czasu, 668
    - daty, 663
    - łańcuchów znaków, 661
    - tablic, 139
  - MySQL, 201, 661
  - obiekty, 335
  - PHP, 107
  - projektu
    - createTable, 606
    - destroySession, 606
    - queryMysql, 606
    - sanitizeString, 607
    - showProfile, 607
  - przekazywanie argumentów, 111
  - składnia, 331
  - tablica arguments, 332
  - zwracanie
    - tablicy, 111, 334
    - wartości, 110, 333
    - zmiennych globalnych, 113
  - zwrotne, 494

## G

- GD, Graphics Draw, 34
- generowanie identyfikatora sesji, 291
- geolokacja, 521, 585, 586
- GPS, 521, 585
- gradient, 410, 532
  - kołowy, 535
  - liniowy, 534
- grupowanie, 198, 357

## H

- hermetyzacja, 117
- hierarchia DOM, 307
- HTML, 26
  - instrukcje JavaScript, 455
  - JavaScript, 293

- komentarze, 295
- tekst, 293, 311
- HTML5, 29, 33, 35, 165, 519
  - aplikacje, 34
    - offline, 594
    - sieciowe, 525
  - atrybut
    - autocomplete, 269
    - autofocus, 269
    - form, 270
    - list, 271
    - min, 271
    - placeholder, 269
    - required, 269
    - step, 271
    - width, 270
  - atrybuty nadpisanie, 270
  - element audio, 575
  - element video, 578
  - formularze, 524
  - geolokacja, 521, 585
  - inne znaczniki, 603
  - komunikacja między dokumentami, 598
  - magazyn danych, 524
  - magazyn lokalny, 590
  - mikrodane, 525, 601
  - nowe API, 34
  - obiekt canvas, 520, 527
  - obsługa audio i wideo, 523, 573
  - pole wejściowe typu
    - color, 272
    - number, 272
    - range, 272
  - przeciągnij i upuść, 596
  - selektory daty i czasu, 272
  - składnia, 33
  - usługi GPS, 585
  - web workers, 525, 593
- HTTP, 26

## I

- IDE, Integrated Development Environment, 52
- identyfikator ID, 389, 309
- identyfikator sesji, 291
- identyfikatory wstawionych wierszy, 244
- importowanie stylów CSS, 388
- indeksy, 185
- indeksy FULLTEXT, 189
- informacje
  - na temat CSS, 654
  - na temat HTML5, 654
  - na temat JavaScriptu, 654



- na temat MySQL, 653
  - na temat PHP, 653
  - na temat technologii AJAX, 654
  - o elemencie nadrzędnym, 511
  - o metodzie addColorStop, 534
  - o tabeli, 240
  - inicjowanie sesji, 285
  - inkrementacja, 303
  - instalowanie pakietu
    - LAMP, 50
    - XAMPP, 40–46, 49
  - instancja, 116
  - instrukcja, *Patrz także* polecenie
    - break, 96, 324
    - continue, 103, 327
    - date, 150
    - echo, 68, 72
    - else, 92, 322
    - elseif, 93
    - flock, 158
    - for
      - wyrażenie inicjalizujące, 101
      - wyrażenie modyfikujące, 101
      - wyrażenie warunkowe, 101
    - foreach, 135
    - if, 91, 322
    - include, 114, 229
    - include\_once, 114
    - mysqldump, 221
    - print, 72
    - require, 115
    - require\_once, 115
    - switch, 95, 323
      - akcja domyślna, 96
      - alternatywna składnia, 96
  - instrukcje
    - MySQL, 173
    - wielowierszowe, 68
  - interfejs, 117
  - interfejs cgi-fcgi, 279
  - Internet Explorer, 308, 469
  - interwał, 463
  - IP, Internet Protocol, 586
- J**
- JavaScript, 31, 35, 293
    - asocjacyjność, 316
    - debugowanie kodu, 297
    - dodawanie nowych elementów, 457
    - dostęp do CSS, 449, 452
    - funkcje, 305, 331
    - komentarze, 299
    - konstrukcja try ... catch, 321
    - literały, 314
    - łączenie łańcuchów, 303
    - obiekty, 335
    - operatory, 301, 315
    - pętle, 325
    - przerwania, 460
    - tablice, 339
    - typ zmiennej, 304
    - typowanie jawne, 328
    - weryfikowanie danych, 349
    - wyrażenia, 313
      - regularne, 355, 362
      - warunkowe, 322
    - zmiennie, 299, 305, 314
    - znaki modyfikujące, 303
  - jednostka
    - cal, 403
    - centymetr, 403
    - em, 404
    - ex, 404
    - milimetr, 403
    - pica, 403
    - piksel, 403
    - procent, 404
    - punkt, 403
  - język
    - COBOL, 30
    - Perl, 29
    - PHP, 29
    - SQL, 30
  - języki skryptowe, 29
  - jQuery, 35, 467
    - bez selektorów, 512
    - dołączanie, 468
    - dynamiczne stosowanie klas, 499
    - efekty specjalne, 487
    - manipulowanie drzewem DOM, 494
    - metoda ready, 476
    - metody, 676–688
    - modyfikowanie wymiarów, 499
    - narzędzia Ajax, 514
    - obiekty, 671
    - obsługa zdarzeń, 475
    - rozszerzenia, 515
    - selektory, 473, 671
    - składnia, 471
    - zdarzenia, 477
    - zdarzenie onload, 476
  - jQuery Mobile, 516
  - jQuery User Interface, 515

## K

- kanał RSS, 34, 384
- kasowanie pliku, 155
- katalog root, 47
- klasa, 116, 335
  - Subscriber, 127
  - User, 127
  - XMLHttpRequest, 380
- klasy znaków, 357
- klonowanie obiektów, 120
- klucz, 181
- klucz główny, 186, 188, 206
- kod PHP, 55
- kodek audio
  - AAC, 574
  - MP3, 574
  - PCM, 574
  - Vorbis, 574
- kodek wideo
  - H.264, 579
  - MP4, 578
  - OGG, 578
  - Theora, 579
  - VP8, 579
  - VP9, 579
  - WebM, 579
- kolory
  - HSL, 436
  - HSLA, 436
  - RGB, 437
  - RGBA, 437
- komentarze, 57, 299, 391
- komentarze HTML, 295
- komponenty modyfikatora konwersji, 147, 148
- kommunikacja
  - między dokumentami, 598
  - z przeglądarką, 276
- komunikat
  - błędu, 297, 298
  - do ramki, 598
  - o błędzie krytycznym, 229
- konfigurowanie
  - pakietu XAMPP, 46
  - serwera, 39
  - właściwości tła, 426
- konflikt między bibliotekami, 472
- konkatenacja łańcuchów znaków, 67, 303
- konsola błędów, 297
- konstrukcja
  - if ... else if, 323
  - JOIN ... ON, 200
  - MATCH ... AGAINST, 194, 195

- try ... catch, 321
- UPDATE ... SET, 196
- konstruktor, 121
- konstruktory podklas, 128
- konwencja bumpyCaps, 336
- kończenie sesji, 288
- kopia zapasowa
  - pliku, 222
  - tabeli, 224
  - wszystkich tabel, 224
- kopiowanie
  - obrazu, 529
  - plików, 155
  - z elementu canvas, 557
- kreślenie ścieżek, 543
- krzywe, 549
- kwalifikator
  - DISTINCT, 191
  - LIMIT, 194

## L

- LAMP, 39, 50
- liczba
  - selektorów, 400
  - zwracanych wyników, 194
- Linux, 50
- literały, 83
- localhost, 47
- logi serwera Apache, 46
- logowanie, 50, 228, 614
- lokalizacja, 586, 589

## Ł

- łańcuchowanie metod, 493
- łączenie
  - selektorów, 474
  - tabel, 198, 200, 244
  - zdarzeń i obiektów, 456

## M

- Mac OS X, 49
- MAC, Media Access Control, 586
- magazyn
  - danych, 524
  - lokalny, 590, 591
- magiczne cudzysłowy, 247
- MAMP, 39
- manipulowanie drzewem DOM, 494
- mapa interaktywna, 589

- marginsy, 416
- metaznaki, 361
- metoda, 117
  - \$.each, 512
  - \$.map, 513
  - addColorStop, 534
  - arc, 550
  - arcTo, 552
  - attr, 496
  - bezierCurveTo, 554
  - clearRect, 531
  - clip, 546
  - createImageData, 562
  - createLinearGradient, 532
  - createRadialGradient, 535
  - css, 473
  - drawImage, 555
  - fadeIn, 489
  - fadeOut, 489
  - fadeTo, 489
  - fadeToggle, 489
  - fillRect, 530
  - fillText, 540
  - forEach, 343, 344
  - get, 514
  - GET, 378, 380
  - getImageData, 559
  - height, 499
  - hide, 479
  - html, 495
  - innerHeight, 502
  - innerWidth, 502
  - is, 511
  - isPointInPath, 548
  - join, 344
  - lineTo, 544
  - measureText, 541
  - moveTo, 544
  - outerHeight, 502
  - outerWidth, 502
  - parents, 505, 506
  - parentsUntil, 505, 506
  - pop, 344
  - post, 514
  - push, 344
  - putImageData, 561
  - quadraticCurveTo, 552
  - query, 230
  - ready, 476
  - rect, 544
  - restore, 566
  - reverse, 345
  - rotate, 566
  - save, 566
  - scale, 564
  - setTransform, 570
  - slideUp, 479
  - sort, 345
  - stroke, 544
  - strokeRect, 531
  - strokeText, 538
  - text, 495
  - toggle, 489
  - transform, 568
  - translate, 567
  - val, 496
  - width, 499
- metody
  - definiowania reguł, 399
  - do obsługi tablic, 342
  - jQuery, 676–688
  - obiektu XMLHttpRequest, 374
  - statyczne, 123, 125, 338
  - typu final, 129
- mikrodane, 525, 601, 602
- model pudełkowy, 416
- moduł wiadomości, 628
- modyfikator konwersji, 147, 148
- modyfikatory
  - formatowania daty, 151
  - ogólne, 362
- modyfikowanie
  - elementów, 472
  - stylów, 453
  - wymiarów, 499
- MySQL, 30, 167
  - archiwizacja, 221
  - bezpieczny dostęp, 247, 250
  - dostęp z phpMyAdmin, 201
  - dostęp z wiersza poleceń, 168, 172
  - działania prewencyjne, 247
  - elementy zastępcze, 248
  - funkcje, 201, 661
  - indeksy, 185
  - instrukcje, 173
  - na zdalnym serwerze, 171
  - normalizacja, 207
  - obsługa tabeli, 239
  - obsługa zaawansowana, 205
  - przywracanie danych, 221
  - relacje, 214
  - słowa z grupy stopwords, 657
  - stosowanie PHP, 227
  - transakcje, 217
  - tworzenie tabeli, 175
  - typy danych, 177

## MySQL

- weryfikacja danych, 246
- wprowadzanie danych, 182
- zapytania, 190, 227

## N

- nagłówek dokumentu, 295
- nawiązywanie połączenia, 229
- nawigowanie, 502
- nazwa
  - kolumny, 184
  - tabeli, 183
  - zmiennej, 63
- nieudana autoryzacja, 280
- normalizacja, 207, 214
- numer ISBN, 180

## O

- obiekt, 107
- obiekt, 116
  - canvas, 520
    - dostęp, 527
    - efekty graficzne, 562
    - kreślenie ścieżek, 543
    - metoda clip, 546
    - metoda isPointInPath, 548
    - obsługa obrazków, 555
    - przekształcenia, 564
    - rysowanie linii, 541
    - tworzenie, 527
    - umieszczanie napisów, 538
    - wypełnianie obszarów, 545
    - zastosowanie krzywych, 549
  - localStorage, 591
  - XMLHttpRequest, 372
    - metody, 374
    - właściwości, 373
- objektowy model dokumentu, DOM, 293, 307
- obiekty
  - jQuery, 674
  - w JavaScriptcie, 335
- obliczanie specyficzności, 400
- obracanie obrazków, 568
- obsługa
  - audio i wideo, 523
  - błędów, 349
  - czasu, 668
  - daty, 663
  - daty i czasu, 149
  - edytora kodu, 51
  - formularzy, 253

- FTP, 51
- grafiki, 34
- łańcuchów znaków, 661
- MySQL, 205
- mysqli, 251
- myszy, 485
- naciśnięć klawiszy, 481
- obrazków, 555
- plików, 152
- sesji, 285
- starszych przeglądarek, 295, 577, 581
- środowiska IDE, 52
- tablic, 139, 342
- transakcji, 218
- zdarzeń, 475, 479
- żądania Ajax, 375
- żądań i odpowiedzi, 28
- oczekiwanie na gotowość dokumentu, 476
- oczyszczanie danych wejściowych, 264
- odbieranie wiadomości, 599
- odczytywanie
  - adresu URL, 308
  - danych, 242, 254
  - pliku, 153, 154
  - wymiarów elementów, 501
  - zmiennych sesji, 287
- odstępny, 408, 419
- odsyłacz, 497
- odtwarzacz
  - flowplayer.swf, 581
  - wideo Flash, 578, 582
- odtworzenie
  - danych, 224
  - dźwięku, 576
  - filmów, 523, 579
- odwołanie do obiektu, 119
- okno
  - autoryzacji HTTP, 278
  - Wymagane uwierzytelnienie, 278
- określanie
  - czasu trwania sesji, 289
  - formatu obrazu, 530
  - precyzji, 146
  - zakresu, 358
- określenia kolorów, 409
- OOP, object-oriented programming, 116
- opcja
  - AUTO\_INCREMENT, 243
  - register\_globals, 256
- opcje kompozycji, 563, 565
- operator, 84, 315
  - \$, 424
  - \*, 424

- ?, 97, 324
- ^, 424
- <<<, 68
- extends, 126
- identyczności, 88
- parent, 127
- równoważności, 87, 316
- zaprzeczenia identyczności, 88
- zaprzeczenia równości, 88
- operatory
  - arytmetyczne, 63, 301
  - logiczne, 65, 89, 201, 302, 318
  - porównania, 64, 89, 302, 318
  - przypisania, 63, 302
  - relacji, 87, 316
- opóźnienia, 461
- opóźnienie przejścia, 444

## P

- pakiet
  - LAMP, 50
  - XAMPP, 40
  - WAMP, 48
- panel sterowania XAMPP, 45
- parametry instrukcji GRANT, 174
- pętla, 325
  - do ... while, 100, 326
  - for, 101, 326
  - foreach ... as, 134
  - while, 98, 325
- PHP, 29, 55
  - asocjacyjność, 86
  - autoryzacja, 283
  - deklaracja typu, 70
  - dodawanie elementów do HTML, 55
  - dołączanie pliku, 114
  - dynamiczne linkowanie, 105
  - dziedziczenie, 126
  - funkcja printf, 145
  - funkcje, 73, 107
  - instrukcje wielowierszowe, 68
  - komentarze, 57
  - komunikacja z MySQL, 227
  - literały, 83
  - łączenie łańcuchów, 67
  - nazwy zmiennych, 63
  - obiekty, 116
  - obsługa daty i czasu, 149
  - obsługa plików, 152
  - operatory, 63
  - operatory relacji, 87
  - pętla foreach ... as, 134

- pętle, 98
- pobieranie rezultatu, 230
- priorytet operatorów, 84
- rzutowanie, 104
- składnia, 57, 58
- sprawdzanie zgodności wersji, 115
- stałe, 71
- stałe predefiniowane, 71
- tablice, 61, 131
- typy operatorów, 84
- uruchamianie kodu, 56
- weryfikacja formularza, 364
- wyrażenia, 81
  - regularne, 363
  - warunkowe, 91
- wywołania systemowe, 163
- zamykanie połączenia, 233
- zapytanie do MySQL, 230
- zasięg zmiennych, 74–77
- zmienianie wartości zmiennych, 66
- zmiennne, 59, 83
- znaki modyfikujące, 67
- żądanie pliku, 114

PHP 5

- destruktory, 122
- metody statyczne, 123
- zasięg właściwości i metod, 124

phpMyAdmin, 201

pierwsza postać normalna, 207

piksel, 558

planowanie tworzenia kopii zapasowych, 225

plik

- checkuser.php, 614
- clock.appcache, 595
- clock.html, 595
- clock.js, 595
- formtest.php, 253, 255
- friends.php, 623
- functions.php, 606, 607
- header.php, 608
- index.php, 611
- javascript.js, 632
- login.php, 228, 230, 283, 615
- logout.php, 629
- members.php, 621
- messages.php, 626
- OSC.js, 587
- profile.php, 616
- setup.php, 609
- signup.php, 611
- styles.css, 629
- sqltest.php, 233, 238
- urlget.php, 515

- plik
  - urlpost.php, 514
  - worker.js, 594
  - xmlget.html, 383
- pliki
  - .htm, 55
  - .php, 55
  - .csv, 225
  - aktualizowanie, 156
  - kasowanie, 155
  - kopia zapasowa, 222
  - kopiowanie, 155
  - odczytywanie, 153, 158
  - przenoszenie, 155
  - sprawdzanie istnienia, 152
  - tryby otwarcia, 154
  - tworzenie, 152
  - wielokrotne otwarcie, 157
  - wysyłanie, 159
- pobieranie
  - rezultatu, 230
  - wiersza danych, 232
- podklasa, 117, 128
- podstawianie niepożądaných danych, 249
- pola
  - opcji, 259
  - ukryte, 262
- pole
  - tekstowe, 258
  - tekstowe wielowierszowe, 258
  - wejściowe
    - color, 272
    - number, 272
    - range, 272
- polecenia SQL, 173
- polecenie
  - ALTER, 181, 183, 186
  - BEGIN, 219
  - CHANGE, 184
  - COMMIT, 219
  - CREATE, 174, 181, 186
  - DELETE, 192
  - SELECT, 190
  - DROP, 184, 187
  - EXPLAIN, 220
  - GRANT, 174
  - INSERT, 182
  - ROLLBACK, 219
- połączenie z MySQL, 229
- położenie
  - bezwzględne, 411
  - stałe, 412
  - względne, 412
- powtarzanie
  - cykliczne, 461
  - obrazka, 537
- praca zdalna, 50
- precyzja rezultatu, 146
- priorytet
  - operatorów, 84, 86, 315
  - reguły, 401
- procedura żądanie/odpowiedź, 26
- profil użytkownika, 620
- program
  - adduser.php, 364
  - Ajax, 374
  - convert.php, 266
  - FileZilla, 51
  - FireFTP, 51
  - phpMyAdmin, 201
  - query.php, 231
  - sqltest.php, 233
  - xmlget.php, 380
- programowanie obiektowe, OOP, 116
- projekt serwisu społecznościowego, 605
  - checkuser.php, 614
  - friends.php, 623
  - functions.php, 606
  - header.php, 608
  - index.php, 611
  - javascript.js, 632
  - login.php, 615
  - logout.php, 629
  - members.php, 621
  - messages.php, 626
  - profile.php, 616
  - setup.php, 609
  - signup.php, 611
  - styles.css, 629
- projektowanie bazy, 205
- prywatność, 250
- przechowywanie loginów i haseł, 281
- przechwytywanie
  - naciśnięć klawiszy, 480
  - zdarzeń myszy, 483
- przeciąganie i upuszczanie obiektów, 596
- przejmowanie sesji, 289
- przejścia, 443
  - czas trwania, 444
  - dynamika, 444
  - opóźnienie, 444
  - skrótowa składnia, 445
  - właściwości, 443
- przekazywanie
  - argumentów przez referencję, 111
  - łańcucha znaków, 460
  - niepożądaných danych, 249

- przekształcenia, 441, 564
- przekształcenia 3D, 442
- przeliczanie temperatur, 266, 267
- przełączniki, 261
- przenoszenie pliku, 155
- przerwanie, 460
  - setInterval, 461
  - setTimeout, 460
- przerywanie pętli, 102, 327
- przesłanianie, 549
- przesuwanie elementów, 490
- przesyłanie żądań XML, 380
- przetwarzanie
  - obrazu, 559, 618
  - selekcji, 509
  - wyselekcjonowanych elementów, 510
- przezroczystość, 435
- przycisk wysyłania, 264
- przypisywanie tekstu do zmiennej, 69
- przywracanie danych, 221
- pseudoklasy, 413
- punkt
  - dostępowy WiFi, 586
  - kontrolny, 534

## R

- ramka, 418
- ramka iframe, 598, 600
- ramki w CSS3, 430
- reguły CSS, 390
  - skrącanie, 415
  - wiele deklaracji, 390
  - zastosowanie komentarzy, 391
- reguły dotyczące ramek, 418
- relacja
  - jeden do jednego, 214
  - jeden do wielu, 215
  - wiele do wielu, 216
- relacyjne bazy danych, 206
- rodzaje
  - luków, 551
  - pól, 258
- rozmieszczanie elementów, 411
- rozszerzanie obiektów, 339
- rozszerzenia jQuery, 515
- rozszerzenie FireFTP, 52
- RSS, Really Simple Syndication, 34
- rysowanie
  - chmury, 553
  - linii, 541
  - luków, 550, 553
  - obrazków, 557

- prostokątów, 531
- w widocznym obszarze, 548
- rzutowanie
  - jawne, 104
  - niejawne, 104

## S

- sekcja
  - <body>, 597
  - <script>, 452
- selekcja
  - elementów, 474, 508
  - grupowa, 398
- selektor
  - atributu, 396, 423
  - dziecka, 394
  - elementu, 474
  - klasy, 396, 474
  - potomka, 393
  - typu, 393
  - uniwersalny, 397, 630
- selektory
  - arkuszy stylów, 399
  - daty i czasu, 272
  - identyfikatorów, 395, 474
  - jQuery, 671
- serwer
  - Apache, 34
  - MySQL, 172
  - WWW, 34
  - XAMPP, 47
- serwisy
  - informacyjne, 655
  - społecznościowe, 605
  - typu BBS, 25
- sesja, 285
  - bezpieczeństwo, 289
  - czas trwania, 289
  - inicjowanie, 285
  - kończenie, 288
- sieci CDN, 469
- skalowanie obrazu, 555
- składnia
  - heredoc, 68
  - PHP, 57, 58
- składowanie danych, 218
- skrącanie reguł, 415
- słowa z grupy stopwords, 657
- słowo kluczowe
  - array, 133
  - AS, 200
  - COUNT, 191

słowo kluczowe

- DROP, 184
  - global, 113
- MODIFY, 183
- ORDER BY, 197
- prototype, 337
- this, 455, 478
- var, 305
- WHERE, 192

snippets, 603

solenie, salting, 281

sortowanie, 139, 197

specyficzność reguły, 400

sprawdzanie

- adresu e-mail, 354
- danych logowania, 280
- dostępności nazwy, 613
- hasła, 354
- imienia, 353
- istnienia pliku, 152
- nazwiska, 353
- nazwy użytkownika, 353
- poprawności daty, 151
- wieku, 354

SQL, Structured Query Language, 30, 167

stała, 71

- DATE\_ATOM, 150
- DATE\_COOKIE, 150
- DATE\_RSS, 150
- DATE\_W3C, 150

stałe predefiniowane, 71

starsze przeglądarki, 295, 577, 581

sterowanie odtwarzaniem, 581

stopniowe zanikanie elementu, 489

stosowanie normalizacji, 214

strona

- główna serwisu, 611

HTML, 387

- logowania, 617
- rejestracji, 614
- wylogowania, 629

style

- bezpośrednie, 393
- domyślne, 392
- użytkownika, 392
- wewnętrzne, 393
- zagnieżdżone, 389

superklasa, 117

## Ś

środowisko

- IDE, 52
- IDE dla PHP, 54
- phpDesigner, 52

## T

tabele

- dodawanie danych, 241
- identyfikatory wstawionych wierszy, 244
- odczytywanie danych, 242
- tworzenie, 239
- usuwanie, 241
- usuwanie danych, 243
- wyświetlanie informacji, 240
- zastosowanie blokad, 244

tablica, 131, 300

- \$\_COOKIE, 277
- \$\_FILES, 160, 161
- \$\_POST, 235
- \$\_SERVER, 279
- \$\_SESSION, 288
- arguments, 332
- data, 560

tablice

- asocjacyjne, 133, 341
- dodawanie pozycji, 133
- dwuwymiarowe, 61
- indeksowane numerycznie, 131
- numeryczne, 340
- pętla foreach ... as, 134
- w JavaScriptcie, 339
- wielowymiarowe, 136, 341

technologia

- Ajax, 29, 32, 371, 514
- przeciągnij i upuść, 596
- web workers, 525, 593

testowanie instalacji XAMPP, 47

tła, 425

transakcje, 217

tryby otwarcia pliku, 154

trzecia postać normalna, 212

tworzenie

- bazy danych, 174
- ciasteczka, 276
- cieni, 557
- formularzy, 253
- indeksu, 185
- kopii zapasowej tabeli, 224
- kopii zapasowej wszystkich tabel, 224, 225
- metod, 122
- obiektu, 118, 337
- pliku, 152
- pliku logowania, 228
- tabeli, 175, 239
- użytkowników, 174
- zapytań, 190



typ  
  AUTO\_INCREMENT, 180  
  BINARY, 178  
  CHAR, 177  
  DATE, 180  
  INT UNSIGNED, 181  
  SMALLINT, 187  
  TEXT, 178  
  TIME, 180  
  UNSIGNED SMALLINT, 183  
  VARCHAR, 178

typografia, 405

typowanie  
  jawne, 328  
  zmiennych, 304

typy  
  danych, 177  
    binarnych, 178  
    BLOB, 179  
    liczbowych, 179  
    TEXT, 178  
  literałów, 314  
  łańcuchów, 67

## U

uchwyt pliku, file handle, 153, 156

układ  
  strony, 416  
  wielokolumnowy, 434

ukrywanie elementów, 488

uruchamianie wiersza poleceń, 168

usługi GPS, 585

ustanawianie sesji, 286

usuwanie  
  ciasteczek, 277  
  danych, 192, 233, 243  
  elementów, 459, 496  
  kolumny, 184  
  rekordu, 236  
  tabeli, 184, 241  
  znajomych, 622

używanie XML, 384

## W

walidacja, 161

WAMP, 39, 48

wartości  
  boolowskie, 81  
  domyślne, 257

wartość  
  FALSE, 82

  NULL, 82, 181  
  TRUE, 82  
wcięcia, 408  
Web 2.0, 32  
web workers, 525, 593  
wersje PHP, 115  
weryfikacja  
  danych, 349  
  formularza, 364  
wielkość znaków, 408  
wiersz poleceń, 168  
  obsługa serwera, 172  
  w Linuksie, 170  
  w OS X, 169  
  w Windows, 168  
własność  
  font, 539  
  globalAlpha, 564  
  globalCompositeOperation, 562  
  lineCap, 541  
  lineJoin, 541  
  lineWidth, 541  
  miterLimit, 543  
  textAlign, 539  
  textBaseLine, 539  
właściwości  
  obiektu XMLHttpRequest, 373  
  okien, 454  
  przejsć, 443  
  statyczne, 125, 338  
właściwość, 117, *Patrz także* własność  
  auto, 428  
  background-clip, 425  
  background-origin, 425  
  background-size, 427  
  border-color, 430  
  border-radius, 430  
  box-sizing, 425  
  innerHTML, 462  
  offsetTop, 458  
  opacity, 437  
  overflow, 434  
  readyState, 375, 376  
  responseText, 382  
  responseXML, 382  
  textDecoration, 452  
  text-overflow, 438  
  text-shadow, 438  
  word-wrap, 439  
włączenie ciasteczek, 291  
wskaźnik pliku, file pointer, 156  
współdzielony serwer, 292  
wstawianie danych, 233

- wybieranie
    - danych, 190
    - fragmentu obrazu, 556
  - wycinki informacji, 603
  - wypełnianie
    - gradientem, 532
    - obszarów, 545
    - tekstu wzorkiem, 540
    - wzorkami, 536
    - złożonej ścieżki, 545
  - wyrażenia
    - logiczne, 82, 91
    - regularne
      - dopasowanie rozmyte, 356
      - grupowanie, 357
      - klasy znaków, 357
      - metaznaki, 355, 361
      - modyfikatory ogólne, 362
      - określanie zakresu, 358
      - w JavaScriptcie, 362
      - w PHP, 363
      - zaprzeczenie, 358
    - warunkowe, 91, 322
    - z elementami zastępczymi, 248
  - wyrównanie, 408
  - wysyłanie
    - plików, 159
    - zapytań, 237
    - zadań i odpowiedzi, 27
  - wyświetlanie
    - bieżącego profilu, 618
    - elementów, 488
    - formularza, 237
    - listy użytkowników, 622
    - mapy, 522, 586
    - profilu użytkownika, 622
    - znajomych, 626
  - wywołania systemowe, 163
  - wyzwalanie zdarzeń, 457
  - wzorki, 536
- X**
- XAMPP, 40
  - XHTML, 165
  - XML, 382
  - XSS, cross-site scripting, 250
- Z**
- zadanie działające w tle, 593
  - zakończenia linii, 542
  - zamykanie połączenia, 233
  - zaokrąglanie rogów, 431
  - zapisywanie danych, 225
  - zapobieganie próbom ataków, 246
  - zaprzeczenie, 358
  - zapytania pomocnicze, 245
  - zapytanie do MySQL, 230
  - zarządzanie stylami tekstu, 407
  - zasięg
    - właściwości i metod, 124
    - zmiennych, 74, 114
  - zastosowanie
    - ciasteczek, 275
    - CSS, 32
    - elementów zastępczych, 248
    - elementu canvas, 520
    - filtra, 503
    - funkcji alert, 311
    - funkcji checkdate, 150
    - funkcji console.log, 310
    - funkcji document.write, 311
    - funkcji printf, 145
    - funkcji sprintf, 149
    - identyfikatorów ID, 389
    - instrukcji CREATE INDEX, 186
    - JavaScriptu, 31
    - jQuery, 471, 474
    - klas, 389
    - komentarzy, 57, 299
    - krzywych, 549
    - magazynu lokalnego, 590
    - metody each, 512
    - metody get, 378, 514
    - metody post, 514
    - MySQL, 30
    - mysql, 251
    - obiektowego modelu dokumentu, 309
    - obiektu XMLHttpRequest, 372
    - opcji AUTO\_INCREMENT, 243
    - operatorów logicznych, 201
    - PHP, 29
    - platform Ajax, 384
    - przerwania setInterval, 461
    - przerwania setTimeout, 460
    - skryptów, 295
    - symbolu \$, 309
    - średników, 389
    - tablicy \$\_FILES, 160
    - technologii Ajax, 371, 514
    - właściwości auto, 428
  - zatrzymywanie animacji, 494
  - zawartość obiektu, 420
  - zawijanie tekstu, 259
  - zdalny dostęp do serwera MySQL, 50

- zdarzenia, 457, 477
- zdarzenia myszy, 484
- zdarzenie
  - blur, 477
  - click, 479
  - dblclick, 479
  - focus, 477
  - keypress, 480
  - mousemove, 482
  - onerror, 320
  - onload, 476
  - submit, 486
- zewnętrzne arkusze stylów, 393
- zintegrowane środowisko programistyczne, IDE, 52
- zmiana
  - nazwy kolumny, 184
  - nazwy tabeli, 183
  - typu danych, 183, 328
- zmienna
  - \$\_SERVER, 78
  - \$fh, 153
- zmienne, 83, 299, 314
  - globalne, 75, 114, 305
  - lokalne, 74, 114, 306
  - numeryczne, 60, 300
  - przypisywanie wartości, 66
  - statyczne, 76, 114
  - superglobalne, 77, 78
  - tekstowe, 59
  - znakowe, 300
- znacznik
  - \_END, 69
  - <audio>, 575
  - <div>, 403, 458
  - <form>, 253
  - <head>, 387
  - <input>, 256
  - <noscript>, 294
  - <?php, 56
  - <script>, 293, 455
  - <select>, 262
  - <source>, 580
  - <span>, 403
  - <video>, 578
- znaczniki HTML5, 603
- znak
  - dolara, 58, 119, 309
  - lewego ukośnika, 247
  - średnika, 58, 172, 299, 389
- znaki
  - modyfikujące, 67, 303
  - nowego wiersza, 69
  - podwójnego ukośnika, 296
- zwracanie
  - tablicy, 111, 334
  - wartości, 110, 333
  - zmiennych globalnych, 113

**Ż**

źródła stylów, 398

**Ż**

żądanie GET, 380



# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ 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 WYDAWNICZA

 **Helion SA**

## Kompendium wiedzy dla twórcy stron i aplikacji sieciowych!

PHP wraz z bazą danych MySQL oraz językiem JavaScript to potężne trio, dzięki któremu możesz zbudować aplikację internetową dowolnej wielkości. Jeżeli do tego dołożysz możliwości najnowszej wersji języka HTML (oznaczonej cyfrą 5), CSS3 oraz bibliotekę jQuery, nic nie będzie w stanie zatrzymać Twojej kreatywności.

Jeżeli chcesz opanować te narzędzia i stworzyć atrakcyjną oraz funkcjonalną aplikację internetową, trafiłeś na idealną książkę. Znajdziesz tu niezbędną wiedzę o języku PHP, bazie danych MySQL, HTML5, CSS3, JavaScriptcie i jQuery. Już od pierwszych stron będziesz poznawać składnię i konstrukcje języka programowania PHP, techniki programowania obiektowego oraz praktyczne porady związane z używaniem PHP. Następnie uzupełnisz wiedzę na temat bazy danych MySQL. Dowiesz się, jak tworzyć zapytania SQL oraz w jaki sposób wykorzystywać dane zawarte w bazie z poziomu PHP. Po opanowaniu „strony serwerowej” przejdziesz do nauki technik tworzenia interaktywnych stron WWW. Zobaczysz, jak używać języka JavaScript, jakie nowości zawiera HTML5 oraz jak wielki potencjał kryją w sobie CSS3 i jQuery. Książka ta jest ciekawą lekturą dla pasjonatów chcących tworzyć własne, zaawansowane aplikacje.

**Robin Nixon** – publicysta, autor setek artykułów poświęconych technologiom komputerowym, związany z branżą IT od wczesnych lat 80. XX wieku. Stworzył liczne strony WWW przy użyciu narzędzi open source. Jest specjalistą w zakresie PHP, MySQL, JavaScriptu i HTML.

Dzięki tej książce:

- przygotujesz środowisko pracy
- poznasz język PHP oraz jego możliwości
- wykorzystasz możliwości bazy danych MySQL
- zbudujesz aplikację internetową

**Helion** 

32799 numer katalogowy

księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne

 **0 801 339900**

 **0 601 339900**

Sprawdź najnowsze promocje:  
● <http://helion.pl/promocje>  
Książki najchętniej czytane:  
● <http://helion.pl/bestsellery>  
Zamów informacje o nowościach:  
● <http://helion.pl/nowości>

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-283-0842-8



9 788328 308428

Informatyka w najlepszym wydaniu

cena: 99,00 zł