

PHP i MySQL. Dla każdego

Wydanie II

Dołącz do grona twórców dynamicznych serwisów WWW – poznaj najbardziej popularne technologie w branży!

- Opanuj zasady programowania w języku PHP
- Poznaj sposoby korzystania z bazy MySQL
- Naucz się łączyć możliwości tych technologii
- Dowiedz się, jak za ich pomocą tworzyć praktyczne rozwiązania

PHP i MySQL

Dla każdego

Marcin Lis



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.

Redaktor prowadzący: Ewelina Burska

Projekt okładki: Maciej Pasek

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie?phmdk2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Materiały do książki można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/phmdk2.zip>

ISBN: 978-83-246-4797-2

Copyright © Helion 2013

Printed in Poland.

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

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

Spis treści

Wstęp	11
Część I Skrypty PHP — dynamiczne generowanie stron internetowych	13
Rozdział 1. Podstawy	15
Czym jest PHP?	15
Krótka historia PHP	16
Niezbędne narzędzia	16
Instalacja w systemie Windows	17
Serwer WWW	18
Środowisko PHP	21
Testowanie instalacji	23
Instalacja w Linuksie	23
Instalacja przy użyciu pakietów	23
Konfiguracja PHP	27
Typowe problemy	27
Brak połączenia z serwerem	27
Serwer nie działa po instalacji PHP	28
Serwer działa, ale nie obsługuje PHP	29
Pierwszy skrypt	30
Jak to działa?	31
Rozdział 2. Znaczniki, zmienne i typy danych	33
Umieszczanie skryptów w kodzie HTML	33
Znaczniki kanoniczne (klasyczne)	33
Znaczniki skryptów HTML	34
Znaczniki typu SGML	34
Znaczniki typu ASP	34
Proste wyświetlanie danych	35
Skrypty zewnętrzne	35
Instrukcja include	36
Instrukcja require	37
Więcej o dołączaniu plików	38
Komentarze w skryptach	39
Komentarz blokowy	40
Komentarz jednowierszowy	40
Komentarz jednowierszowy uniksowy	41

Typy danych	41
Typy skalarne	42
Typy złożone	46
Typy specjalne	47
Zmienne	47
Zmienne w PHP	47
Tworzenie zmiennych	48
Jak wykryć typ zmiennej?	49
Zmienne superglobalne	51
Stałe	53
Stałe w PHP	53
Stałe predefiniowane	54
Operatory	55
Operatory arytmetyczne	55
Operatory inkrementacji i dekrementacji	56
Operatory bitowe	59
Operatory logiczne	61
Operatory relacyjne	63
Operator łańcuchowy	63
Operatory przypisania	64
Operatory tablicowe	66
Pozostałe operatory	67
Priorytety operatorów	70
Konwersje typów	70
Zmiana typu zmiennej	70
Rzutowanie typów	72
Funkcje konwersji	73
Zasady konwersji	75
Rozdział 3. Instrukcje sterujące i funkcje	77
Instrukcje warunkowe	77
Instrukcja if...else	77
Instrukcja if...else if	78
Zagnieżdżanie instrukcji warunkowych	80
Wyrażenia warunkowe	84
Operator warunkowy	85
Instrukcja wyboru Switch	85
Pętle	89
Pętla typu for	89
Pętla typu while	93
Pętla typu do...while	95
Pętla typu foreach	96
Składnia alternatywna	98
Instrukcje warunkowe	98
Instrukcja switch	99
Pętle	100
Instrukcje break i continue	101
Instrukcja break	101
Instrukcja continue	103
Funkcje	104
Budowa funkcji	104
Argumenty funkcji	105
Zwracanie wartości	107
Zasięg zmiennych	108
Argumenty funkcji raz jeszcze	112

Rozdział 4. Tablice	117
Rodzaje tablic w PHP	117
Tablice zwykłe	117
Tablice asocjacyjne	120
Tablice wielowymiarowe	124
Tworzenie tablic wielowymiarowych	124
Tablice nieregularne	129
Operacje na tablicach	130
Sortowanie tablic klasycznych	130
Sortowanie tablic asocjacyjnych	133
Implozja i eksplozja	135
Operacje na elementach tablic	136
Zmiana kolejności elementów	136
Poruszanie się po tablicy	137
Dodawanie i pobieranie elementów	139
Liczba elementów tablicy	141
Rozdział 5. Programowanie zorientowane obiektowo	143
Podstawy programowania obiektowego	143
Tworzenie klas	143
Tworzenie obiektów	145
Odwołania do składowych	146
Wskazanie this	149
Konstruktory i destruktory	150
Budowa konstruktora	150
Argumenty konstruktorów	152
Destruktry	153
Dziedziczenie	153
Czym jest dziedziczenie?	153
Przesłanianie składowych	157
Konstruktory klas bazowych	159
Modyfikatory dostępu	160
Wyjątki	162
Zgłaszanie wyjątków	163
Przechwytywanie wyjątków	164
Obsługa kilku wyjątków	165
Rozdział 6. Przetwarzanie danych z przeglądarki	169
Metoda GET	170
Metoda POST	174
Wysyłanie plików (upload)	176
Odbieranie plików (download)	179
Wysłanie pojedynczego pliku	180
Wysyłanie pliku wybranego z listy	181
Automatyczne generowanie listy plików	184
Lista plików przechowywana w pliku tekstowym	187
Rozdział 7. Ciągi znaków, data i czas	193
Ciągi znaków	193
Formatowanie ciągów	195
Porównywanie ciągów	204
Przeszukiwanie ciągów	206
Przetwarzanie ciągów	208

Data i czas	212
Funkcja checkdate	212
Funkcja date	213
Funkcja getdate	215
Funkcja gmdate	216
Funkcja localtime	217
Funkcja microtime	218
Funkcja mktime	218
Funkcja strftime	218
Funkcja strtotime	221
Funkcja time	221
Rozdział 8. System plików	223
Obsługa struktury plików i katalogów	223
Odczyt zawartości katalogu	223
Tworzenie i usuwanie katalogów	227
Zmiana katalogu bieżącego	228
Odczytywanie informacji o plikach	228
Miejsce na dysku	230
Usuwanie zawartości katalogu	231
Nawigacja po katalogach	232
Obsługa plików	235
Otwieranie i zamykanie plików	235
Odczyt danych	237
Zapis danych	243
Poruszanie się po danych w pliku	247
Synchronizacja dostępu	248
Wykorzystanie plików do przechowywania danych	249
Zwykły licznik tekstowy	250
Licznik graficzny	252
Głosowanie	254
Prosty system logowania	260
Generowanie listy odnośników	263
Rozdział 9. Cookies i sesje	265
Krótko o cookies	265
Zapis i odczyt cookies	265
Jak zapisać cookie?	265
Jak odczytać cookie?	268
Jak usunąć cookie?	268
Korzystanie z cookies	269
Mechanizm sesji	271
Obsługa sesji	272
Rozpoczynanie sesji	272
Kończenie sesji	273
Konfiguracja sesji	273
Zmienne sesji	275
Implementacja sesji	276
Uwierzytelnianie z wykorzystaniem mechanizmu sesji	278
Śledzenie użytkownika	284

Część II Tworzenie baz danych w MySQL 287**Rozdział 10. Podstawy MySQL 289**

Czym jest MySQL?	289
Instalacja i konfiguracja	289
Instalacja w systemie Windows	290
Konfiguracja w systemie Windows	293
Instalacja w systemie Linux	294
Zarządzanie serwerem	297
Uruchamianie serwera	297
Kończenie pracy serwera	298
Koncepcja relacyjnych baz danych	300
Tabele	300
Klucze	300
Relacje	301
Jak projektować tabele bazy?	304
Określenie celu	304
Duplikowanie danych (informacje nadmiarowe)	305
Informacje atomowe	306
Puste pola	307
Jednoznaczna identyfikacja rekordów	308
Tworzenie i usuwanie baz	308
Łączenie z serwerem	308
Tworzenie i usuwanie baz	310
Zarządzanie kontami użytkowników	311
Tworzenie kont użytkowników	311
Nadawanie uprawnień	311
Nazwy użytkowników	314
Odbieranie praw	316
Zmiana nazwy konta użytkownika	316
Usuwanie kont użytkowników	317
Sprawdzanie przywilejów	317
Inne czynności zarządzające	318
Praca z wieloma bazami	318
Pobieranie listy baz i tabel	318
Kodowanie znaków	319
Wczytywanie poleceń z plików zewnętrznych	322

Rozdział 11. Podstawy SQL 323

Czym jest SQL?	323
Typy danych w kolumnach	324
Typy liczbowe	324
Typy daty i czasu	327
Typy łańcuchowe	329
Obsługa tabel	331
Tworzenie tabel	331
Pobranie struktury tabeli	334
Modyfikacja tabel	335
Usuwanie tabel	337
Zapytania wprowadzające dane	337
Pierwsza postać instrukcji INSERT	338
Druga postać instrukcji INSERT	339
Wstawianie wielu wierszy	340

Zapytania pobierające dane	341
Pobieranie zawartości całej tabeli	342
Sortowanie wyników	342
Pobieranie zawartości wybranych kolumn	344
Zmiana nazw kolumn w wynikach zapytania	345
Selektywne pobieranie danych	345
Ograniczanie liczby wierszy w wynikach zapytania	351
Zapytania modyfikujące dane	351
Zapytania usuwające dane	353
Wstawianie specjalne	354
Rozdział 12. Więcej o SQL	357
Pobieranie danych z wielu tabel	357
Złączenia	357
Typy złączeń	359
Agregacja (grupowanie) danych	363
Funkcje statystyczne	363
Grupowanie wyników zapytań	367
Warunki grupowania	369
Funkcje agregujące w złączeniach	370
Typy tabel	374
Indeksy	374
Więzy integralności — klucze obce	377
Tworzenie ograniczeń	377
Dodawanie i usuwanie ograniczeń w istniejących tabelach	379
Podzapytania	380
Podzapytania proste	381
Podzapytania skorelowane	382
Podzapytania w klauzuli FROM	384
Podzapytania w instrukcjach INSERT, UPDATE, DELETE	385
Rozdział 13. Tworzenie bazy w praktyce	389
Założenia	389
Diagramy tabel	390
Tworzenie tabel	393
Indeksy i więzy integralności	402
Baza w praktyce	406
Rozdział 14. Współpraca PHP i MySQL	413
Konfiguracja PHP	414
Obsługa bazy za pomocą mysqli (interfejs proceduralny)	415
Łączenie z bazą danych	415
Kończenie połączenia z bazą danych	416
Zmiana domyślnej bazy danych	417
Testowanie połączenia z bazą	417
Obsługa bazy za pomocą mysqli (interfejs obiektowy)	418
Łączenie z bazą danych	418
Kończenie połączenia z bazą danych	418
Zmiana domyślnej bazy danych	419
Testowanie połączenia z bazą	419
Obsługa bazy za pomocą PDO	420
Nawiązywanie połączenia	420
Zamykanie połączenia	420
Testowanie połączenia z bazą	421

Wykonywanie zapytań pobierających dane	421
Styl proceduralny — mysqli	421
Styl obiektowy — mysqli	429
Styl obiektowy — PDO	432
Zapytania typu INSERT, UPDATE, DELETE	437
Styl proceduralny — mysqli	437
Styl obiektowy — mysqli	443
Styl obiektowy — PDO	445
Wybór sposobu obsługi	448
Problem polskich liter	449
Część III PHP i MySQL w praktyce	453
Rozdział 15. Autoryzacje	455
Proste uwierzytelnianie	455
Zasady logowania	459
Uwierzytelnianie z wykorzystaniem sesji	461
Rejestracja nowych użytkowników	467
Rozdział 16. Generowanie statystyk w portalu	479
Wstępne założenia i struktura danych	479
Struktura portalu	481
Funkcje pomocnicze	484
Jak rozpoznać przeglądarkę i system operacyjny?	488
Zapisywanie historii odwiedzin	489
Liczba użytkowników na stronie	491
Część główna	494
Obsługa logowania	497
Generowanie statystyk	505
Rozdział 17. Zarządzanie kontami użytkowników	515
Modyfikacja bazy danych	516
Struktura części administracyjnej	517
Obsługa logowania	522
Sterowanie skryptem zarządzania	527
Wyświetlanie listy użytkowników	531
Dodawanie i modyfikacje rekordów	535
Wyszukiwanie użytkowników	544
Usuwanie danych	549
Rozdział 18. System news	551
Ogólna struktura serwisu i bazy danych	552
Newsy w części frontowej	554
Rozbudowa systemu przywilejów	561
Zarządzanie nowościami w części administracyjnej	565
Wyświetlanie listy wiadomości	568
Dodawanie i edycja wiadomości	571
Wyszukiwanie wiadomości	579
Usuwanie wiadomości i kody powrotów	584
Rozdział 19. Subskrypcje	587
Struktura bazy danych	587
Subskrypcje w części frontowej	589
Struktura części administracyjnej	595
Moduł zarządzania subskrypcjami	598

Rozdział 20. Tworzenie sklepu internetowego	603
Główna część serwisu	603
Logowanie i wylogowanie	609
Rejestracja nowych użytkowników	614
Wyszukiwanie danych	621
Prezentacja szczegółowych danych książki	627
Obsługa koszyka	629
Struktura koszyka	629
Dodawanie książek do koszyka	630
Wyświetlanie zawartości	632
Modyfikacja	636
Integracja koszyka ze sklepem	637
Obsługa zamówień	638
Podsumowanie zamówienia	639
Zapisanie zamówienia w systemie	640
Skorowidz	645

Rozdział 4.

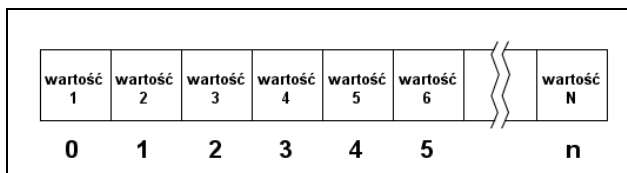
Tablice

Rodzaje tablic w PHP

Tablice to występujące w większości języków programowania struktury, pozwalające na przechowywanie zbioru danych określonego typu. Tablicę można sobie wyobrazić jako wektor elementów, taki jak zaprezentowany na rysunku 4.1. Zawartością pojedynczej komórki tablicy może być wartość dowolnego typu danych (inaczej niż w klasycznych językach programowania). W PHP dostępne są dwa rodzaje tablic: klasyczne (indeksowane numerycznie) oraz asocjacyjne. Dostęp do poszczególnych danych zawartych w tablicy uzyskuje się poprzez podanie indeksu (inaczej klucza), pod którym dana wartość została zapisana.

Rysunek 4.1.

*Struktura
typowej tablicy*



Tablice zwykłe

Aby utworzyć prostą tablicę indeksowaną numerycznie, należy użyć słowa kluczowego `array` w schematycznej postaci:

```
$tablica = array(wartość1, wartość2, ..., wartośćN);
```

gdzie: *tablica* to nazwa zmiennej tablicowej, dzięki której będzie można się do tej tablicy odwoływać, natomiast *wartość1*, *wartość2* itd. to wartości kolejnych komórek. W przypadku dużej liczby wartości w celu zwiększenia czytelności można również zastosować zapis w postaci:

```
$tablica = array  
(  
    wartość1,  
    wartość2,
```

```

    ....
    wartośćN
);

```

Zobaczymy, jak to będzie wyglądać w praktyce. Zobrazowano to w skrypcie widocznym na listingu 4.1.

Listing 4.1. *Deklaracja prostej tablicy*

```

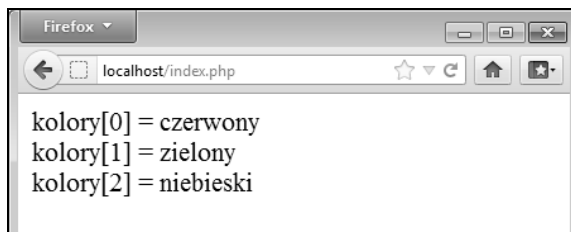
<?php
$kolory = array("czerwony", "zielony", "niebieski");
echo "kolory[0] = $kolory[0] <br />";
echo "kolory[1] = $kolory[1] <br />";
echo "kolory[2] = $kolory[2] <br />";
?>

```

Powstała tutaj tablica `$kolory`, której kolejnym komórkom zostały przypisane ciągi znaków określające kolory: czerwony, zielony i niebieski. Aby uzyskać dostęp do wartości zapisanej w danej komórce, należy podać jej numer (indeks) w nawiasie kwadratowym występującym za nazwą tablicy. Należy przy tym pamiętać, że indeksowanie tablicy zaczyna się od zera, co oznacza, iż indeksem pierwszej komórki jest 0, a NIE 1. Aby zatem odczytać zawartość pierwszej komórki, piszemy `$kolory[0]`, drugiej komórki — `$kolory[1]`, a trzeciej komórki — `$kolory[2]`. Dzięki temu po uruchomieniu skryptu na ekranie ukaże się widok przedstawiony na rysunku 4.2.

Rysunek 4.2.

*Wyświetlenie
zawartości tablicy
kolory*



Do odczytu zawartości tablicy można wykorzystać również pętlę. Są one przydatne w szczególności wtedy, gdy tablica ma duże rozmiary. Na listingu 4.2 został przedstawiony skrypt realizujący takie samo zadanie jak skrypt z listingu 4.1 (czyli utworzenie tablicy i wyświetlenie jej zawartości), który wykorzystuje jednak pętlę typu `for`.

Listing 4.2. *Wykorzystanie pętli for do wyświetlenia zawartości tablicy*

```

<?php
$kolory = array("czerwony", "zielony", "niebieski");
for($i = 0; $i < 3; $i++){
    echo "kolory[$i] = $kolory[$i] <br />";
}
?>

```

Tablica może zostać również utworzona poprzez bezpośrednie przypisywanie wartości jej komórkom. Przykładowo zamiast pisać:

```
$kolory = array("czerwony", "zielony", "niebieski");
```

można wykorzystać serię instrukcji w postaci:

```
$kolory[0] = "czerwony";  
$kolory[1] = "zielony";  
$kolory[2] = "niebieski";
```

W ten sam sposób można również zmieniać zawartość poszczególnych komórek. Taką technikę zobrazowano w skrypcie przedstawionym na listingu 4.3.

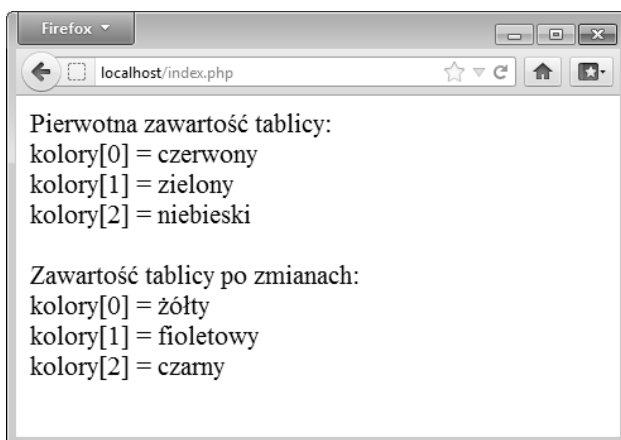
Listing 4.3. *Modyfikacja komórek tablicy*

```
<?php  
$kolory[0] = "czerwony";  
$kolory[1] = "zielony";  
$kolory[2] = "niebieski";  
  
echo "Pierwotna zawartość tablicy: <br />";  
for($i = 0; $i < 3; $i++){  
    $kolor = $kolory[$i];  
    echo "kolory[$i] = $kolor <br />";  
}  
  
$kolory[0] = "żółty";  
$kolory[1] = "fioletowy";  
$kolory[2] = "czarny";  
  
echo "<br />Zawartość tablicy po zmianach: <br />";  
for($i = 0; $i < 3; $i++){  
    $kolor = $kolory[$i];  
    echo "kolory[$i] = $kolor <br />";  
}  
?>
```

W pierwszej części skryptu powstała tablica `kolory`, której kolejnym indeksom przypisane zostały wartości `czerwony`, `zielony` i `niebieski`. Ścisłej rzecz ujmując, tablica powstała po wykonaniu instrukcji `$kolory[0] = "czerwony";`. Po napotkaniu tej instrukcji i stwierdzeniu, że w skrypcie nie ma tablicy o nazwie `kolory`, PHP tworzy ją, a następnie komórce o indeksie 0 przypisuje wartość z prawej strony operatora przypisania `=`. Następne dwie instrukcje to nic innego jak utworzenie kolejnych dwóch komórek w tablicy `kolory` i przypisanie im wskazanych wartości. Po przeprowadzeniu wymienionych operacji wykonywana jest pętla `for`, która wyświetla zawartość całej tablicy w przeglądarce.

Za pętlą znajduje się instrukcja `$kolory[0] = "żółty";`. Ponieważ istnieje już tablica `kolory`, instrukcja ta powoduje przypisanie komórce o indeksie 0 ciągu znaków `żółty`. W tym momencie zostaje również utracona poprzednia zawartość tej komórki, czyli ciąg `czerwony`. Podobnie działają dwie kolejne instrukcje. Zamieniają występujące w tablicy wartości z komórek 1 i 2 na ciągi znaków `fioletowy` i `czarny`. Po przeprowadzeniu tych operacji jest wykonywana druga pętla `for`, która wysyła do przeglądarki aktualną zawartość tablicy. Tym samym po wykonaniu skryptu na ekranie pojawi się widok zaprezentowany na rysunku 4.3.

Rysunek 4.3.
Ilustracja działania skryptu z listingu 4.3



Tablice asocjacyjne

Oprócz tablic indeksowanych numerycznie istnieją w PHP również **tablice asocjacyjne**. W tablicach tego typu każdemu indeksowi można nadać unikalną nazwę, czyli zamiast indeksów 0, 1, 2 itd. mogą występować indeksy: kolor, autor, procesor itp. Najczęściej też zamiast terminu indeks stosuje się inny termin, a mianowicie **klucz**. Mówimy zatem, że w tablicy asocjacyjnej występują pary **klucz – wartość**, w których każdy klucz jednoznacznie identyfikuje przypisaną mu wartość. Tablicę tego typu tworzy się (podobnie jak w przypadku tablic klasycznych indeksowanych numerycznie) za pomocą słowa kluczowego `array`, konstrukcja ta ma jednak nieco inną postać. Schematycznie wygląda to następująco:

```
array
(
    klucz1 => wartość1,
    klucz2 => wartość2,
    ...
    kluczn => wartośćn
);
```

Na listingu 4.4 został przedstawiony krótki skrypt, w którym pokazano, w jaki sposób utworzyć tablicę asocjacyjną i odczytać zapisane w niej wartości.

Listing 4.4. Utworzenie tablicy asocjacyjnej

```
<?php
$kolory = array
(
    "kolor1" => "czerwony",
    "kolor2" => "zielony",
    "kolor3" => "niebieski"
);
echo "Zawartość tablicy:<br />";
echo "kolory['kolor1'] = ";
echo $kolory['kolor1'];
```

```

echo "<br />kolory['kolor2'] = ";
echo $kolory['kolor2'];

echo "<br />kolory['kolor3'] = ";
echo $kolory['kolor3'];
?>

```

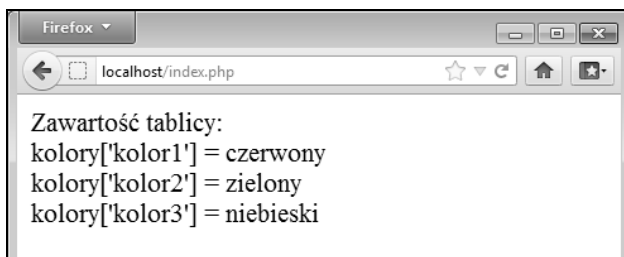
W skrypcie została utworzona tablica `$kolory`, która zawiera trzy klucze o nazwach: `kolor1`, `kolor2` i `kolor3`. Kluczowi `kolor1` został przypisany ciąg znaków `czerwony`, kluczowi `kolor2` — ciąg znaków `zielony`, a kluczowi `kolor3` — ciąg znaków `niebieski`. Dzięki temu po zastosowaniu konstrukcji w schematycznej postaci:

```
nazwa_tablicy['nazwa_klucza']
```

otrzymujemy wartość odpowiadającą danemu kluczowi. Ta konstrukcja została użyta do wyświetlenia zawartości poszczególnych kluczy tablicy w przeglądarce. Instrukcja `echo $kolory['kolor1'];` wyświetla zawartość klucza `kolor1`, instrukcja `echo $kolory['kolor2'];` — klucza `kolor2`, a instrukcja `echo $kolory['kolor3'];` — klucza `kolor3`. Tym samym na ekranie zobaczymy widok zaprezentowany na rysunku 4.4.

Rysunek 4.4.

Efekt działania skryptu z listingu 4.4



Drugim ze sposobów tworzenia tablicy asocjacyjnej jest użycie składni z nawiasem kwadratowym, podobnie jak miało to miejsce w przypadku tablic indeksowanych numerycznie. Schematycznie taka konstrukcja ma postać:

```
nazwa_tablicy['nazwa_klucza'] = wartość_klucza;
```

Na listingu 4.5 został przedstawiony skrypt, który realizuje takie samo zadanie jak skrypt 4.4, czyli utworzenie tablicy asocjacyjnej i wyświetlenie jej zawartości, ale wykorzystuje zaprezentowaną powyżej składnię.

Listing 4.5. Drugi sposób tworzenia tablic asocjacyjnych

```

<?php
$kolory['kolor1'] = "czerwony";
$kolory['kolor2'] = "zielony";
$kolory['kolor3'] = "niebieski";

echo "Zawartość tablicy:<br />";
echo "kolory['kolor1'] = ";
echo $kolory['kolor1'];

echo "<br />kolory['kolor2'] = ";
echo $kolory['kolor2'];

```

```

echo "<br />kolory['kolor3'] = ";
echo $kolory['kolor3'];
?>

```

Pierwsza instrukcja tego skryptu powoduje utworzenie tablicy asocjacyjnej `$kolory` oraz umieszczenie w niej klucza o nazwie `kolor1`, powiązane go z ciągiem znaków `czerwony`. Kolejne dwie instrukcje powodują umieszczenie w istniejącej już tablicy dwóch kolejnych kluczy: `kolor2` i `kolor3` oraz przypisanie do nich odpowiadających im wartości. Zawartość poszczególnych kluczy tak utworzonej tablicy jest następnie wysyłana do przeglądarki za pomocą serii instrukcji `echo`.

Do odczytu tablic asocjacyjnych można, podobnie jak w przypadku tablic klasycznych, użyć pętli. Nie może być to jednak pętla typu `for`, gdyż nie zdoła ona stwierdzić, jakie są wartości kluczy. Dlatego też tablice asocjacyjne są obsługiwane przez pętlę typu `foreach` (por. rozdział 3, sekcja „Pętla typu `foreach`”). Taka pętla potrafi pobrać kolejne wartości kluczy. Jak to zrobić, zobrazowano w skrypcie z listingu 4.6.

Listing 4.6. Wykorzystanie pętli typu `foreach`

```

<?php
$kolory['kolor1'] = "czerwony";
$kolory['kolor2'] = "zielony";
$kolory['kolor3'] = "niebieski";

echo "Zawartość tablicy:<br />";
foreach($kolory as $kolor){
    echo $kolor;
    echo "<br />";
}
?>

```

Konstrukcja tego typu pętli oznacza, że w każdym jej przebiegu pod zmienną `$kolor` będzie podstawiana wartość kolejnego klucza. A zatem zmienna `$kolor` w pierwszym przebiegu pętli będzie zawierała ciąg znaków `czerwony`, w drugim przebiegu — ciąg znaków `zielony`, a w trzecim przebiegu — ciąg znaków `niebieski`. W momencie gdy zostaną odczytane wartości wszystkich kluczy, pętla zakończy działanie. W ten sposób uzyskamy jednak jedynie wartości kluczy, nie zaś nazwy kluczy. Jeśli również ta informacja jest potrzebna, trzeba zastosować drugą wersję pętli `foreach`. Przykład tej konstrukcji został zaprezentowany na listingu 4.7.

Listing 4.7. Inna wersja pętli `foreach`

```

<?php
$kolory['kolor1'] = "czerwony";
$kolory['kolor2'] = "zielony";
$kolory['kolor3'] = "niebieski";

echo "Zawartość tablicy:<br />";
foreach($kolory as $klucz => $kolor){
    echo "kolory['$klucz'] = $kolor";
    echo "<br />";
}

```



```
}  
?>
```

Tym razem w każdym przebiegu pętli pod zmienną `$klucz` podstawiana jest nazwa kolejnego klucza, a pod zmienną `$kolor` — wartość przypisana temu kluczowi. Dzięki temu za pomocą instrukcji `echo` można wysłać do przeglądarki wszystkie istotne informacje o zawartości tablicy. Efekt działania kodu będzie taki sam jak skryptu z listingu 4.4 (rysunek 4.4).

Modyfikacji zawartości tablic asocjacyjnych dokonuje się tak samo jak zmian w przypadku tablic klasycznych. Oczywiście zamiast indeksów numerycznych trzeba zastępować wartości kluczy. Aby zatem przypisać nową wartość już istniejącemu kluczowi, trzeba skorzystać z konstrukcji, której schematyczna postać jest następująca:

```
nazwa_tablicy['nazwa_klucza'] = wartość;
```

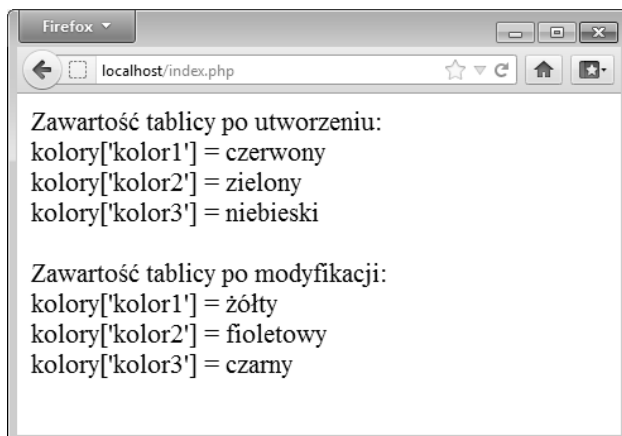
Na listingu 4.8 pokazany został przykładowy skrypt, który wykonuje modyfikację zawartości tablicy asocjacyjnej `$kolory`.

Listing 4.8. Modyfikacja zawartości tablicy asocjacyjnej

```
<?php  
$kolory['kolor1'] = "czerwony";  
$kolory['kolor2'] = "zielony";  
$kolory['kolor3'] = "niebieski";  
  
echo "Zawartość tablicy po utworzeniu:<br />";  
foreach($kolory as $klucz => $kolor){  
    echo "kolory['$klucz'] = $kolor";  
    echo "<br />";  
}  
  
$kolory['kolor1'] = "żółty";  
$kolory['kolor2'] = "fioletowy";  
$kolory['kolor3'] = "czarny";  
  
echo "<br />Zawartość tablicy po modyfikacji:<br />";  
foreach($kolory as $klucz => $kolor){  
    echo "kolory['$klucz'] = $kolor";  
    echo "<br />";  
}
```

Tablica `$kolory` jest tu tworzona analogicznie jak w poprzednim przykładzie. Tak samo jest również wyświetlana jej pierwotna zawartość. Kluczowi `kolor1` został przypisany ciąg znaków `czerwony`, kluczowi `kolor2` — ciąg znaków `zielony`, a kluczowi `kolor3` — ciąg znaków `niebieski`. Po wykonaniu pętli `foreach`, która wyświetla te dane na ekranie, wykonywana jest instrukcja `$kolory['kolor1'] = "żółty"`. Ponieważ tablica `$kolory` już istnieje i jest w niej zawarty klucz `kolor1`, następuje modyfikacja przypisanej do niego wartości z `czerwony` na `żółty`. Analogiczne operacje wykonywane są z wartościami kluczy `kolor2` i `kolor3`. Po wykonaniu tych modyfikacji zawartość tablicy jest ponownie wysyłana do przeglądarki. Tym samym po uruchomieniu skryptu na ekranie pojawi się widok zaprezentowany na rysunku 4.5.

Rysunek 4.5.
Zawartość tablicy
została
zmodyfikowana



Tablice wielowymiarowe

Do tej pory omawiane były tablice jednowymiarowe, czyli takie, które są wektorami elementów, o strukturze przedstawionej na rysunku 4.1. Aby odczytać dane z pojedynczej komórki, wystarczyło podać jej indeks lub w przypadku tablic asocjacyjnych — nazwę klucza. PHP umożliwia jednak budowanie bardziej skomplikowanych struktur — **tablic wielowymiarowych**. Przykładowa struktura prostej tablicy dwuwymiarowej została przedstawiona na rysunku 4.6. Jak widać, aby otrzymać wartość danej komórki, trzeba znać dwie liczby określające jej położenie: numer rzędu i numer kolumny. Na przykład komórka zawierająca wartość 8 znajduje się w rzędzie o indeksie 1 i kolumnie o indeksie 2.

Rysunek 4.6.
Struktura
przykładowej tablicy
dwuwymiarowej

	0	1	2	3	4
0	wartość 1	wartość 2	wartość 3	wartość 4	wartość 5
1	wartość 6	wartość 7	wartość 8	wartość 9	wartość 10

Tworzenie tablic wielowymiarowych

Do tworzenia tablic wielowymiarowych w PHP wykorzystuje się fakt, że pojedyncza komórka zwykłej tablicy jednowymiarowej może zawierać dane dowolnego typu, a zatem również inną tablicę. Wynika z tego, że tablica dwuwymiarowa to nic innego jak tablica jednowymiarowa, w której komórkach zawarte zostały inne tablice jednowymiarowe.

Spróbujmy wykonać prosty przykład. Na listingu 4.9 został zaprezentowany kod tworzący tablicę dwuwymiarową, w której komórkach zostały zawarte kolejne liczby od 1 do 6, a następnie wyświetlający jej zawartość na ekranie.

Listing 4.9. Tworzenie tablicy dwuwymiarowej

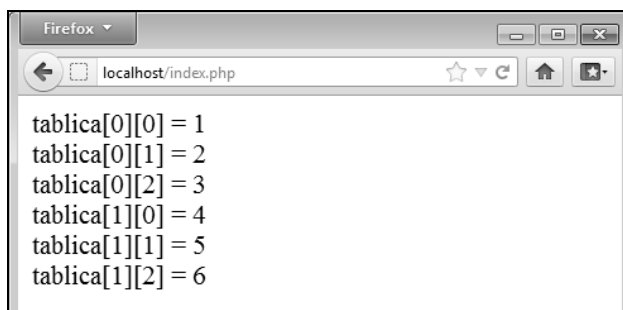
```
<?php
$tablica = array
(
    array(1, 2, 3),
    array(4, 5, 6)
);
echo "tablica[0][0] = " . $tablica[0][0] . "<br />";
echo "tablica[0][1] = " . $tablica[0][1] . "<br />";
echo "tablica[0][2] = " . $tablica[0][2] . "<br />";
echo "tablica[1][0] = " . $tablica[1][0] . "<br />";
echo "tablica[1][1] = " . $tablica[1][1] . "<br />";
echo "tablica[1][2] = " . $tablica[1][2] . "<br />";
?>
```

Konstrukcja tworząca tablicę `$tablica` dokładnie odzwierciedla sposób, w jaki ona powstaje. W pierwszej komórce (o indeksie 0) została umieszczona tablica trójelementowa zawierająca liczby 1, 2, 3, natomiast w komórce drugiej (o indeksie 1) została umieszczona tablica (również trójelementowa) zawierająca liczby 4, 5, 6. Powstała więc w ten sposób struktura o dwóch rzędach i trzech kolumnach. Dostęp do poszczególnych komórek wymaga zatem podania numeru wiersza i kolumny, co schematycznie wygląda następująco:

```
$tablica[wiersz][kolumna]
```

Ten sposób odwoływania się do komórek tablicy jest wykorzystywany (w instrukcjach `echo`) do wyświetlenia wszystkich zawartych w niej wartości w przeglądarce (rysunek 4.7).

Rysunek 4.7.
Wyświetlenie
zawartości tablicy
dwuwymiarowej



Do odczytu zawartości takiej tablicy można również wykorzystać dwie zagnieżdżone pętle `for`. Taki sposób jest szczególnie przydatny wówczas, gdy tablica ma dużą liczbę wierszy i kolumn. W kodzie z listingu 4.10 zobrazowano, jak wykonać takie zadanie dla tablicy powstałej w poprzednim przykładzie.

Listing 4.10. Wykorzystanie pętli *for* do odczytu tablicy

```
<?php
$tablica = array
(
    array(1, 2, 3),
    array(4, 5, 6)
);
for($i = 0; $i < 2; $i++){
    for($j = 0; $j < 3; $j++){
        $wart = $tablica[$i][$j];
        echo "tablica[$i][$j] = $wart";
        echo "<br />";
    }
    echo "<br />";
}
?>
```

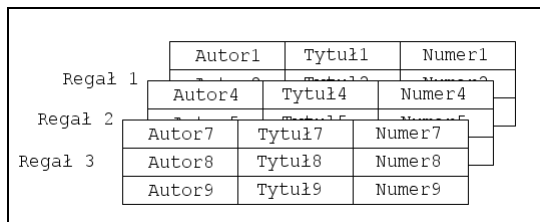
Zewnętrzna pętla *for* (ze zmienną iteracyjną *\$i*) kontroluje numer aktualnie odczytowanego wiersza tablicy, natomiast wewnętrzna pętla *for* (ze zmienną iteracyjną *\$j*) kontroluje numer aktualnie odczytywanej kolumny tablicy. Wartości kolejnych komórek są po odczytaniu zapisywane w zmiennej pomocniczej *\$wart*, która jest następnie wykorzystywana jako parametr w instrukcji *echo*.

Tablice wielowymiarowe nie muszą być indeksowane numerycznie — mogą być również strukturami asocjacyjnymi. Każdemu indeksowi można przypisać jego własną nazwę. Spróbujmy więc utworzyć i taką tablicę. Załóżmy, że mają być w niej przechowywane dane dotyczące książek w bibliotece. Pojedynczy wiersz będzie przechowywał dane dotyczące tytułu, autora i numeru katalogowego. Podstawowa tablica będzie mogła przechowywać wiele takich wierszy, a więc opisywać wiele książek. Jej struktura będzie zatem następująca:

```
$tablica = array(
    array("Autor" => "Autor1",
        "Tytuł" => "Tytuł1",
        "Numer" => "Numer1"),
    array("Autor" => "Autor2",
        "Tytuł" => "Tytuł2",
        "Numer" => "Numer2"),
    array("Autor" => "Autor3",
        "Tytuł" => "Tytuł3",
        "Numer" => "Numer3")
);
```

W ten sposób otrzymalibyśmy sam zbiór książek. Książki w bibliotece zazwyczaj stoją jednak na regałach. Można by więc wprowadzić dodatkową tablicę opisującą regały. Jej zawartością byłyby tablice opisujące książki. Powstałaby w ten sposób struktura trójwymiarowa, schematycznie przedstawiona na rysunku 4.8. Spróbujmy zbudować taką tablicę, wprowadzając do niej przykładowe dane, a następnie wyświetlimy jej zawartość na ekranie. To zadanie realizuje skrypt widoczny na listingu 4.11.

Rysunek 4.8.
Schematyczna
struktura tablicy
trójwymiarowej



Listing 4.11. Skrypt obsługujący tablicę trójwymiarową

```

<?php
$biblioteka = array(
    'regał1' => array
    (
        array("Autor" => "Marcin Lis",
            "Tytuł" => "PHP5. Praktyczny kurs",
            "Numer" => "123"),
        array("Autor" => "Marcin Lis",
            "Tytuł" => "Tworzenie stron WWW. Praktyczny kurs",
            "Numer" => "234"),
        array("Autor" => "Marcin Lis",
            "Tytuł" => "JavaScript. Praktyczny kurs",
            "Numer" => "345")
    ),
    'regał2' => array
    (
        array("Autor" => "Orson Scott Card",
            "Tytuł" => "Gra Endera",
            "Numer" => "321"),
        array("Autor" => "Orson Scott Card",
            "Tytuł" => "Cień Endera",
            "Numer" => "432"),
        array("Autor" => "Orson Scott Card",
            "Tytuł" => "Mistrz Pieśni",
            "Numer" => "543")
    ),
    'regał3' => array
    (
        array("Autor" => "Alex Kava",
            "Tytuł" => "Zło Konieczne",
            "Numer" => "213"),
        array("Autor" => "Kathy Reichs",
            "Tytuł" => "Pogrzebane Tajemnice",
            "Numer" => "324"),
        array("Autor" => "Harlan Coben",
            "Tytuł" => "Nie mów nikomu",
            "Numer" => "435")
    )
);

foreach($biblioteka as $regał_nazwa => $regał){
    echo "Regał: $regał_nazwa<br />";
    foreach($regał as $ksiazka){
        $autor = $ksiazka['Autor'];
        $tytuł = $ksiazka['Tytuł'];
    }
}

```

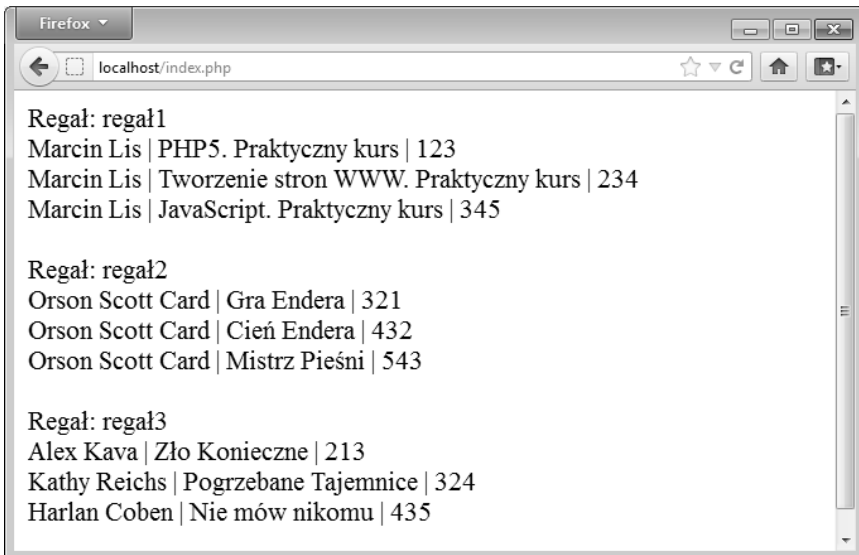
```

    $numer = $ksiazka['Numer'];
    echo "$autor | $tytuł | $numer";
    echo "<br />";
  }
  echo "<br />";
}

```

W kodzie została umieszczona główna tablica o nazwie `$biblioteka`. Zawiera ona trzy klucze o nazwach `regał1`, `regał2` i `regał3`. Pod każdym kluczem znajdują się kolejne tablice, które zawierają informacje opisujące książki w danym regale. Każda taka tablica składa się z serii tablic jednowymiarowych o kluczach `Autor`, `Tytuł` i `Numer`. Razem tworzy to pełny opis książek w bibliotece. Ponieważ ręczne pobieranie danych w celu wyświetlenia całej zawartości tablicy `$biblioteka` byłoby bardzo niewygodne i czasochłonne, do ich prezentacji zostały wykorzystane dwie zagnieżdżone pętle `foreach`.

Pętla zewnętrzna odczytuje zawartość kluczy tablicy głównej `$biblioteka`. Pod zmienną `$regal_nazwa` podstawiane są nazwy odczytanych kluczy, natomiast pod zmienną `$regal` — ich zawartość. Zawartością każdego klucza jest tablica zawierająca spis książek z danego regału, a zatem do jej odczytania wykorzystywana jest wewnętrzna pętla `foreach`. Pętla ta odczytuje zawartość kolejnych komórek tablicy `$regal`, podstawiając je pod zmienną `$ksiazka`. Zawartość tej zmiennej jest po prostu tablicą jednowymiarową, która opisuje pojedynczą książkę. Indeksami tej tablicy są więc: `Autor`, `Tytuł` i `Numer`. Dane te są odczytywane, zapisywane w zmiennych pomocniczych i wysyłane do przeglądarki za pomocą instrukcji `echo`. Ostatecznie na ekranie zobaczymy zawartość całej biblioteki z podziałem na regały, tak jak zostało to przedstawione na rysunku 4.9.



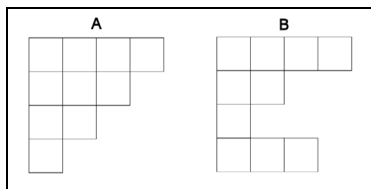
Rysunek 4.9. Efekt działania skryptu z listingu 4.11

Tablice nieregularne

Tablice wielowymiarowe wcale nie muszą mieć takich regularnie prostokątnych kształtów jak dotychczas prezentowane. Prostokątnych, to znaczy takich, gdzie w każdym wierszu znajduje się taka sama liczba komórek (czyli struktur podobnych do prezentowanej na rysunku 4.6). Nic nie stoi na przeszkodzie, aby stworzyć strukturę trójkątną (rysunek 4.10 A) lub też całkiem nieregularną (rysunek 4.10 B). Przygotowywanie takich struktur wymaga jednak więcej pracy niż w przypadku tablic regularnych, gdyż przeważnie każdy wiersz trzeba tu tworzyć oddzielnie.

Rysunek 4.10.

*Przykłady
nieregularnych tablic
wielowymiarowych*



Jak tworzyć tego typu struktury? Wiadomo już, że tablice wielowymiarowe to tak naprawdę tablice tablic jednowymiarowych. A zatem tablica dwuwymiarowa to tablica jednowymiarowa zawierająca szereg tablic jednowymiarowych, tablica trójwymiarowa to tablica jednowymiarowa zawierająca w sobie tablice dwuwymiarowe itd. Spróbujmy zatem stworzyć strukturę widoczną na rysunku 4.10 B, wypełnioną wartościami od 1 do 10, i wyświetlić jej zawartość w przeglądarce. To zadanie realizuje kod widoczny na listingu 4.12.

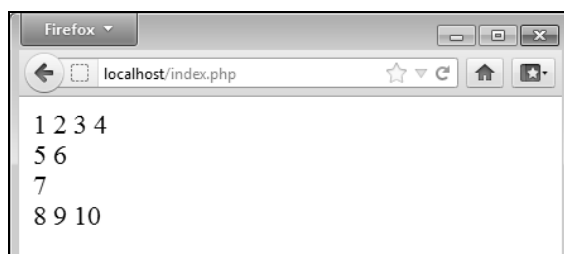
Listing 4.12. Tworzenie tablicy nieregularnej

```
<?php
$tablica = array
(
    array(1, 2, 3, 4),
    array(5, 6),
    array(7),
    array(8, 9, 10)
);
foreach($tablica as $tab){
    foreach($tab as $val){
        echo "$val ";
    }
    echo "<br />";
}
?>
```

Postać skryptu nie powinna być żadnym zaskoczeniem. Sposób tworzenia tablicy jest analogiczny do przedstawionego w poprzednich przykładach, z tą różnicą że tym razem tablice składowe mają różne wielkości. Pierwsza zawiera cztery komórki, druga — dwie, trzecia — jedną, a czwarta — trzy. Sposób odczytywania zawartości jest również podobny jak w przykładzie z listingu 4.11, a nawet nieco prostszy. Pętla zewnętrzna odczytuje kolejne komórki tablicy \$tablica. Każda z tych komórek zawiera kolejną

tablicę o pewnej liczbie elementów, które odczytywane są za pomocą wewnętrznej pętli `foreach`. Tym samym po uruchomieniu skryptu zobaczymy taki widok jak na rysunku 4.11.

Rysunek 4.11.
Zawartość tablicy
nieregularnej
z przykładu 4.12



Operacje na tablicach

Sortowanie tablic klasycznych

Jedną z operacji często wykonywanych na tablicach jest sortowanie, czyli ustawienie elementów w określonym porządku. PHP oferuje kilka wbudowanych funkcji sortujących. Zobaczmy, w jaki sposób można z nich korzystać. Funkcją podstawową jest `sort`. Działa ona zarówno na wartościach liczbowych, jak i na ciągach znaków. Jako argument jej wywołania należy podać nazwę tablicy. Spójrzmy na listing 4.13. Zawiera on kod sortujący dwie różne tablice.

Listing 4.13. Sortowanie za pomocą funkcji `sort`

```
<?php
$tab1 = array(5, 7, 3, 1, 8, 2, 0, 4, 9, 6);
$tab2 = array('jeden', 'dwa', 'trzy', 'cztery', 'pięć');

echo "Zawartość tablic przed sortowaniem: <br />";
foreach($tab1 as $val){
    echo "$val ";
}
echo "<br />";
foreach($tab2 as $val){
    echo "$val ";
}

sort($tab1);
sort($tab2);

echo "<br /><br />Zawartość tablic po sortowaniu: <br />";
foreach($tab1 as $val){
    echo "$val ";
}
echo "<br />";
foreach($tab2 as $val){
```



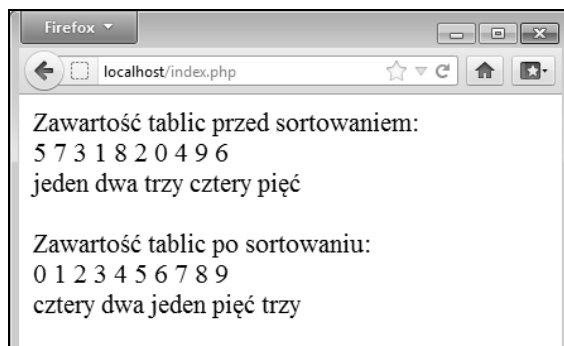
```

    echo "$val ";
}
?>

```

Efekt wykonania skryptu został przedstawiony na rysunku 4.12. Jak widać, obie tablice zostały poprawnie posortowane. Oczywiście w przypadku tablicy `$tab1` sortowane były liczby, więc wartości zostały ustawione od najmniejszej do największej, natomiast w przypadku tablicy `$tab2` sortowane były ciągi znaków, a zatem słowa zostały ustawione w porządku alfabetycznym. Co jednak zrobić w sytuacji, gdyby trzeba było wykonać sortowanie odwrotne, czyli np. ustawić wartości z tablicy `$tab1` od największej do najmniejszej? Nie ma z tym najmniejszego problemu. Wystarczy użyć funkcji `rsort` (z ang. *reverse sort*), która wykona to zadanie.

Rysunek 4.12.
Efekt sortowania
tablic



Więcej problemów przysporzy sytuacja, w której konieczne będzie ustawienie elementów tablicy w specyficznej kolejności, odmiennej od standardowego porządku. Do wyboru są wtedy dwie drogi. Trzeba albo samodzielnie napisać całą funkcję wykonującą sortowanie, albo też wykorzystać specjalną wersję funkcji sortującej — `usort` — w połączeniu z funkcją porównującą dwa elementy. Schematyczne wywołanie takiej funkcji ma postać:

```
usort($tablica, 'nazwa_funkcji')
```

gdzie `$tablica` to nazwa tablicy, której elementy będą sortowane, a `nazwa_funkcji` to nazwa funkcji dokonującej porównania dwóch elementów. Ta ostatnia funkcja będzie otrzymywała dwa elementy sortowanej tablicy w postaci argumentów, musi natomiast zwracać:

- ♦ wartość mniejszą od zera, jeśli pierwszy argument jest mniejszy od drugiego;
- ♦ wartość większą od zera, jeśli pierwszy argument jest większy od drugiego;
- ♦ wartość równą zero, jeśli pierwszy argument jest równy drugiemu.

Zobaczmy, jak to działa na konkretnym przykładzie. Powstanie skrypt, który zawartość tablicy przechowującej liczby całkowite będzie sortował w taki sposób, że najpierw umieszczone będą wartości podzielne przez dwa (od najmniejszej do największej), a dopiero po nich wartości niepodzielne przez dwa — również od najmniejszej do największej. To zadanie realizuje kod z listingu 4.14.

Listing 4.14. Realizacja niestandardowego sortowania

```
<?php
function sortuj($e1, $e2)
{
    if($e1 % 2 == 0){
        if($e2 % 2 == 0){
            return $e1 - $e2;
        }
        else{
            return -1;
        }
    }
    else{
        if($e2 % 2 == 0){
            return 1;
        }
        else{
            return $e1 - $e2;
        }
    }
}

$tab1 = array(5, 7, 3, 1, 8, 2, 0, 4, 9, 6);

echo "Zawartość tablicy przed sortowaniem: <br />";
foreach($tab1 as $val){
    echo("$val ");
}
echo "<br />";

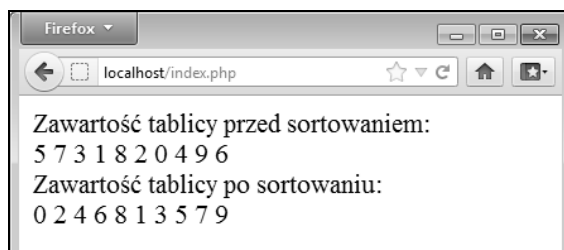
usort($tab1, 'sortuj');

echo "Zawartość tablicy po sortowaniu: <br />";
foreach($tab1 as $val){
    echo "$val ";
}
?>
```

Tablica jest tworzona w standardowy sposób i jej początkowa zawartość jest wyświetlana na ekranie. Następnie wywoływana jest funkcja `usort`, która wykonuje operację sortowania, a potem zawartość posortowanej tablicy jest ponownie wyświetlana na ekranie. Tym samym w przeglądarce ukaże się obraz widoczny na rysunku 4.13. Jak widać, układ liczb jest zgodny z założeniami — najpierw umieszczone są liczby podzielne przez dwa (od najmniejszej do największej), a za nimi liczby niepodzielne przez dwa (również od najmniejszej do największej). Za takie uporządkowanie elementów tablicy odpowiada kombinacja funkcji `usort` i `sortuj`.

Funkcja `usort` realizuje algorytm sortowania typu QuickSort. Ponieważ sortowanie ma odbywać się według niestandardowych zasad, trzeba tej funkcji dostarczyć dodatkową funkcję, która będzie porównywała dwa dowolne elementy tablicy `tab1`. Tą funkcją jest `sortuj`. Przy porównywaniu dwóch dowolnych elementów tablicy `tab1` możliwe są cztery różne sytuacje:

Rysunek 4.13.
Efekt
niestandardowego
sortowania



1. Pierwszy argument jest podzielny przez dwa ($e1 \% 2$ równe 0) i drugi argument jest również podzielny przez dwa ($e2 \% 2$ równe 0). W takiej sytuacji należy zwrócić wartość mniejszą od zera, jeśli pierwszy argument jest mniejszy; wartość większą od zera, jeśli drugi argument jest mniejszy; lub wartość 0, jeśli argumenty są równe. Zapewnia to instrukcja `return $e1 - $e2;`.
2. Pierwszy argument jest podzielny przez dwa ($e1 \% 2$ równe 0), natomiast drugi argument nie jest podzielny przez dwa ($e2 \% 2$ różne od 0). W takiej sytuacji argument pierwszy zawsze powinien znaleźć się przed argumentem drugim, a zatem należy zwrócić wartość mniejszą od zera. Zapewnia to instrukcja `return -1;`.
3. Pierwszy argument nie jest podzielny przez dwa ($e1 \% 2$ różne od 0), a drugi argument jest podzielny przez dwa ($e2 \% 2$ równe 0). W takiej sytuacji argument pierwszy zawsze powinien znaleźć się za argumentem drugim, a zatem należy zwrócić wartość większą od zera. Zapewnia to instrukcja `return 1;`.
4. Pierwszy argument nie jest podzielny przez dwa ($e1 \% 2$ różne od 0) i drugi argument również nie jest podzielny przez dwa ($e2 \% 2$ różne od 0). W takiej sytuacji należy zwrócić wartość mniejszą od zera, jeśli pierwszy argument jest mniejszy; wartość większą od zera, jeśli drugi argument jest mniejszy; oraz wartość 0, jeśli argumenty są równe. Zapewnia to instrukcja `return $e1 - $e2;`.

Sortowanie tablic asocjacyjnych

W przypadku tablic asocjacyjnych nie można użyć zwykłej funkcji `sort`, gdyż spowoduje ona utratę kluczy. Łatwo się o tym przekonać, uruchamiając skrypt widoczny na listingu 4.15. Została w nim utworzona tablica `$tab` zawierająca cztery klucze z przypisanymi wartościami całkowitymi. Tablica ta została następnie posortowana za pomocą funkcji `sort`. Zawartość przed sortowaniem i po nim została wyświetlona za pomocą pętli `foreach` i instrukcji `echo`. Jak widać na rysunku 4.14, efekt takiego działania nie jest zadowalający. Co prawda wartości zostały posortowane, ale jednocześnie zostały utracone nazwy indeksów.

Listing 4.15. *Użycie funkcji `sort` do sortowania tablicy asocjacyjnej*

```
<?php
$tab = array
(
    'indeks1' => 5,
    'indeks9' => 1,
```

```

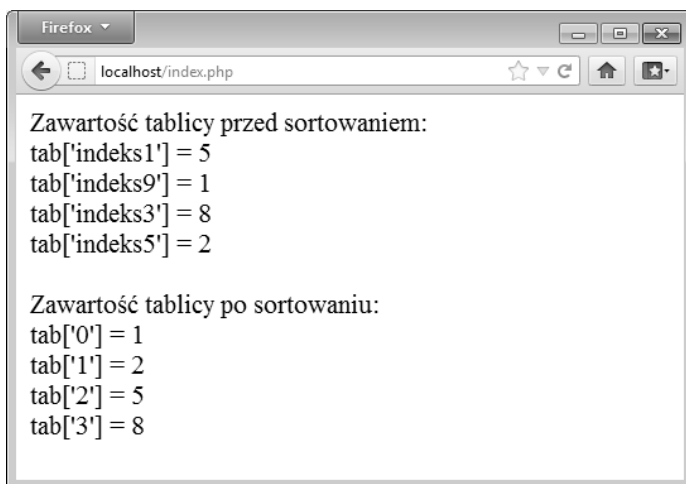
    'indeks3' => 8,
    'indeks5' => 2
);
echo "Zawartość tablicy przed sortowaniem:<br />";
foreach($tab as $key => $val){
    echo "tab['$key'] = $val";
    echo "<br />";
}

sort($tab);

echo "<br />Zawartość tablicy po sortowaniu:<br />";
foreach($tab as $key => $val){
    echo "tab['$key'] = $val";
    echo "<br />";
}
?>

```

Rysunek 4.14.
*Utrata nazw indeksów
 po nieprawidłowym
 sortowaniu tablicy
 asocjacyjnej*



Aby zatem posortować tablicę asocjacyjną, trzeba użyć innych funkcji, a mianowicie: `asort` i `ksort`. Pierwsza z nich sortuje tablicę względem wartości poszczególnych kluczy, natomiast druga względem samych kluczy. Oznacza to, że jeśli w skrypcie z listingu 4.15 zamieni się funkcję `sort` na `asort`, po sortowaniu kolejność będzie następująca:

```

tab['indeks9'] = 1
tab['indeks5'] = 2
tab['indeks1'] = 5
tab['indeks3'] = 8

```

Natomiast po zamianie funkcji `sort` na `ksort` uzyska się wynik:

```

tab['indeks1'] = 5
tab['indeks3'] = 8
tab['indeks5'] = 2
tab['indeks9'] = 1

```

Sortowanie może się odbywać również w porządku odwrotnym, czyli od wartości największej do najmniejszej. Służą do tego celu funkcje `arsort` (sortowanie względem wartości) i `krsort` (sortowanie względem kluczy).

Implodzja i eksplozja

Bardzo ciekawymi i (jak się okaże przy realizacji praktycznych projektów) użytecznymi funkcjami są `implode` i `explode` (stąd też powyższy podtytuł). Pierwsza powoduje zwrócenie wszystkich elementów tablicy rozdzielonych znakami separatora jako ciągu znaków. Wywołanie ma schematyczną postać:

```
implode(separator, tablica)
```

Jeśli na przykład istnieje tablica `$arr` o postaci:

```
$arr = array('jeden', 'dwa', 'trzy');
```

to wykonanie instrukcji:

```
echo implode(' ', $arr);
```

spowoduje wyświetlenie ciągu znaków:

```
jeden, dwa, trzy
```

Podobnie, efektem działania instrukcji:

```
echo implode('_', $arr);
```

będzie ciąg:

```
jeden_dwa_trzy
```

Funkcja `explode` (jak sama nazwa wskazuje) działa odwrotnie niż `implode`. Tworzy więc tablicę składającą się z fragmentów ciągu znaków wydzielonych przez znaki separatora. Jej wywołanie ma ogólną postać:

```
explode(separator, ciąg[, ile]);
```

Jeśli na przykład istnieje ciąg `$str` w postaci:

```
$str = 'jeden, dwa, trzy, cztery';
```

to wykonanie instrukcji:

```
$arr = explode(' ', $str);
```

spowoduje powstanie tablicy o następujących elementach:

```
Array
(
    [0] => jeden
    [1] => dwa
    [2] => trzy
    [3] => cztery
)
```

Gdy opcjonalny argument `ile` ma wartość dodatnią, wskazuje maksymalną liczbę elementów tablicy wynikowej. Ostatni element będzie zawierał pozostałą część ciągu. Jeżeli zatem istnieje taki ciąg `$str` jak podany wyżej, efektem wykonania instrukcji:

```
$arr = explode(' ', $str, 3);
```

będzie tablica:

```
Array
(
    [0] => jeden
    [1] => dwa
    [2] => trzy, cztery
)
```

Jeśli argument `ile` ma wartość ujemną, określa, ile elementów należy usunąć z końca tablicy wynikowej. Przykładowo dla zdefiniowanego wyżej ciągu `$str` efektem działania instrukcji:

```
$arr = explode(' ', $str, -2);
```

będzie tablica:

```
Array
(
    [0] => jeden
    [1] => dwa
)
```

Operacje na elementach tablic

Zmiana kolejności elementów

Jeśli chcemy odwrócić kolejność elementów w tablicy, czyli spowodować, aby pierwszy stał się ostatnim, drugi przedostatnim itd., możemy zastosować funkcję `array_reverse`. Jako argument tej funkcji należy przekazać nazwę tablicy. Tablica ze zmienioną kolejnością elementów zostanie zwrócona jako wynik działania funkcji, a zawartość oryginalnej tablicy nie zostanie naruszona. Sposób działania funkcji `array_reverse` zobrazowano w przykładzie widocznym na listingu 4.16.

Listing 4.16. *Odwrócenie kolejności elementów tablicy*

```
<?php
$tab1 = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
$tab2 = array_reverse($tab1);

echo "Zawartość tablicy tab1:<br />";
foreach($tab1 as $val){
    echo("$val ");
}
echo "<br />Zawartość tablicy tab2:<br />";
foreach($tab2 as $val){
```

```
    echo "$val ";
}
?>
```

Poruszanie się po tablicy

Każda tablica w PHP ma wewnętrzny wskaźnik wskazujący jej bieżący element. Po utworzeniu tablicy wskaźnik ten jest ustawiony na pierwszy element. Podczas wykonywania operacji na elementach tablicy jego położenie może się zmieniać. Istnieją funkcje, które wykorzystują go do własnych potrzeb, istnieje również możliwość bezpośredniej manipulacji pozycją wskaźnika. Jedną z takich funkcji jest `each`. Jej zadaniem jest pobranie aktualnego elementu tablicy i przesunięcie wskaźnika o jedno miejsce w przód. Jeżeli wskaźnik znajdzie się na końcu tablicy, wywołanie `each` powoduje zwrócenie wartości `false`. Takie działanie funkcji `each` umożliwia zastosowanie jej w pętli `while` przetwarzającej elementy tablicy. Należy jedynie pamiętać, że wynikiem działania `each` jest w rzeczywistości czteroelementowa tablica (!) zawierająca cztery klucze: `0`, `1`, `key`, `value`, gdzie `0` i `key` przechowują pobrany klucz (indeks), a `1` i `value` odpowiadają mu wartość. Zobrazowano to w przykładzie z listingu 4.17.

Listing 4.17. Użycie funkcji `each` do odczytu elementów tablicy

```
<?php
$tab = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
while($val = each($tab)){
    echo "val[0] = $val[0] | ";
    echo "val[1] = $val[1] | ";
    echo "val['key'] = $val[key] | ";
    echo "val['value'] = $val[value] ";
    echo "<br />";
}
?>
```

Funkcje, które pozwalają na bezpośrednią modyfikację wewnętrznego wskaźnika tablicy, to:

- ♦ `reset` — resetuje wskaźnik tablicy, ustawiając go na pierwszym elemencie. Funkcja jednocześnie zwraca wartość pierwszego elementu.
- ♦ `next` — przesuwa wskaźnik tablicy na następny element i zwraca wartość tego elementu. Jeśli aktualną pozycją wskaźnika tablicy jest jej ostatni element, funkcja zwraca wartość `false`.
- ♦ `prev` — przesuwa wskaźnik tablicy na poprzedni element (w stosunku do pozycji bieżącej) i zwraca wartość tego elementu. Jeśli aktualną pozycją wskaźnika tablicy jest jej pierwszy element, funkcja zwraca wartość `false`.
- ♦ `end` — ustawia wskaźnik tablicy na jej ostatnim elemencie i zwraca wartość tego elementu.

Oprócz wymienionych wyżej funkcji modyfikujących wewnętrzny wskaźnik istnieją również dwie funkcje, które pobierają aktualny element tablicy. Są to: `current` i `pos`.

Przykłady wykorzystania tego typu konstrukcji języka zostały zaprezentowane na listingu 4.18. Efekt działania skryptu jest natomiast przedstawiony na rysunku 4.15.

Listing 4.18. Wykorzystanie funkcji operujących na wewnętrznym wskaźniku tablicy

```
<?php
$tab = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
$val = end($tab);

echo "Wynik działania end(\$tab): $val<br />";

prev($tab);
prev($tab);

$val = current($tab);
echo "Po dwukrotnym wykonaniu prev(\$tab) ";
echo "aktualnym elementem jest: $val<br />";

$val = reset($tab);
echo "Po wykonaniu reset(\$tab) aktualnym elementem jest: $val<br />";

next($tab);
next($tab);

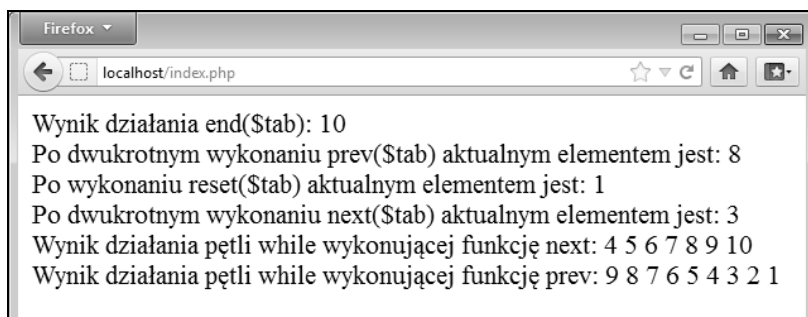
$val = current($tab);
echo "Po dwukrotnym wykonaniu next(\$tab) ";
echo "aktualnym elementem jest: $val<br />";

echo "Wynik działania pętli while wykonującej funkcję next: ";
while($val = next($tab)){
    echo "$val ";
}

end($tab);

echo "<br />Wynik działania pętli while wykonującej funkcję prev: ";
while($val = prev($tab)){
    echo "$val ";
}
?>
```

Rysunek 4.15.
Efekt działania skryptu wykorzystującego funkcje operujące na wskaźniku tablicy



Na początku skryptu powstała tablica `$tab` zawierająca 10 kolejnych liczb całkowitych. Następnie została wykonana operacja `end($tab)`, a jej wynik — przypisany zmiennej `$val`. Wartością tej zmiennej stała się więc wartość znajdująca się w ostatniej komórce tablicy, czyli 10. W kolejnym kroku zostały wykonane dwie operacje `prev($tab)`, co oznacza, że wewnętrzny wskaźnik tablicy został przesunięty o dwie pozycje do tyłu. Przekonujemy się o tym, pobierając aktualny element tablicy (`$val = current($tab);`) i wyświetlając go w przeglądarce za pomocą instrukcji `echo`.

Kolejny krok to wykonanie funkcji `reset`, która przesuwa wskaźnik na początek tablicy (a zatem aktualnym elementem staje się komórka o indeksie 0). Po wykonaniu funkcji `reset` dwukrotnie wykonywana jest funkcja `next`, czyli wskaźnik jest przesuwany o dwie pozycje do przodu i wskazuje na trzeci element (o indeksie 2). Dalej w kodzie została umieszczona pętla `while` przeglądająca kolejne elementy tablicy. Wykorzystuje ona fakt, że funkcja `next` przesuwa wskaźnik o jedno miejsce i zwraca wartość wskazanego elementu. W przypadku gdy wskaźnik zostanie przesunięty za ostatni element, funkcja zwraca wartość `false`, co jest warunkiem zakończenia pętli.

Ponieważ po ostatnim wykonaniu funkcji `next` wskaźnik tablicy został przesunięty za ostatni element, po zakończeniu pętli jest wykonywana funkcja `end`, która przesuwa go z powrotem na ostatni element. Dzięki temu może poprawnie zadziałać kolejna pętla `while`, która wykonuje serię funkcji `prev`, przesuujących wskaźnik tablicy do tyłu, za każdym wywołaniem o jedną pozycję. Gdy wskaźnik znajdzie się przed pierwszym elementem, wywołanie funkcji `prev` zwróci wartość `false` i tym samym pętla zakończy działanie.

Dodawanie i pobieranie elementów

W PHP istnieją wbudowane funkcje, które pozwalają na dodawanie i usuwanie elementów z początku i z końca tablicy. Są to: `array_pop`, `array_shift`, `array_put` i `array_unshift`. Funkcja `array_pop` pobiera element znajdujący się na końcu tablicy i zwraca jego wartość. Tym samym tablica zostaje skrócona o ostatni element. Schematycznie operacja taka ma postać:

```
$zmienna = array_pop($tablica);
```

Podobne zadanie wykonuje `array_shift`, ale z tą różnicą że usuwany jest pierwszy element. Jeżeli tablica była indeksowana numerycznie, wówczas wszystkie elementy zostaną przenumerowane, czyli indeks każdego z nich zmniejszy się o jeden.

Funkcja `array_push` działa odwrotnie niż `array_pop`. Dodaje ona elementy przekazane w postaci argumentów na końcu tablicy. Schematycznie operację tę można przedstawić jako:

```
array_push($tablica, element1, element2, ..., elementN);
```

Funkcja zwraca wartość określającą liczbę elementów w powiększonej tablicy. Podobnie jak `array_push` działa `array_unshift` — dodaje ona określoną liczbę elementów na początku tablicy. Jeśli tablica była indeksowana numerycznie, zostanie ona również odpowiednio przenumerowana. Wywołanie funkcji `array_unshift` ma schematyczną postać:

```
array_unshift($tablica, element1, element2, ..., elementN);
```

Sposób wykorzystania wymienionych funkcji w działającym skrypcie zobrazowano w kodzie widocznym na listingu 4.19. Efekt jego działania został natomiast zaprezentowany na rysunku 4.16.

Listing 4.19. *Ilustracja działania funkcji modyfikujących zawartość tablicy*

```
<?php
$tab = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
echo "Pierwotna zawartość tablicy: ";
foreach($tab as $val){
    echo "$val ";
}

$val = array_pop($tab);
echo("<br />Wynik pierwszej operacji pop: $val <br />");
$val = array_pop($tab);
echo "Wynik drugiej operacji pop: $val <br />";

echo "Aktualna zawartość tablicy: ";
foreach($tab as $val){
    echo "$val ";
}

$val = array_shift($tab);
echo "<br />Wynik pierwszej operacji shift: $val <br />";
$val = array_shift($tab);
echo "Wynik drugiej operacji shift: $val <br />";

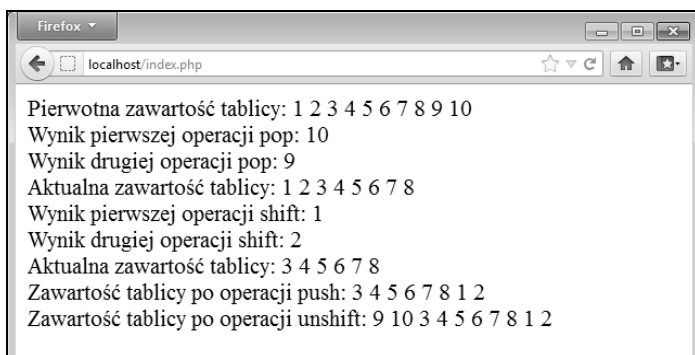
echo "Aktualna zawartość tablicy: ";
foreach($tab as $val){
    echo "$val ";
}

array_push($tab, 1, 2);
echo "<br />Zawartość tablicy po operacji push: ";
foreach($tab as $val){
    echo "$val ";
}

array_unshift($tab, 9, 10);
echo "<br />Zawartość tablicy po operacji unshift: ";
foreach($tab as $val){
    echo "$val ";
}
?>
```

W skrypcie tworzona jest tablica \$tab, która początkowo zawiera uporządkowane rosnąco wartości od 1 do 10. Wykonanie dwóch operacji array_pop(\$tab); powoduje usunięcie dwóch ostatnich wartości, a zatem pozostają komórki z wartościami od 1 do 8. Następnie są wykonywane dwie operacje array_shift(\$tab);, które usuwają dwie pierwsze komórki; tym samym w tablicy pozostają wartości od 3 do 8. Należy zwrócić uwagę, że przenumerowaniu uległy również indeksy komórek. Wartość 3 znajduje się

Rysunek 4.16.
Efekt działania
skryptu z listingu 4.19



obecnie pod indeksem 0, wartość 4 pod indeksem 1 itd. Kolejną wykonywaną operacją jest `array_push($tab, 1, 2)`, która powoduje dodanie na końcu tablicy dwóch komórek, pierwszej o wartości 1 i drugiej o wartości 2. Operacja `array_unshift($tab, 9, 10)`; powoduje natomiast dodanie na początku tablicy dwóch komórek, pierwszej o wartości 9 i drugiej o wartości 10. Ostatecznie tablica zawiera zatem ciąg wartości 9, 10, 3, 4, 5, 6, 7, 8, 1, 2, tak jak jest to widoczne na rysunku 4.16.

Liczba elementów tablicy

W wielu przypadkach bardzo przydaje się ustalenie rozmiaru tablicy, czyli stwierdzenie, ile zawiera ona elementów. W PHP wykorzystywana jest w tym celu funkcja `count` (zamiast `count` można również użyć `sizeof`, która jest aliasem dla `count`). Jeśli zatem zostanie wykonany kod:

```
$tab = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
$rozmiar = count($tab);
```

w zmiennej `$rozmiar` zostanie zapisana wartość 10.


Nieco bardziej złożoną czynnością jest stwierdzenie, ile razy dana wartość występuje w tablicy. W PHP istnieje specjalna funkcja, która wykonuje to zadanie: `array_count_values`. Zwraca ona tablicę asocjacyjną, której kluczami są wartości tablicy oryginalnej, natomiast wartościami kluczy jest liczba wystąpień tych wartości w tablicy oryginalnej. Najlepiej zobaczyć, jak to działa, zapoznając się z konkretnym przykładem przedstawionym na listingu 4.20. Efekt jego działania został zaprezentowany na rysunku 4.17.

Listing 4.20. Ilustracja działania funkcji `array_count_values`

```

<?php
$tab = array(1, 2, 5, 1, 5, 1, 5, 1, 8, 2);
$values = array_count_values($tab);
foreach($values as $key => $val){
    echo "[$key] => $val <br />";
}
?>
  
```

Rysunek 4.17.
*Efekt działania funkcji
array_count_values*



```
[1] => 4
[2] => 2
[5] => 3
[8] => 1
```

W skrypcie została utworzona tablica \$tab zawierająca zbiór liczb. Wykonanie funkcji array_count_values spowodowało zwrócenie tablicy asocjacyjnej, której zawartość została wyświetlona w przeglądarce. Widać wyraźnie, że wartość 1 w oryginalnej tablicy występuje cztery razy, wartość 2 — dwa razy, wartość 5 — trzy razy, a wartość 8 — jeden raz.

Skorowidz

A

adres
 IP, 491
 lokalny, 21, 170
agregacja danych, 363
aktualizacja zawartości koszyka, 636
algorytm QuickSort, 132
aplikacja ApacheMonitor, 20
argumenty
 domyślne, 113
 funkcji, 105
 konstruktorów, 152
atrybut
 AUTO_INCREMENT, 332
 INDEX, 332
 NOT NULL, 332
 PRIMARY KEY, 332
atrybuty kolumny, 332
autoryzacja, 455

B

białe znaki, 195
blok
 case, 596, 637
 switch, 596
blokada zapisu do pliku, 249
błąd
 krytyczny, fatal error, 163
 logowania, LOGIN_FAILED, 503
 serwera, SERVER_ERROR, 503
błędny identyfikator, 539
błędy logiczne skryptu, 166

C

ciąg formatujący, 201
ciągi znaków, 193

cookies, 265
CSS, 482
czas ważności cookie, 268
część administracyjna serwisu, 517, 595

D

dane z formularza rejestracyjnego, 618
definicja formularza, 169
deklaracja tablicy, 118
deskryptor, 236
destruktory, 153
diagram tabel i relacji, 391, 393
dodawanie
 elementów do tablicy, 139
 książek do koszyka, 630
 nowych użytkowników, 535
 rekordów do bazy, 443
 wiadomości, 571
 wiersza danych, 443, 446
domyślne kodowanie znaków, 449
dostęp do
 części administracyjnej, 595
 danych z formularza, 171
 składowych, 158, 161
 chronionych, 161
 klasy, 146
 prywatnych, 161
 publicznych, 161
dziedziczenie, 153, 154
dzielenie ciągów znaków, 210

E

edycja
 konta użytkownika, 537
 koszyka, 634
 użytkowników, 544
elementy formularza, 170

F

- filtrowanie danych, 619
- formatowanie ciągu, 199
- formularz, 170
 - ankiety, 254
 - do dodawania danych, 573
 - do edycji danych, 572
 - do odszukiwania wiadomości, 579
 - HTML, 439
 - logowania, 260, 282, 468, 610
 - rejestracyjny, 469, 614
 - umożliwiający oddanie głosu, 255
 - wyboru zakresu, 505
 - wysłający opinie, 244
 - wyszukiwania książek, 621
- funkcja
 - addStatRecord, 490
 - addToDBTable, 441
 - argumenty, 441
 - array_count_values, 141
 - array_pop, 139
 - array_push, 139
 - array_reverse, 136
 - array_shift, 139
 - array_unshift, 139
 - arsort, 135
 - chdir, 228
 - checkdate, 212
 - checkFileName, 184
 - checkPass, 261, 281, 457, 463
 - closedir, 223
 - count, 141
 - crypt, 460
 - current, 137
 - Date, 213
 - DATE_SUB, 492
 - delDir, 231
 - disk_free_space, 230
 - disk_total_space, 230
 - end, 137
 - explode, 135, 210
 - feof, 241
 - fgetc, 240
 - fgets, 237, 239
 - file_exists, 184, 228
 - file_get_contents, 242, 245
 - file_put_contents, 243
 - filesize, 180, 229, 230
 - filter_input, 577, 619
 - flock, 248
 - fopen, 235, 236
 - fpass thru, 242
 - fread, 180, 241
 - fseek, 247
 - ftell, 247
 - func_num_args, 115
 - fwrite, 243
 - get_browser, 489
 - getAllUsersOnline, 493
 - getCounter, 250
 - getcwd, 228
 - getdate, 215
 - getDirSize, 230
 - getQueryResultAsTableRows, 486
 - getRegUsersOnline, 492
 - getShortStats, 494
 - gettype, 51
 - gmdate, 216
 - header, 180, 463
 - implode, 135, 210
 - in_array, 184
 - initDB, 485
 - intval, 74, 441
 - is_file, 229
 - isset, 172, 184
 - ksort, 135
 - LIKE, 348
 - listDir, 233
 - localtime, 217
 - ltrim, 196
 - microtime, 218
 - mkdir, 227
 - mktime, 218
 - move_uploaded_file, 178, 179
 - mysql_query, 422
 - mysqli_fetch_all, 422
 - mysqli_fetch_array, 422
 - mysqli_fetch_assoc, 422
 - mysqli_character_set_name, 451
 - mysqli_connect, 415, 417
 - mysqli_fetch_field, 422
 - mysqli_fetch_field_direct, 422
 - mysqli_fetch_fields, 422
 - mysqli_fetch_object, 422
 - mysqli_fetch_row, 422
 - mysqli_free_result, 422
 - mysqli_query, 437
 - funkcja next, 137
 - nl2br, 197, 238
 - NOT LIKE, 348
 - opendir, 223, 234
 - pos, 137
 - prev, 137
 - print, 199
 - printf, 199, 204
 - printList, 189
 - readdir, 185, 223

readfile, 242
rejestruj, 475
reset, 137
rewind, 248
rmdir, 228
rtrim, 196
scandir, 225
send, 190
session_destroy, 273
session_start, 272, 463, 474
setcookie, 265
 argumenty, 265
setlocale, 197
settype, 74
show, 256
sort, 130, 133
sortuj, 132
sprintf, 228, 258
str_replace, 208
str_replace, 208, 234
strncasecmp, 204
strncmp, 204
strncasecmp, 204
strncmp, 204
strpos, 206
stripos, 206
strlen, 457
strnatcasecmp, 204
strnatcmp, 204
strncasecmp, 204
strncmp, 204
strpos, 206
stripos, 206
strtok, 210
strtotime, 221
strtr, 208
substr, 211
substr_replace, 210
SUM, 409
time, 221
trim, 196
urlencode, 234
usersAdmin, 530
usort, 131, 205
utf8_decode, 458
vote, 258
funkcje, 104
 agregujące, 364
 agregujące w złączeniach, 370
 argumenty, 105
 kontrolujące typ zmiennych, 50
 konwersji, 73
 odczytujące dane, 422
 przeszukujące ciągi, 206
 sklepu, 604
 skrótów, 459

statystyczne, 367
zamieniające podciągi, 208
zmieniające wielkość liter, 198
zwracanie wartości, 107

G

generowanie
 listy odnośników, 263
 listy plików, 185
 statystyk, 505
główna część serwisu, 495
GMT, Greenwich Mean Time, 216
graficzny licznik odwiedzin, 252
grupowanie, 363
 danych, 372
 wyników zapytań, 367

H

hash, 459
hierarchia wyjątków, 167
historia odwiedzin, 489

I

identyfikacja
 przeładowarki, 488
 rekordów, 308
identyfikator
 książki, 642
 połączenia z bazą, 452
 użytkownika, 490, 575, 579
 wartsw, 482
 wpisu, 491
 zamówienia, 642, 644
 zasobów, 422
IIS, Internet Information Services, 17
iloczyn logiczny, 62
implementacja sesji, 276
indeksy, 374, 402
informacje
 atomowe, 306
 nadmiarowe, 305
 o pliku, 177, 180, 228
 o zamówieniu, 643
 statystyczne, 373
instalacja
 MySQL, 290
 PHP, 17, 23, 25
 serwera Apache, 20
instancja, 145
instrukcja
 ALTER TABLE, 379
 break, 88, 101

instrukcja

continue, 103
 CREATE, 339
 CREATE INDEX, 376
 CREATE TABLE, 331, 341
 DELETE, 353, 385
 echo, 148, 199
 globals, 110
 if, 98
 if...else, 77, 99
 if...else if, 99
 include, 36, 457
 INSERT, 338, 385
 INSERT INTO, 337
 new Exception(), 165
 parent::show(), 158
 REPLACE, 354
 REPLACE INTO, 594
 require, 37, 148
 return, 107
 SELECT, 341, 342
 switch, 85, 87, 88, 99, 520, 589
 throw, 163
 try...catch, 164
 tworząca tabelę, 383, 480

- Autorzy, 395
- AutorzyPseudonimy, 400
- Klienci, 396
- Książki, 394
- KsiążkiAutorzy, 395
- KsiążkiZamowienia, 398
- Opinie, 398
- Recenzje, 399
- Wydawnictwa, 396
- Zamowienia, 397

 UPDATE, 351, 385

instrukcje

warunkowe, 77, 98
 warunkowe zagnieżdżone, 80

integracja koszyka ze sklepem, 637

interakcja serwera WWW i przeglądarki, 31

interfejs

obiektowy, 418, 430
 obiektowy mysqli, 443
 PDO, 447
 proceduralny, 415

J

język

DCL, 324
 DDL, 323
 DML, 323
 SEQUEL, 323

K

katalog php/tmp, 235

klasa, 144

Basket, 629
 Exception, 163
 LogicException, 166
 MyDB, 554
 mysqli, 429
 NewsAdmin, 568, 584
 Portal, 484, 491
 PortalAdmin, 517, 518, 520
 Registration, 614, 620
 RuntimeException, 166
 SubscriptionsAdmin, 595, 597
 User, 562
 UsersAdmin, 517, 530

klasy

bazowa, 154
 potomna, 154, 156
 zawierająca konstruktor, 151

klauszula

default, 179
 DISTINCT, 602
 FROM, 384
 GROUP BY, 409
 HAVING, 369
 LIMIT, 351
 ORDER BY, 342
 WHERE, 351, 352

klient mysqli, 322

klucz – wartość, 120

klucz, 300

główny, 305
 obcy, 305, 377, 379

kod

błędu, 165
 formularza HTML, 170, 172

kodowanie

znaków, 319, 449, 451
 znaków specjalnych, 45

kody

formatujące dla printf, 200
 powrotów, 585

komenda

cmd, 19
 runas, 19

komentarz

blokowy, 40
 jednowierszowy, 40
 jednowierszowy uniksowy, 41

komunikat, 165

o nieprawidłowym indeksie, 426

- konfiguracja
 - MySQL, 293
 - PHP, 27, 414
 - sesji, 273
 - koniec linii, 196
 - konstruktory, 151
 - klas bazowych, 159
 - konwersja, 75
 - kończenie
 - połączenia z bazą danych, 416, 418
 - pracy serwera, 298
 - sesji, 273
- L**
- liczba
 - argumentów, 115
 - użytkowników, 491
 - zamówionych egzemplarzy, 635
 - licznik
 - graficzny, 252
 - odwiedzin, 250, 271
 - tekstowy, 250
 - lista
 - adresów e-mail, 596
 - plików, 182, 187
 - subskrypcji, 591, 598
 - wiadomości, 571, 583
 - logiczna alternatywa wykluczająca, 62
 - logowanie, 262, 280, 460, 609
 - lokalizacja skryptu, 169
- Ł**
- łączenie
 - ciągów znaków, 210
 - łańcuchów, 194
 - z bazą danych, 415, 418
 - z serwerem MySQL, 308, 310
- M**
- mechanizm sesji, 272
 - metoda
 - addStatRecord, 496
 - addToBasekt, 638
 - character_set_name, 451
 - checkout, 639
 - checkNewsEditRights, 575
 - deleteNews, 567, 584
 - deleteUser, 549, 584
 - editNews, 576
 - editUser, 540, 542
 - execute, 445
 - fetch
 - argumenty, 433
 - fetch_all, 430
 - fetch_array, 430, 432
 - fetch_assoc, 430
 - fetch_field_direct, 430
 - fetch_fields, 487
 - fetch_row, 503
 - fetchAll, 433
 - fetchColumn, 433
 - fetchObject, 433
 - GET, 170
 - getAllUsersOnline, 491, 494
 - getCode, 164
 - getMessage, 164
 - getEmailsList, 600
 - getFile, 164
 - getFullStats, 507, 512
 - getHeaderMenu, 564
 - getLine, 164
 - getMessage, 421
 - getNews, 555
 - getNewsHeaders, 555
 - getNewsList, 555, 558
 - getPagination, 534
 - getRegUsersOnline, 491, 494
 - getQuerySingleResult, 485, 491, 533, 542, 576
 - getShortStats, 491
 - getSubscriptions, 591
 - getUserInfo, 497
 - initDB, 495, 522
 - login, 501, 522, 524, 590, 611
 - logout, 504, 522, 525
 - modify, 636
 - modifyBasket, 638
 - newsAdmin, 565, 566
 - POST, 174
 - query, 429, 443
 - quote, 448
 - real_escape_string, 458, 477
 - registerUser, 616, 618, 621
 - saveOrder, 640, 643
 - saveSubscriptions, 593
 - searchNews, 567
 - searchUser, 546, 548
 - show, 149, 157, 632
 - showBasket, 637
 - showBookDetails, 627
 - showEditForm, 538, 567, 573
 - showId, 156
 - showList, 531, 567, 598
 - showNews, 556
 - showRegistrationForm, 616, 620
 - showSearchForm, 545, 548, 567, 579
 - showSearchResult, 622, 625
 - subscriptionsAdmin, 596, 597

metoda
 updateStatRecord, 504
 usersAdmin, 522, 528, 543, 550

metoday, methods, 143
 wyszukująca wiadomości, 580
 klasy Basket, 629
 klasy Portal, 484, 555, 620

modyfikacja
 bazy danych, 516
 metody login, 611
 pliku index.php, 563
 pliku portal_admin.php, 562
 stylów CSS, 553
 tabel, 335
 tabeli Klienci, 604
 tablicy, 123

modyfikator
 AUTO_INCREMENT, 336
 UNSIGNED, 324
 ZEROFILL, 324

modyfikatory dostępu, access modifiers, 160

MySQL, 289
 instalacja w Linuksie, 294
 instalacja w Windows, 290
 konfiguracja, 293, 295

N

nadawanie uprawnień, 311

nagłówki
 Content-Disposition, 180
 Content-Length, 180
 Content-Type, 180
 headerMainDiv, 607
 Location, 190
 serwisu, 555
 Set-Cookie, 265
 User-Agent, 488

nawiązywanie połączenia z bazą, 420

nazwy użytkowników, 314

negacja logiczna, 63

O

obiekt, 143
 \$basket, 644
 \$db_obj, 458
 \$dbo, 432, 487
 \$portal, 526, 596
 \$sa, 597
 MyDB, 616

obliczenie wielkości katalogu, 230

obsługa
 bazy za pomocą mysqli, 418
 bazy za pomocą PDO, 420

kodów powrotów, 585
 koszyka, 629
 logowania, 497, 522
 MySQL w PHP, 414
 PDO, 415
 plików, 235
 połączeń z bazami, 414
 serwera, 25
 sesji, 272
 subskrypcji, 589
 tabel, 331
 wyjątków, 165
 zamówień, 638

odbieranie plików, download, 179

odbieranie praw, 316

odczyt
 całego pliku, 242
 danych, 237, 430
 danych z formularza, 175
 danych z tabeli przy użyciu PDO, 434
 danych z tablicy asocjacyjnej, 436
 danych za pomocą pętli for, 436
 elementów tablicy, 137
 określonej liczby bajtów, 241
 tablicy, 126
 wartości cookie, 268
 wierszy tekstu, 237
 zawartości katalogu, 223
 zawartości pliku, 237, 240
 zawartości tabeli, 423
 znak po znaku, 240

odwołanie do składowych klasy, 149

ograniczenie ze względu na klucze obce, 378

okno
 Menedżera pakietów, 24
 monitora usługi Apache, 20

OOP, Object Oriented Programming, 143

opcja
 file_uploads, 176
 LIKE, 319
 post_max_size, 176

opcje konfiguracyjne sesji, 273

operacje w bazie danych, 604

operator
 +, 72, 194
 ++, 57
 +=, 65
 ->, 146
 IN, 349
 indeksowania tablic, 66
 kontroli błędów, 67
 kontroli typów, 69
 konwersji typów, 69
 łańcuchowy, 63

- łączenia tablic, 66
- new, 145
- NOT IN, 349
- rozdzielania wyrażeń, 70
- tworzenia obiektów, 70
- warunkowy, 67, 85
- wykonania polecenia zewnętrznego, 68
- zakresu ::, 158
- operatory, 55
 - arytmetyczne, 55
 - bitowe, 59
 - dekrementacji, 56
 - inkrementacji, 56
 - logiczne, 61
 - logiczne w MySQL, 347
 - porównywania tablic, 67
 - przypisania, 64, 65
 - relacyjne, 63
 - relacyjne w MySQL, 346
 - tablicowe, 66
- otwieranie pliku, 235

P

- pakiety PHP, 21
- parametr
 - \$limit, 559
 - \$pass, 262
 - \$timeout, 492
 - action, 169, 497, 498, 519
 - blokowanie, 248
 - flagi, 244
 - href, 556
 - newsId, 574
 - obiekt, 312
 - page, 531
 - prawa, 312
 - skąd, 247
 - tryb, 236
 - Typ, 493
 - wtd, 528
- parametry
 - połączenia z bazą, 455
 - znacznika form, 176
- PDO, PHP Data Objects, 414, 432
- pętla
 - do...while, 95
 - for, 89–93, 100
 - foreach, 96, 100
 - while, 93, 100
- pętle, 102
- PHP, 15
 - instalacja w Linuksie, 23
 - instalacja w Windows, 17
 - konfiguracja, 27, 414
- plik
 - 404.html, 190
 - adminLogin Form.php, 522
 - bad_login.html, 457
 - basket.php, 629
 - browscap.ini, 489
 - downloads.txt, 189
 - editUserForm.php, 536
 - error_server.html, 457
 - form.html, 270
 - form.php, 279
 - httpd.conf, 22
 - index1.html, 457
 - index.php, 26, 517, 620
 - instalatora MSI, 21
 - links.txt, 264
 - login.html, 455
 - login.php, 455
 - loginForm.php, 499, 610
 - main.php, 270, 463, 606
 - my.ini, 321
 - mydb.php, 554
 - new_user_form.php, 469
 - newsAdminMenu.php, 565
 - newsEditForm.php, 573
 - orderNoLoginInfoDiv.php, 640
 - passwords.txt, 279
 - php.ini, 22, 176
 - php_pdo_mysql.dll, 415
 - php5apache2_2.dll, 22, 29
 - portal.php, 485, 501
 - portal_admin.php, 518
 - searchForm.php, 622
 - searchUserForm.php, 544
 - statTable.php, 512
 - subscriptionsAdminMenu.php, 596
 - usersAdminMenu.php, 528
- pliki
 - .php, 22
 - konfiguracyjne serwera, 22
- pobieranie
 - danych, 425, 560
 - danych z tabeli, 341, 344
 - danych z wielu tabel, 357
 - liczby wierszy, 533
 - listy adresów IP, 511
 - listy użytkowników, 511
 - selektywne danych, 345
 - statystyk przeglądarek, 511
 - statystyk systemów, 511
 - wyników zapytań, 426, 437, 510
 - zawartości całej tabeli, 342
- podzapytania
 - proste, 381
 - skorelowane, 382

podzapytania
 w instrukcjach, 385
 w klauzuli FROM, 384
 podział wiersza, 196
 pojedynczy wiersz, 617, 593
 pola, fields, 143
 pola wyboru, 592
 pole error, 177
 pole klasy, 144
 polecenia do modyfikowania tabeli, 335
 polecenie
 cmd.exe, 309
 CREATE DATABASE, 310, 318
 CREATE USER, 311
 DESCRIBE, 334
 DROP DATABASE, 310
 DROP TABLE, 337
 DROP USER, 317
 GRANT, 311
 RENAME USER, 316
 REVOKE, 316
 SHOW COLLATION, 320
 SHOW COLUMNS, 334
 SHOW DATABASES, 318
 SHOW TABLES, 319
 sudo, 25
 systemctl, 24
 polskie litery, 449
 połączenie z bazą, 435
 porównywanie ciągów, 204
 porządkowanie leksykograficzne, 204
 poziom
 bazy danych, database level, 312
 globalny, global level, 312
 kolumny, column level, 312
 tabeli, table level, 312
 poziomy przywilejów, 312
 pozycja wskaźnika, 247
 prawo do edycji, 574
 prezentacja szczegółowych danych, 627
 priorytety operatorów, 71
 procedura logowania, 280, 527
 programowanie obiektowe, 143
 przechwytywanie wyjątków, 164
 przekazywanie argumentów, 112
 przez referencję, reference, 113
 przez wartość, by value, 112
 przesłanianie składowych, members overriding, 157
 przeszukiwanie ciągów, 206
 przetwarzanie żądania, 413
 przypadki case, 555
 przywilej, 561
 Administrator, 561
 Edycja news, 561
 Zarządzanie użytkownikami, 561
 PWS, Personal Web Server, 17

R

reguły porównywania znaków, 320
 rejestracja użytkowników, 467, 614
 rejestrowanie
 MySQL, 294
 użytkownika, 618
 relacje, 301
 relacje między tabelami, 392
 relacyjne bazy danych, 289, 300
 rozpoczynanie sesji, 272
 rzutowanie typów, 72

S

schemat przetwarzania danych, 414
 sekcja body, 496
 serwer
 Apache, 18
 bazy danych, 297
 lokalny localhost, 171
 MySQL, 17
 WWW, 17, 18
 sesja, 272
 składnia
 alternatywna, 98
 heredoc, 46
 nowdoc, 46
 skrypt
 dodający wiersz, 437
 dodający wiersz danych, 444
 download.php, 182, 186
 form.php, 281, 461, 464
 generujący
 główną część serwisu, 283
 listę, 188
 listę odnośników, 263
 listę plików, 185
 index.php, 461
 index1.php, 284
 index2.php, 285
 kontynuujący sesję, 277
 kończący sesję, 278
 licznika odwiedzin, 250
 login.php, 279, 461, 500
 logout.php, 279, 461, 466, 501
 logowania, 459
 main.php, 465
 new_user.php, 472
 obsługujący pole wyboru, 173
 odbierający dane, 246
 odbierający dane z formularza, 440
 odbierający plik, 178
 PHP weryfikujący dane, 455

- przetwarzający dane, 270
 - przetwarzający dane z ankiety, 255
 - rozpoczynający sesję, 276
 - rozwiązujący równania kwadratowe, 81
 - testujący połączenie, 417
 - umożliwiający nawigację po katalogach, 232
 - umożliwiający wylogowanie, 284
 - weryfikujący dane, 261
 - wysyłający plik, 181–189
 - wyświetlający napis, 30
 - wyświetlający zawartość katalogu, 224
- skrypty zewnętrzne, 35
- słowo kluczowe
- array, 117
 - construct, 151
 - destruct, 153
 - elseif, 79
 - extends, 153
 - function, 144
 - globals, 109
 - parent, 158
 - private, 161
 - protected, 161
 - public, 144, 161
 - this, 150
- sortowana lista plików, 226
- sortowanie, 130
- ciągów, 205
 - niestandardowe, 132
- tablic, 131
- tablic asocjacyjnych, 133
- w tabeli, 342
- sól, salt, 459
- sposoby obsługi baz MySQL, 448
- spójność danych, 402
- sprawdzanie
- poprawności danych, 471
 - przywilejów, 317
- SQL, Structured Query Language, 323
- stała
- INVALID_NEWS_ID, 575
 - INVALID_USER_NAME, 476
 - LOGIN_FAILED, 485
 - LOGIN_OK, 485
 - SERVER_ERROR, 485
- stałe, 53
- predefiniowane, 54
 - w portal_admin.php, 521
 - w skrypcie sklepu internetowego, 609
- statystyki, 479, 505
- statystyki odwiedzin, 507
- strona testowa serwera WWW, 26
- struktura
- bazy, 304
 - klasy Basket, 630
 - klasy Portal, 608
 - klasy Registration, 614
- styl
- obiektowy, 429, 432, 443, 445
 - proceduralny, 421, 437
- style CSS, 483
- style formatujące tabelę, 513
- subskrypcja biuletynów, 593
- subskrypcje, 587
- suma logiczna, 62
- synchronizacja dostępu, 248
- system news, 551
- system przywilejów, 561
- szkielet portalu, 482

Ś

- ścieżka dostępu do pliku, 29
- śledzenie
 - odwołań do podstron, 286
 - użytkownika, 284
- środowisko PHP, 21

T

- tabela
- Klienci, 603
 - News, 552
 - Stats, 480, 489
 - Subskrypcje, 587
 - Users, 458
 - Uzytkownicy_Subskrypcje, 588
- tabele, 300
- tablica
- \$_COOKIE, 268
 - \$_FILES, 177
 - \$_GET, 171
 - \$_POST, 175, 617
 - \$_SERVER, 488
 - \$_SESSION, 275
 - \$arr, 227
 - \$fieldsFromForm, 620
 - Klienci, 612
- tablice, 117
- asocjacyjne, 120
 - dwuwymiarowe, 125
 - jednowymiarowe, 124
 - nieregularne, 129
 - trójwymiarowe, 127
 - wielowymiarowe, 124
- testowanie
- działania PHP, 26
 - instalacji Apache'a i PHP, 23
 - połączenia, 421

- testowanie
 - połączenia z bazą, 417–421
 - serwera WWW, 21, 25
 - tęczowe tablice, rainbow tables, 459
 - tryb
 - MYSQLI_STORE_RESULT, 422
 - MYSQLI_USE_RESULT, 422
 - przetwarzania zapytań, 422
 - tworzenie
 - baz, 308
 - baz przy użyciu serwera MySQL, 389
 - indeksów, 403
 - indeksów i kluczy obcych, 405
 - catalogów, 227
 - klas, 143
 - kluczy obcych, 403
 - kont użytkowników, 311
 - obiektów, 145
 - obiektu klasy PDO, 420
 - ograniczeń, 377
 - sklepu internetowego, 603
 - stałych, 54
 - struktury portalu, 482
 - systemu news, 554
 - tabel, 304, 331, 358, 393
 - tabel bazy, 401
 - tablicy
 - asocjacyjnej, 120, 121
 - dwuwymiarowej, 125
 - nieregularnej, 129
 - zmiennych, 48
 - typ
 - array, 46
 - boolean, 42, 75
 - float, 42, 76
 - integer, 42, 75
 - MIME, 180
 - null, 47
 - object, 46
 - resource, 47
 - string, 43, 76
 - typy
 - BINARY i VARBINARY, 330
 - BLOB, 330
 - BLOB i TEXT, 330
 - całkowitoliczbowe, 325
 - CHAR i VARCHAR, 329
 - daty i czasu, 327
 - ENUM i SET, 331
 - liczbowe, 324
 - łańcuchowe, 329
 - podzapytań
 - proste, simple, 380
 - skorelowane, correlated, 380
 - proste, 86
 - relacji
 - jeden do jednego, 302
 - jeden do wielu, 302
 - wiele do wielu, 302
 - tabel w MySQL, 375
 - złączeń, 359
 - zmiennoprzecinkowe, 326
- ## U
- uprawnienia, privileges, 312
 - CREATE, 314
 - DELETE, 314
 - INSERT, 314
 - SELECT, 314
 - UPDATE, 314
 - uruchamianie
 - serwera MySQL, 297, 298
 - serwera WWW, 19, 24
 - ustawienia dla środowiska deweloperskiego, 23
 - produkcyjnego, 23
 - usuwanie
 - baz, 310
 - białych znaków, 195
 - cookie, 268
 - danych, 353, 549
 - elementów z tablicy, 139
 - indeksu, 377
 - catalogów, 228
 - kont użytkowników, 317
 - tabel, 337
 - wiadomości, 584
 - zawartości katalogu, 231
 - znaczników HTML, 239, 245
 - uwierzytelnianie, 278, 455
 - uwierzytelnianie z wykorzystaniem sesji, 461
- ## W
- warstwa
 - basketDiv, 634
 - footerDiv, 496
 - główna, 519
 - headerDiv, 519, 527, 564
 - headerMainDiv, 589
 - headerUserInfoDiv, 496
 - mainContentDiv, 496, 497, 498, 519, 607
 - nagłówek, 519
 - podkładowa, 519
 - registrationFormDiv, 617
 - rightSideDiv, 553, 555
 - searchResultsDiv, 626
 - z komunikatem, 608

- warstwy
 - formatujące formularz, 506
 - HTML, 482
 - wartości
 - argumentu
 - filtr, 577
 - typ_danych, 577
 - egzemplarzy, 636
 - parametru
 - action, 497, 519
 - flagi, 244
 - prawa, 312, 313
 - tryb, 236
 - Typ, 493
 - wtd, 528, 565
 - zamówienia, 636
 - wartość
 - FALSE, 347
 - FILE_APPEND, 244
 - FILE_USE_INCLUDE_PATH, 244
 - LOCK_EX, 244
 - NULL, 339
 - TRUE, 347
 - wczytywanie poleceń, 322
 - weryfikacja danych, 174, 246
 - wiadomości, 552
 - widok listy wiadomości, 560
 - wielkość liter, 197
 - wiersze w tabelach, 392
 - więzy integralności, 402
 - właściwości, properties, 143
 - wprowadzanie danych, 269, 337, 439
 - do bazy, 406
 - do tabeli, 447
 - wskazanie this, 149
 - wskaźnik pozycji w pliku, 247
 - wskaźnik tablicy, 138
 - wstawianie wielu wierszy, 340
 - wygląd
 - formularza logowania, 500
 - formularza rejestracyjnego, 615
 - strony administracyjnej, 523
 - wyjątek typu PDOException, 448
 - wyjątki, 162
 - wylogowanie, 494, 613
 - wyniki wyszukiwania, 549
 - wrażenia regularne, regular expressions, 476
 - wrażenia warunkowe, 84
 - wysyłanie
 - danych do skryptu, 506
 - plików, upload, 176, 180, 182
 - wyszukiwanie, 623, 624
 - danych, 621
 - pojedynczej wiadomości, 583
 - użytkowników, 544
 - wiadomości, 579
 - wyświetlenie
 - formularza logowania, 604
 - formularza rejestracyjnego, 604, 616
 - formularza wyszukiwania, 604
 - informacji, 604
 - listy baz danych, 318
 - listy użytkowników, 531
 - listy wiadomości, 568
 - strony głównej, 604
 - szczegółowych danych, 629
 - wyników głosowania, 256
 - zamówienia, 604
 - zawartości koszyka, 604, 632
 - zawartości tabeli, 427
 - zawartości tablicy, 118
 - wywołanie
 - funkcji implode, 442
 - konstruktor, 153, 160
 - metody editUser, 543
 - skryptu index.php, 534
 - wywoływanie klienta, 309
- ## Z
- zamiana podciągów, 208
 - zamykanie połączenia, 420
 - zapis danych, 243
 - zapisanie
 - użytkownika, 460
 - zamówienia, 640
 - zapisywanie
 - danych w pliku, 243
 - głosów, 258
 - uprawnień, 516
 - zapytanie
 - INSERT INTO, 442
 - pobierające dane, 431
 - złożone, 350
 - zarządzanie
 - księgarnią, 605
 - subskrypcjami, 598
 - zasady logowania, 459
 - zasięg zmiennych, 108
 - zawartość koszyka, 634, 635
 - zestaw znaków, character set, 320
 - zgłaszanie wyjątków, 163
 - zliczanie użytkowników, 492, 493
 - złączenia, 357
 - tabel, 359
 - tabel News i Users, 557
 - trzech tabel, 372

- złączenie
 - INNER JOIN, 360
 - LEFT JOIN, 360, 599
 - RIGHT JOIN, 362
- zmiana
 - domyślnej bazy danych, 417, 419
 - katalogu bieżącego, 228
 - kodowania znaków, 451
 - nazw kolumn, 345
 - nazwy konta użytkownika, 316
- zmienna
 - \$_COOKIE, 52
 - \$_ENV, 52
 - \$_FILES, 52
 - \$_GET, 52
 - \$_POST, 52
 - \$_REQUEST, 53
 - \$_SERVER, 52
 - \$_SESSION, 53
 - \$activeUsers, 510
 - \$allVisits, 510
 - \$browsersInfo, 510
 - \$contents, 181
 - \$count, 252, 602
 - \$datafile, 256
 - \$dir, 225
 - \$fd, 181
 - \$fieldsNames, 620
 - \$file, 225
 - \$filePath, 184, 185
 - \$found, 191
 - \$GLOBALS, 52
 - \$guestVisits, 510
 - \$haslo, 542
 - \$ips, 510
 - \$komunikat, 465
 - \$lastips, 510
 - \$link, 264
 - \$name, 191
 - \$newsId, 557
 - \$page, 559
 - \$path, 191
 - \$query, 442
 - \$readonly, 537, 539
 - \$regVisits, 510
 - \$result, 459, 512
 - \$row, 458
 - \$rowsNo., 602
 - \$separator, 602
 - \$size, 181
 - \$sort, 428
 - \$statsInfo, 509
 - \$systemsInfo, 510
 - \$tab, 210
 - \$timeout, 495
 - \$uploadDir, 179
 - \$userId, 592
 - \$val, 464
 - \$wtd, 537, 539
 - GLOBALS, 109
 - statRecordId, 504
 - upload_max_filesize, 176
 - upload_tmp_dir, 176
 - zalogowany_adm, 527
- zmiennne, 47, 108
 - globalne, 109
 - sesji, 275
 - superglobalne, 51
- znacznik
 - ?>, 32
 - <?php, 32
 -
, 196
 - <div>, 482
 - <form>, 169
 - , 252
 - , 189
 - <table>, 424, 617
 - <td>, 148, 424
 - <th>, 424
 - <tr>, 424, 486
 - , 225
 - czasu Uniksa, 214
 - form, 176
- znaczniki
 - formatujące dla date, 213
 - formatujące dla strftime, 219
 - kanoniczne, 33
 - skryptów HTML, 34
 - typu ASP, 34
 - typu SGML, 34
- znak
 - \, 235
 - &, ampersand, 113
 - *, 365
 - /, 235
- znaki
 - \n, 600
 - \r, 600
 - \t, 600
 - apostrofu, 43
 - apostrofu lewego, 442, 619
 - apostrofu prostego, 442, 619
 - cudzysłowu, 44
 - potencjalnie niebezpieczne, 183
 - specjalne, 442
 - zwracanie wartości, 107

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>

PHP i MySQL. Dla każdego. Wydanie II

PHP i MySQL są w świecie nowoczesnych, dynamicznych serwisów nierozłącznie niemal jak Bonnie i Clyde, jednak w przeciwieństwie do pary słynnych przestępców mają przed sobą wspaniałą przyszłość. To właśnie dzięki tym dwóm technologiom funkcjonuje większość rozbudowanych witryn internetowych, na nich też opiera się działanie wielu najbardziej popularnych systemów zarządzania treścią. Nic w tym dziwnego, bo duet składający się z PHP i MySQL-a zapewnia ogromne możliwości, dużą elastyczność i wysoką stabilność pracy. Ponadto wystarczy znaleźć odpowiednie źródło wiedzy, by łatwo i szybko poznać obydwa te rozwiązania oraz opanować sposoby stosowania ich w praktyce.

Takim źródłem jest książka *PHP i MySQL. Dla każdego. Wydanie II*. Dzięki niej początkujący twórcy serwisów mogą bez kompleksów wkroczyć w świat nowoczesnych technologii internetowych i nauczyć się sprawnie używać bezpłatnych narzędzi do pisania, testowania czy wdrażania aplikacji WWW, a bardziej zaawansowani – usystematyzować i rozszerzyć posiadaną wiedzę. Podręcznik krok po kroku prezentuje sposoby instalacji i konfiguracji środowiska pracy, omawia składnię i możliwości języka PHP, a także wskazuje, jak poprawnie projektować i tworzyć bazy danych oparte na mechanizmie MySQL. Z książki dowiesz się, jak posługiwać się językiem SQL oraz jak połączyć możliwości PHP i MySQL-a, a ponadto jak za ich pomocą tworzyć praktyczne serwisy WWW.

**Obowiązkowa pozycja w bibliotece
każdego początkującego twórcy
serwisów WWW.**

helion.pl
księgarnia
internetowa



Helion

Nr katalogowy: **10306**



Księgarnia internetowa:
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900



0 601 339900

Sprawdź najnowsze promocje:

👉 <http://helion.pl/promocje>

Książki najczęściej czytane:

👉 <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

👉 <http://helion.pl/nowosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

▼ Instalacja i konfiguracja niezbędnych narzędzi w różnych systemach operacyjnych

▼ Przegląd typów, stałych i operatorów języka PHP, tworzenie zmiennych i ich używanie

▼ Instrukcje sterujące i pętle, budowa funkcji i korzystanie z nich

▼ Tworzenie tablic i ich stosowanie, elementy programowania obiektowego w PHP

▼ Nawigowanie w systemie plików, korzystanie z cookies i obsługa sesji

▼ Zarządzanie bazami danych oraz kontami korzystających z nich użytkowników

▼ Praktyczne przykłady serwisów WWW opartych na języku PHP i bazie MySQL

sięgnij po **WIECEJ**



rodzajowa

ISBN 978-83-246-4797-2



9 788324 647972

Cena 89,00 zł

Informatyka w najlepszym wydaniu