

» Idź do

- Spis treści
- Przykładowy rozdział

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

- Fragmenty książek online

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl
© Helion 1991-2008

PHP i MySQL. Tworzenie stron WWW. Vademecum profesjonalisty Wydanie czwarte

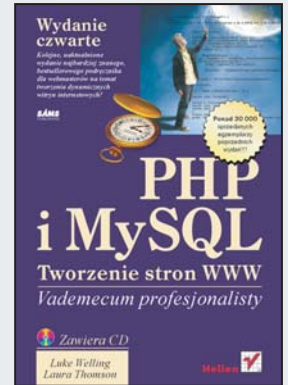
Autorzy: [Luke Welling](#), [Laura Thomson](#)

Tłumaczenie: 978-83-246-0821-8

ISBN: 83-7197-641-0

Tytuł oryginału: [PHP and MySQL Web Development, Fourth Edition](#)

Format: 172×245, stron: 856



Kolejne, uaktualnione wydanie najbardziej znanego, bestsellerowego podręcznika dla webmasterów na temat tworzenia dynamicznych witryn internetowych! Ponad 30 000 sprzedanych egzemplarzy poprzednich wydań!!! Recepta na sukces w przypadku tworzenia profesjonalnych stron WWW jest niezwykle prosta: wystarczą ogromne możliwości PHP, niezrównana wydajność MySQL i wielka, chętna do pomocy społeczność, skupiona wokół tego tandemu. Wynik? Błyskawiczna realizacja zaawansowanych stron i aplikacji internetowych. Wszystko, czego Ci jeszcze trzeba, to fachowa wiedza, pozwalająca wykorzystać ten potencjał!

PHP i MySQL to jeden z najlepszych, najpopularniejszych zestawów do tworzenia rozwiązań internetowych, a książka, którą trzymasz w rękach, to czwarte wydanie bestsellerowego i kultowego już kompendium wiedzy na temat tych narzędzi. Dzięki niej zorientujesz się w każdym z aspektów wykorzystania PHP wraz z bazą danych MySQL. Poznasz język PHP, metody przechowywania i wyszukiwania danych oraz operacje na plikach. Dowiesz się, jak tworzyć uniwersalny kod i sprawnie pracować z bazą danych. Nauczysz się szybko lokalizować i rozwiązywać problemy oraz zapewniać bezpieczeństwo aplikacjom, a także efektywnie stosować technologię AJAX. Podręcznik zawiera także mnóstwo praktycznych przykładów, demonstrujących wykorzystanie PHP i MySQL do realizacji różnych zadań związanych z funkcjonowaniem dynamicznych witryn WWW.

- Wybrane zalety PHP i MySQL
- Podstawy języka PHP
- Metody przechowywania i wyszukiwania danych
- Operacje na plikach
- Zastosowanie tablic
- Wykorzystanie wyrażeń regularnych oraz operacje na ciągach znaków
- Tworzenie uniwersalnego kodu – ponowne jego wykorzystanie
- Obiekty w PHP
- Obsługa wyjątków i błędów
- Praca z bazą danych MySQL
- Nawiązywanie połączenia z bazą z poziomu PHP
- Administracja MySQL
- Zaawansowane zagadnienia, związane z bazą MySQL
- Zapewnienie bezpieczeństwa tworzonym rozwiązaniom
- Metody uwierzytelniania przy użyciu PHP i MySQL
- Wykorzystanie protokołów i funkcji sieci
- Generowanie grafik
- Wykorzystanie sesji
- Obsługa poczty elektronicznej za pomocą PHP
- Użycie technologii AJAX

Kultowe kompendium wiedzy na temat tworzenia dynamicznych witryn!

Spis treści

	O autorach	23
	O współautorach	25
	Wprowadzenie	27
Część I	Stosowanie PHP	37
Rozdział 1.	Podstawowy kurs PHP	39
	Zastosowanie PHP	40
	Tworzenie przykładowej aplikacji: „Części samochodowe Janka”	40
	Formularz zamówienia	40
	Przetwarzanie formularza	41
	Osadzanie PHP w HTML	42
	Zastosowanie znaczników PHP	43
	Instrukcje PHP	44
	Odstępy	44
	Komentarze	45
	Dodawanie zawartości dynamicznej	45
	Wywoływanie funkcji	46
	Używanie funkcji date()	46
	Dostęp do zmiennych formularza	47
	Zmienne formularza	47
	Łączenie ciągów	49
	Zmienne i ciągi znaków	50
	Identyfikatory	51
	Typy zmiennych	51
	Typy danych w PHP	51
	Siła typu	52
	Rzutowanie typu	52
	Zmienne zmiennych	53
	Deklarowanie i używanie stałych	53
	Zasięg zmiennych	54
	Używanie operatorów	55
	Operatory arytmetyczne	55
	Operatory ciągów	56
	Operatory przypisania	56
	Operatory porównań	58
	Operatory logiczne	59
	Operatory bitowe	60
	Pozostałe operatory	60

Obliczanie sum w formularzu	62
Pierwszeństwo i kolejność	63
Zarządzanie zmiennymi	65
Sprawdzanie i ustawianie typów zmiennych	65
Sprawdzanie stanu zmiennej	66
Reinterpretacja zmiennych	67
Podejmowanie decyzji za pomocą instrukcji warunkowych	67
Instrukcja if	67
Bloki kodu	68
Instrukcja else	68
Instrukcja elseif	69
Instrukcja switch	69
Porównanie różnych instrukcji warunkowych	71
Powtarzanie działań przy użyciu iteracji	71
Pętla while	72
Pętla for i foreach	73
Pętla do..while	74
Wyłamywanie się ze struktury skryptu	75
Używanie alternatywnych składni struktur sterujących	75
Używanie struktury declare	76
W następnym rozdziale	76
Rozdział 2. Przechowywanie i wyszukiwanie danych	77
Zapisywanie danych do późniejszego użycia	77
Przechowywanie i wyszukiwanie zamówień Janka	78
Przetwarzanie plików	79
Otwieranie pliku	79
Tryby otwarcia pliku	79
Stosowanie funkcji fopen() do otwarcia pliku	80
Otwieranie pliku przez protokół FTP lub HTTP	82
Problemy z otwieraniem plików	82
Zapisywanie danych w pliku	84
Parametry funkcji fwrite()	85
Formaty plików	85
Zamykanie pliku	86
Odczyt z pliku	87
Otwieranie pliku w celu odczytu — fopen()	89
Wiedzieć, kiedy przestać — feof()	89
Odczytywanie pliku wiersz po wierszu — fgets(), fgets() i fgetsv()	89
Odczyt całego pliku — readfile(), fpassthru(), file()	90
Odczyt pojedynczego znaku — fgetc()	91
Odczytywanie zadanej długości — fread()	91
Inne przydatne funkcje plikowe	91
Sprawdzanie istnienia pliku — file_exists()	92
Określanie wielkości pliku — filesize()	92
Kasowanie pliku — unlink()	92
Poruszanie się wewnątrz pliku — rewind(), fseek() i ftell()	92
Blokowanie pliku	93
Lepszy sposób obróbki danych — systemy zarządzania bazami danych	94
Problemy związane ze stosowaniem plików jednorodnych	95
Jak RDBMS rozwiązują powyższe problemy?	95
Propozycje dalszych lektur	96
W następnym rozdziale	96

Rozdział 3.	Stosowanie tablic	97
	Czym są tablice?	97
	Tablice indeksowane numerycznie	98
	Inicjowanie tablic indeksowanych numerycznie	98
	Dostęp do zawartości tablicy	99
	Dostęp do tablic przy zastosowaniu pętli	100
	Tablice z innymi indeksami	100
	Inicjowanie tablicy	100
	Dostęp do elementów tablicy	101
	Stosowanie pętli	101
	Operatory tablicowe	103
	Tablice wielowymiarowe	103
	Sortowanie tablic	106
	Stosowanie funkcji <code>sort()</code>	106
	Stosowanie funkcji <code>asort()</code> i <code>ksort()</code> do porządkowania tablic	107
	Sortowanie odwrotne	107
	Sortowanie tablic wielowymiarowych	108
	Typy sortowań definiowane przez użytkownika	108
	Odwrotne sortowanie zdefiniowane przez użytkownika	109
	Zmiany kolejności elementów w tablicach	110
	Stosowanie funkcji <code>shuffle()</code>	110
	Stosowanie funkcji <code>array_reverse()</code>	111
	Ładowanie tablic z plików	112
	Wykonywanie innych działań na tablicach	114
	Poruszanie się wewnątrz tablicy	
	— funkcje <code>each()</code> , <code>current()</code> , <code>reset()</code> , <code>end()</code> , <code>next()</code> , <code>pos()</code> i <code>prev()</code>	114
	Dołączanie dowolnej funkcji do każdego elementu tablicy — funkcja <code>array_walk()</code>	115
	Liczenie elementów tablicy: <code>count()</code> , <code>sizeof()</code> i <code>array_count_values()</code>	116
	Konwersja tablic na zmienne skalarne — funkcja <code>extract()</code>	117
	Propozycje dalszych lektur	118
	W następnym rozdziale	118
Rozdział 4.	Manipulowanie ciągami i wyrażenia regularne	119
	Przykładowa aplikacja — Inteligentny Formularz Pocztowy	119
	Formatowanie ciągów	121
	Przycinanie ciągów — funkcje <code>chop()</code> , <code>ltrim()</code> i <code>trim()</code>	121
	Formatowanie ciągów w celu ich prezentacji	122
	Formatowanie ciągów do przechowania — funkcje <code>addslashes()</code> i <code>stripslashes()</code>	125
	Łączenie i rozdzielanie ciągów za pomocą funkcji ciągów	127
	Stosowanie funkcji <code>explode()</code> , <code>implode()</code> i <code>join()</code>	127
	Stosowanie funkcji <code>strtok()</code>	128
	Stosowanie funkcji <code>substr()</code>	128
	Porównywanie ciągów	129
	Porządkowanie ciągów — funkcje <code>strcmp()</code> , <code>strcasecmp()</code> i <code>strnatcmp()</code>	129
	Sprawdzanie długości ciągu za pomocą funkcji <code>strlen()</code>	130
	Dopasowywanie i zamiana podciągów za pomocą funkcji ciągów	130
	Znajdowanie ciągów w ciągach — funkcje <code>strpos()</code> , <code>strchr()</code> , <code>strchr()</code> i <code>stristr()</code>	131
	Odnajdywanie pozycji podciągu — funkcje <code>strpos()</code> i <code>strrpos()</code>	131
	Zamiana podciągów — funkcje <code>str_replace()</code> i <code>substr_replace()</code>	132
	Wprowadzenie do wyrażen regularnych	133
	Podstawy	133
	Zbiory i klasy znaków	134
	Powtarzalność	135
	Podwyrażenia	135
	Podwyrażenia policzalne	135

	Kotwiczenie na początku lub na końcu ciągu	136
	Rozgałęzianie	136
	Dopasowywanie specjalnych znaków literowych	136
	Podsumowanie znaków specjalnych	137
	Umieszczanie wszystkiego razem (Inteligentny Formularz)	137
	Odnajdywanie podciągów za pomocą wyrażeń regularnych	138
	Zamiana podciągów za pomocą wyrażeń regularnych	139
	Rozdzielanie ciągów za pomocą wyrażeń regularnych	139
	Propozycje dalszych lektur	140
	W następnym rozdziale	140
Rozdział 5.	Ponowne wykorzystanie kodu i tworzenie funkcji	141
	Zalety ponownego stosowania kodu	141
	Koszt	142
	Niezawodność	142
	Spójność	142
	Stosowanie funkcji require() i include()	142
	Rozszerzenia plików i require()	143
	Stosowanie require() w szablonach stron WWW	144
	Stosowanie opcji auto_prepend_file i auto_append_file	148
	Stosowanie funkcji w PHP	149
	Wywoływanie funkcji	149
	Wywołanie niezdefiniowanej funkcji	151
	Wielkość liter a nazwy funkcji	152
	Definiowanie własnych funkcji	152
	Podstawowa struktura funkcji	152
	Nadawanie nazwy funkcji	153
	Parametry	154
	Zasięg	156
	Przekazanie przez referencję czy przekazanie przez wartość?	158
	Stosowanie słowa kluczowego return	159
	Zwracanie wartości przez funkcje	160
	Implementacja rekurencji	161
	Przestrzenie nazw	162
	Propozycje dalszych lektur	163
	W następnym rozdziale	163
Rozdział 6.	Obiektowy PHP	165
	Koncepcje programowania obiektowego	165
	Klasy i obiekty	166
	Polimorfizm	167
	Dziedziczenie	167
	Tworzenie klas, atrybutów i operacji w PHP	168
	Struktura klasy	168
	Konstruktory	168
	Destruktory	169
	Tworzenie egzemplarzy	169
	Stosowanie atrybutów klasy	170
	Kontrolowanie dostępu przy użyciu private i public	172
	Wywoływanie operacji klas	172
	Implementacja dziedziczenia w PHP	173
	Kontrolowanie widoczności w trakcie dziedziczenia przy użyciu private i protected	174
	Unieważnianie	175
	Zapobieganie dziedziczeniu i unieważnianiu przy użyciu final	176
	Wielodziedziczenie	177
	Implementowanie interfejsów	177

	Tworzenie klas	178
	Tworzenie kodu dla własnej klasy	179
	Zaawansowane mechanizmy obiektowe w PHP	186
	Używanie stałych klasowych	186
	Implementowanie metod statycznych	186
	Sprawdzanie typu klasy i wskazywanie typu	186
	Późne wiązania statyczne	187
	Klonowanie obiektów	188
	Używanie klas abstrakcyjnych	188
	Przeciążanie metod przy użyciu <code>__call()</code>	188
	Używanie metody <code>__autoload()</code>	189
	Implementowanie iteratorów i iteracji	190
	Przekształcanie klas w łańcuchy znaków	191
	Używanie API Reflection	192
	W następnym rozdziale	192
Rozdział 7.	Obsługa błędów i wyjątków	195
	Koncepcja obsługi wyjątków	195
	Klasa Exception	196
	Wyjątki definiowane przez użytkownika	197
	Wyjątki w Częściach samochodowych Janka	200
	Wyjątki i inne mechanizmy obsługi błędów w PHP	202
	Propozycje dalszych lektur	203
	W następnym rozdziale	203
Część II	Stosowanie MySQL	205
Rozdział 8.	Projektowanie internetowej bazy danych	207
	Koncepcje relacyjnych baz danych	208
	Tabele	208
	Kolumny	208
	Wiersze	208
	Wartości	208
	Klucze	209
	Schematy	209
	Relacje	210
	Jak zaprojektować internetową bazę danych?	211
	Określ obiekty świata realnego, których model chcesz wykonać	211
	Unikaj przechowywania redundantnych danych	211
	Zapisuj atomowe wartości kolumn	213
	Dobierz właściwe klucze	214
	Pomyśl o zapytaniach, które zadasz bazie	214
	Unikaj tworzenia tabel z wieloma pustymi polami	214
	Typy tabel — podsumowanie	215
	Architektura internetowej bazy danych	215
	Propozycje dalszych lektur	216
	W następnym rozdziale	216
Rozdział 9.	Tworzenie internetowej bazy danych	217
	Użytkowanie monitora MySQL	218
	Logowanie się do serwera MySQL	219
	Tworzenie baz i rejestrowanie użytkowników	220
	Definiowanie użytkowników i przywilejów	220

Wprowadzenie do systemu przywilejów MySQL	221
Zasada najmniejszego przywileju	221
Rejestrowanie użytkowników: polecenie GRANT	221
Typy i poziomy przywilejów	223
Polecenie REVOKE	224
Przykłady użycia poleceń GRANT i REVOKE	225
Rejestrowanie użytkownika łączącego się z internetu	226
Używanie odpowiedniej bazy danych	226
Tworzenie tabel bazy danych	227
Znaczenie dodatkowych atrybutów kolumn	228
Typy kolumn	229
Rzut oka na bazę danych — polecenia SHOW i DESCRIBE	231
Tworzenie indeksów	232
Identyfikatory MySQL	232
Wybór typów danych w kolumnach	233
Typy liczbowe	233
Propozycje dalszych lektur	236
W następnym rozdziale	237
Rozdział 10. Praca z bazą danych MySQL	239
Czym jest SQL?	239
Zapisywanie danych do bazy	240
Wyszukiwanie danych w bazie	242
Wyszukiwanie danych spełniających określone kryteria	243
Wyszukiwanie danych w wielu tabelach	245
Szeregowanie danych w określonym porządku	249
Grupowanie i agregowanie danych	250
Wskazanie wierszy, które mają być wyświetlone	252
Używanie podzapytań	252
Dokonywanie zmian rekordów w bazie danych	255
Zmiana struktury istniejących tabel	255
Usuwanie rekordów z bazy danych	257
Usuwanie tabel	257
Usuwanie całych baz danych	258
Propozycje dalszych lektur	258
W następnym rozdziale	258
Rozdział 11. Łączenie się z bazą MySQL za pomocą PHP	259
Jak działa internetowa baza danych?	259
Wykonywanie zapytań do bazy danych z poziomu strony WWW	262
Sprawdzenie poprawności wpisanych danych	263
Ustanawianie połączenia z bazą danych	264
Wybór właściwej bazy danych	265
Wysyłanie zapytań do bazy danych	265
Odczytywanie rezultatów zapytań	266
Zamykanie połączenia z bazą danych	267
Wstawianie nowych danych do bazy	267
Używanie instrukcji przygotowywanych	270
Używanie innych interfejsów bazodanowych PHP	272
Stosowanie ogólnego interfejsu bazodanowego: PEAR MDB2	272
Propozycje dalszych lektur	274
W następnym rozdziale	274

Rozdział 12. Administrowanie MySQL dla zaawansowanych	275
Szczegóły systemu przywilejów	275
Tabela user	276
Tabele db i host	278
Tabele tables_priv, columns_priv i procs_priv	278
Kontrola dostępu: w jaki sposób MySQL używa tabel przywilejów	279
Zmiana przywilejów: kiedy zmiany zostaną uwzględnione?	281
Ochrona bazy danych	282
MySQL z perspektywy systemu operacyjnego	282
Hasła	282
Przywileje użytkowników	283
MySQL i internet	283
Uzyskiwanie szczegółowych informacji o bazie danych	284
Uzyskiwanie informacji poleceniem SHOW	284
Uzyskiwanie informacji o kolumnach za pomocą polecenia DESCRIBE	286
Jak wykonywane są zapytania: polecenie EXPLAIN	286
Optymalizowanie bazy danych	291
Optymalizacja projektu bazy danych	291
Przywileje	291
Optymalizacja tabel	291
Stosowanie indeksów	292
Używanie wartości domyślnych	292
Więcej wskazówek	292
Tworzenie kopii zapasowej bazy danych MySQL	292
Przywracanie bazy danych MySQL	293
Implementowanie replikacji	293
Konfigurowanie serwera nadrzędnego	294
Transfer danych początkowych	294
Konfigurowanie odbiorcy lub odbiorców	295
Propozycje dalszych lektur	296
W następnym rozdziale	296
 Rozdział 13. Zaawansowane programowanie w MySQL	 297
Instrukcja LOAD DATA INFILE	297
Maszyny zapisu	298
Transakcje	299
Definicje dotyczące transakcji	299
Użycie transakcji w InnoDB	300
Klucze obce	301
Procedury składowane	302
Prosty przykład	302
Zmienne lokalne	304
Kursory i struktury sterujące	305
Propozycje dalszych lektur	308
W następnym rozdziale	308
 Część III E-commerce i bezpieczeństwo	 309
Rozdział 14. Komercyjne witryny internetowe	311
Co chcemy osiągnąć?	311
Rodzaje komercyjnych stron WWW	311
Publikowanie informacji w broszurach internetowych	312
Przyjmowanie zamówień na produkty i usługi	315
Dostarczanie usług lub wyrobów mających postać cyfrową	319

	Zwiększanie wartości produktów i usług	319
	Ograniczanie kosztów	320
	Ryzyko i zagrożenia	320
	Crackerzy	321
	Przyciągnięcie niewystarczającej liczby klientów	321
	Awarie sprzętu komputerowego	322
	Awarie sieci elektrycznych, komunikacyjnych i komputerowych oraz systemu wysyłkowego	322
	Silna konkurencja	322
	Błędy w oprogramowaniu	323
	Zmiany polityki rządowej	323
	Ograniczenie pojemności systemów	323
	Wybór strategii	323
	W następnym rozdziale	324
Rozdział 15.	Bezpieczeństwo komercyjnych stron WWW	325
	Jaką wagę mają posiadane przez nas informacje?	326
	Zagrożenia bezpieczeństwa	326
	Ujawnienie informacji poufnych	327
	Utrata lub zniszczenie danych	328
	Modyfikacje danych	329
	Blokada usługi	330
	Błędy w oprogramowaniu	331
	Zaprzeczenie korzystania z usługi	332
	Użyteczność, wydajność, koszty i bezpieczeństwo	333
	Opracowanie polityki bezpieczeństwa	333
	Zasady uwierzytelniania	334
	Podstawy szyfrowania	335
	Szyfrowanie z kluczem prywatnym	336
	Szyfrowanie z kluczem publicznym	337
	Podpis cyfrowy	338
	Certyfikaty cyfrowe	339
	Bezpieczne serwery WWW	339
	Monitorowanie i zapisywanie zdarzeń	340
	Zapory sieciowe	341
	Tworzenie kopii zapasowych	342
	Tworzenie kopii zapasowych zwykłych plików	342
	Tworzenie kopii zapasowych i odzyskiwanie baz danych MySQL	342
	Bezpieczeństwo fizyczne	343
	W następnym rozdziale	343
Rozdział 16.	Bezpieczeństwo aplikacji internetowych	345
	Strategie zapewniania bezpieczeństwa	345
	Planowanie z wyprzedzeniem	346
	Równowaga między bezpieczeństwem i użytecznością	346
	Monitorowanie bezpieczeństwa	347
	Ogólne podejście do bezpieczeństwa	347
	Rozpoznawanie zagrożeń	347
	Dostęp do danych poufnych i ich modyfikowanie	347
	Utrata lub zniszczenie danych	348
	Zablokowanie usługi	348
	Wstrzykiwanie kodu	349
	Złamanie zabezpieczeń dostępu do serwera	349
	Identyfikacja użytkowników	349
	Crackerzy	350
	Nieświadomi użytkownicy zainfekowanych komputerów	350

Rozczarowani pracownicy	350
Złodzieje sprzętu komputerowego	350
Autorzy systemów	350
Zabezpieczanie kodu źródłowego	351
Filtrowanie danych pochodzących od użytkowników	351
Unieważnianie danych wynikowych	355
Organizacja kodu źródłowego	356
Zawartość kodu źródłowego	357
Zagadnienia dotyczące systemu plików	358
Stabilność kodu i błędy	358
Apostrofy wykonywania poleceń systemu operacyjnego i polecenie exec	359
Zabezpieczanie serwera WWW oraz PHP	360
Regularne uaktualnianie oprogramowania	361
Analiza ustawień w pliku php.ini	362
Konfiguracja serwera WWW	362
Aplikacje internetowe działające na serwerach komercyjnych	364
Bezpieczeństwo serwera bazy danych	365
Użytkownicy i system uprawnień	365
Wysyłanie danych do serwera	366
Łączenie się z serwerem	366
Praca serwera	367
Zabezpieczanie sieci	367
Instalacja zapory sieciowej	368
Wykorzystanie strefy zdemilitaryzowanej	368
Przygotowanie na ataki DoS i DDoS	369
Bezpieczeństwo komputerów i systemów operacyjnych	369
Uaktualnianie systemu operacyjnego	369
Udostępnianie tylko niezbędnych usług	370
Fizyczne zabezpieczenie serwera	370
Planowanie działań na wypadek awarii	371
W następnym rozdziale	372
Rozdział 17. Uwierzytelnianie przy użyciu PHP i MySQL	373
Identyfikacja użytkowników	373
Implementacja kontroli dostępu	374
Przechowywanie haseł dostępu	376
Szyfrowanie haseł	378
Zastrzeżenie więcej niż jednej strony	379
Podstawowa metoda uwierzytelniania	380
Wykorzystanie podstawowej metody uwierzytelniania w PHP	381
Wykorzystanie podstawowej metody uwierzytelniania na serwerze Apache przy użyciu plików .htaccess	383
Wykorzystanie modułu mod_auth_mysql do celów uwierzytelniania	386
Instalacja modułu mod_auth_mysql	386
Praca z mod_auth_mysql	387
Implementacja własnej metody uwierzytelniania	388
Propozycje dalszych lektur	388
W następnym rozdziale	389
Rozdział 18. Zabezpieczanie transakcji przy użyciu PHP i MySQL	391
Zapewnienie bezpieczeństwa transakcji	391
Komputer użytkownika	392
Internet	393
System docelowy	394
Wykorzystanie protokołu Secure Sockets Layer (SSL)	395
Kontrola danych pochodzących od użytkownika	398

Bezpieczne przechowywanie danych	399
Ustalanie, czy powinno się przechowywać numery kart kredytowych	400
Szyfrowanie danych w PHP	401
Instalacja GPG	401
Testowanie GPG	404
Propozycje dalszych lektur	408
W następnej części	408
Część IV Zaawansowane techniki PHP	409
Rozdział 19. Interakcja z systemem plików i serwerem	411
Wprowadzenie do wysyłania plików	411
Kod HTML służący do wysyłania plików	412
Tworzenie PHP obsługującego plik	413
Najczęściej spotykane problemy	417
Stosowanie funkcji katalogowych	418
Odczyt z katalogów	418
Otrzymywanie informacji na temat aktualnego katalogu	421
Tworzenie i usuwanie katalogów	421
Interakcja z systemem plików	422
Otrzymywanie informacji o pliku	422
Zmiana właściwości pliku	424
Tworzenie, usuwanie i przenoszenie plików	425
Stosowanie funkcji uruchamiających programy	425
Interakcja ze środowiskiem: funkcje getenv() i putenv()	427
Propozycje dalszych lektur	428
W następnym rozdziale	428
Rozdział 20. Stosowanie funkcji sieci i protokołu	429
Przegląd protokołów	429
Wysyłanie i odczytywanie poczty elektronicznej	430
Korzystanie z danych z innych witryn WWW	430
Stosowanie funkcji połączeń sieciowych	433
Tworzenie kopii bezpieczeństwa lub kopii lustrzanej pliku	436
Stosowanie FTP w celu utworzenia kopii bezpieczeństwa lub kopii lustrzanej pliku	436
Wysyłanie plików	442
Unikanie przekroczenia dopuszczalnego czasu	442
Stosowanie innych funkcji FTP	443
Propozycje dalszych lektur	443
W następnym rozdziale	444
Rozdział 21. Zarządzanie datą i czasem	445
Uzyskiwanie informacji o dacie i czasie w PHP	445
Stosowanie funkcji date()	445
Obsługa znaczników czasu Uniksa	447
Stosowanie funkcji getdate()	448
Sprawdzanie poprawności dat przy użyciu funkcji checkdate()	449
Formatowanie znaczników czasu	450
Konwersja pomiędzy formatami daty PHP i MySQL	450
Obliczanie dat w PHP	452
Obliczanie dat w MySQL	454
Stosowanie mikrosekund	455
Stosowanie funkcji kalendarzowych	455
Propozycje dalszych lektur	456
W następnym rozdziale	456

Rozdział 22. Generowanie obrazków	457
Konfigurowanie obsługi obrazków w PHP	457
Formaty obrazków	458
JPEG	459
PNG	459
WBMP	459
GIF	459
Tworzenie obrazków	460
Tworzenie kadru obrazka	460
Rysowanie lub umieszczanie tekstu w obrazku	461
Wyświetlanie ostatecznej grafiki	463
Końcowe czynności porządkujące	464
Stosowanie automatycznie generowanych obrazków na innych stronach	464
Stosowanie tekstu i czcionek do tworzenia obrazków	465
Konfiguracja podstawowego kadru	467
Dopasowanie tekstu do przycisku	468
Nadawanie tekstowi odpowiedniej pozycji	470
Wpisywanie tekstu do przycisku	471
Etap końcowy	471
Rysowanie figur i wykresów danych	471
Inne funkcje obrazków	478
Propozycje dalszych lektur	478
W następnym rozdziale	478
Rozdział 23. Stosowanie kontroli sesji w PHP	479
Czym jest kontrola sesji?	479
Podstawowa zasada działania sesji	479
Czym jest cookie?	480
Konfiguracja cookies w PHP	480
Stosowanie cookies w sesji	481
Przechowywanie identyfikatora sesji	481
Implementacja prostych sesji	482
Rozpoczynanie sesji	482
Zgłaszanie zmiennych sesji	482
Stosowanie zmiennych sesji	483
Usuwanie zmiennych i niszczenie sesji	483
Przykład prostej sesji	483
Konfiguracja kontroli sesji	485
Implementacja uwierzytelniania w kontroli sesji	485
Propozycje dalszych lektur	491
W następnym rozdziale	491
Rozdział 24. Inne przydatne własności	493
Stosowanie magicznych cudzysłowów	493
Wykonywanie ciągów — funkcja eval()	494
Zakończenie wykonania — die i exit	495
Serializacja zmiennych i obiektów	495
Pobieranie informacji na temat środowiska PHP	496
Uzyskiwanie informacji na temat załadowanych rozszerzeń	496
Identyfikacja właściciela skryptu	497
Uzyskiwanie informacji na temat daty modyfikacji skryptu	497
Czasowa zmiana środowiska wykonawczego	497
Podświetlanie źródeł	498
Używanie PHP w wierszu poleceń	499
W następnej części	500

Część V	Tworzenie praktycznych projektów PHP i MySQL	501
Rozdział 25.	Stosowanie PHP i MySQL w dużych projektach	503
	Zastosowanie inżynierii oprogramowania w tworzeniu aplikacji WWW	504
	Planowanie i prowadzenie projektu aplikacji WWW	504
	Ponowne stosowanie kodu	505
	Tworzenie kodu łatwego w utrzymaniu	506
	Standardy kodowania	506
	Dzielenie kodu	509
	Stosowanie standardowej struktury katalogów	509
	Dokumentacja i dzielenie wewnętrznych funkcji	510
	Implementacja kontroli wersji	510
	Wybór środowiska programistycznego	511
	Dokumentacja projektów	511
	Prototypowanie	512
	Oddzielanie logiki i zawartości	513
	Optymalizacja kodu	514
	Stosowanie prostych optymalizacji	514
	Stosowanie produktów firmy Zend	514
	Testowanie	515
	Propozycje dalszych lektur	516
	W następnym rozdziale	516
Rozdział 26.	Usuwanie błędów	517
	Błędy programistyczne	517
	Błędy składni	517
	Błędy wykonania	519
	Błędy logiczne	523
	Pomoc w usuwaniu błędów w zmiennych	525
	Poziomy zgłaszania błędów	527
	Zmiana ustawień zgłaszania błędów	528
	Wyzwalanie własnych błędów	529
	Elegancka obsługa błędów	529
	W następnym rozdziale	532
Rozdział 27.	Tworzenie uwierzytelniania użytkowników i personalizacji	533
	Składniki rozwiązania	533
	Identyfikacja użytkownika i personalizacja	534
	Przechowywanie zakładek	535
	Rekomendowanie zakładek	535
	Przegląd rozwiązania	535
	Implementacja bazy danych	537
	Implementacja podstawowej witryny	538
	Implementacja uwierzytelniania użytkowników	540
	Rejestracja użytkowników	540
	Logowanie	545
	Wylogowanie	548
	Zmiana hasła	549
	Ustawianie zapomnianych haseł	551
	Implementacja przechowywania i odczytywania zakładek	555
	Dodawanie zakładek	555
	Wyświetlanie zakładek	557
	Usuwanie zakładek	557
	Implementacja rekomendacji	559
	Rozwijanie projektu i możliwe rozszerzenia	562
	W następnym rozdziale	562

Rozdział 28. Tworzenie koszyka na zakupy	563
Składniki rozwiązania	563
Tworzenie katalogu online	564
Śledzenie zakupów użytkownika podczas przeglądania	564
Implementacja systemu płatności	564
Interfejs administratora	565
Przegląd rozwiązania	565
Implementacja bazy danych	568
Implementacja katalogu online	570
Przedstawianie kategorii	571
Wyświetlanie książek danej kategorii	574
Przedstawianie szczegółowych danych książki	575
Implementacja koszyka na zakupy	577
Stosowanie skryptu pokaz_kosz.php	577
Podgląd koszyka	580
Dodawanie produktów do koszyka	582
Zapisywanie uaktualnionego koszyka	583
Wyświetlanie podsumowania w pasku nagłówka	584
Pobyty w kasie	584
Implementacja płatności	589
Implementacja interfejsu administratora	591
Rozwijanie projektu	598
Zastosowanie istniejącego systemu	598
W następnym rozdziale	598
Rozdział 29. Tworzenie serwisu poczty elektronicznej opartego na WWW	599
Składniki rozwiązania	599
Protokoły poczty: POP3 i IMAP	599
Obsługa POP3 i IMAP w PHP	600
Przegląd rozwiązania	601
Konfiguracja bazy danych	603
Architektura skryptu	604
Logowanie i wylogowanie	608
Konfiguracja kont	611
Tworzenie nowego konta	613
Modyfikacja istniejącego konta	614
Usuwanie konta	614
Odczytywanie poczty	615
Wybór konta	615
Przeglądanie zawartości skrzynki	617
Odczytywanie wiadomości pocztowych	619
Przeglądanie nagłówków wiadomości	622
Usuwanie wiadomości	623
Wysyłanie wiadomości	623
Wysyłanie nowej wiadomości	624
Odpowiadanie i przekazywanie poczty	625
Rozwijanie projektu	626
W następnym rozdziale	627
Rozdział 30. Tworzenie menedżera list pocztowych	629
Składniki rozwiązania	629
Konfiguracja bazy danych list i abonentów	630
Wysyłanie plików	630
Wysyłanie wiadomości z załącznikami	631
Przegląd rozwiązania	631
Konfiguracja bazy danych	633

Architektura skryptu	635
Implementacja logowania	641
Tworzenie nowego konta	641
Logowanie	643
Implementacja funkcji użytkownika	645
Przeglądanie list	646
Przeglądanie informacji na temat listy	650
Przeglądanie archiwum listy	652
Zapisywanie i wypisywanie	653
Zmiana konfiguracji konta	654
Zmiana hasła	654
Wylogowanie	656
Implementacja funkcji administratora	656
Tworzenie nowej listy	657
Wysyłanie nowych wiadomości	658
Obsługa wysyłania wielu plików	661
Podgląd wiadomości	664
Rozsyłanie wiadomości	665
Rozwijanie projektu	670
W następnym rozdziale	670
Rozdział 31. Tworzenie forum WWW	671
Proces	671
Składniki rozwiązania	672
Przegląd rozwiązania	673
Projektowanie bazy danych	674
Przeglądanie drzewa artykułów	676
Rozwijanie i zwijanie	678
Wyświetlanie artykułów	680
Korzystanie z klasy <code>wezel_drzewa</code>	681
Przeglądanie pojedynczych artykułów	687
Dodawanie nowych artykułów	688
Rozszerzenia	694
Wykorzystanie istniejącego systemu	694
W następnym rozdziale	694
Rozdział 32. Tworzenie dokumentów spersonalizowanych w formacie PDF	695
Opis projektu	695
Ocena formatów dokumentów	696
Składniki rozwiązania	700
System pytań i odpowiedzi	700
Oprogramowanie generujące dokumenty	700
Przegląd rozwiązania	703
Zadawanie pytań	704
Ocena odpowiedzi	705
Tworzenie certyfikatu RTF	707
Tworzenie certyfikatu PDF z szablonu	710
Generowanie dokumentu PDF za pomocą PDFlib	713
Skrypt „Witaj, świecie” dla PDFlib	713
Tworzenie certyfikatu za pomocą PDFlib	717
Problemy związane z nagłówkami	723
Rozwijanie projektu	724
W następnym rozdziale	724

Rozdział 33.	Korzystanie z usług sieciowych za pomocą XML i SOAP	725
	Opis projektu: korzystanie z języka XML i usług sieciowych	725
	Podstawy XML	726
	Podstawy usług sieciowych	729
	Składniki rozwiązania	730
	Korzystanie z interfejsu usług sieciowych Amazon.com	730
	Wczytywanie dokumentów XML: odpowiedzi REST	731
	Korzystanie z SOAP za pomocą PHP	732
	Buforowanie	732
	Opis rozwiązania	732
	Aplikacja główna	734
	Wyświetlanie listy książek z danej kategorii	742
	Tworzenie obiektu klasy WynikiWyszukiwania	743
	Korzystanie z REST do wykonywania żądań i odczytywania wyników	750
	Korzystanie z protokołu SOAP do wykonywania żądania i odczytywania wyniku	755
	Buforowanie danych pochodzących z żądania	756
	Konstrukcja koszyka na zakupy	758
	Przejsie do kasy na witrynie Amazon.com	761
	Instalacja kodu źródłowego	762
	Kierunki rozwoju	762
	Literatura	762
Rozdział 34.	Tworzenie aplikacji Web 2.0 z wykorzystaniem technologii Ajax	763
	Czym jest technologia Ajax?	764
	Żądania i odpowiedzi HTTP	764
	DHTML i XML	765
	Kaskadowe arkusze stylów (CSS)	766
	Skrypty działające po stronie klienta	767
	Skrypty działające po stronie serwera	768
	XML i XSLT	768
	Podstawy technologii Ajax	768
	Obiekt XMLHttpRequest	768
	Komunikowanie się z serwerem	770
	Przetwarzanie odpowiedzi serwera	772
	Połączenie wszystkich elementów aplikacji	773
	Dodanie nowych elementów do wcześniejszych projektów	775
	Dodanie elementów Ajaksa do witryny ZakładkaPHP	777
	Źródła dodatkowych informacji	788
	Dodatkowe informacje na temat Document Object Model (DOM)	789
	Biblioteki JavaScript dla aplikacji Ajax	789
	Witryny internetowe przeznaczone dla programistów Ajax	790
Dodatki	791	
Dodatek A	Instalacja PHP i MySQL	793
	Instalacja Apache, PHP i MySQL w systemie UNIX	794
	Instalacja przy użyciu binariów	794
	Instalacja przy użyciu kodów źródłowych	794
	Plik httpd.conf — informacje końcowe	800
	Czy obsługa PHP działa poprawnie?	800
	Czy SSL działa poprawnie?	801

Instalacja Apache, PHP i MySQL w systemie Windows	802
Instalacja MySQL w systemie Windows	803
Instalacja serwera Apache w systemie Windows	804
Instalacja PHP w systemie Windows	806
Instalowanie PEAR	808
Inne konfiguracje	809
Dodatek B	
Zasoby internetowe	811
Zasoby poświęcone PHP	811
Zasoby poświęcone MySQL i SQL	813
Zasoby poświęcone serwerowi Apache	813
Zasoby poświęcone tworzeniu stron WWW	814
Skorowidz	815

Rozdział 2.

Przechowywanie i wyszukiwanie danych

W poprzednim rozdziale omówiliśmy sposoby dostępu do danych umieszczonych w formularzu HTML i metody manipulowania nimi. Teraz przedstawiamy metody zapisywania informacji w celu późniejszego ich wykorzystania. W większości przypadków, włączając w to przykład z poprzedniego rozdziału, celem jest przechowanie danych i późniejsze ich załadowanie. W tym przykładzie należy zapamiętać zamówienie klienta, aby później je zrealizować.

W tym rozdziale opiszemy sposoby zapisania do pliku zamówienia przedstawionego w przykładzie oraz metody późniejszego odczytania tego pliku. Pokażemy również, dlaczego takie rozwiązanie nie zawsze jest najlepsze. Pracując z większą liczbą zamówień, powinno się zamiast niego używać systemu zarządzania bazami danych, takiego jak MySQL.

W tym rozdziale zostaną poruszone następujące zagadnienia:

- zapisywanie danych do późniejszego użycia,
- otwieranie pliku,
- tworzenie i zapisywanie pliku,
- zamykanie pliku,
- czytanie z pliku,
- blokowanie pliku,
- usuwanie pliku,
- inne przydatne informacje na temat plików,
- lepszy sposób obróbki danych: systemy zarządzania bazami danych.

Zapisywanie danych do późniejszego użycia

Istnieją dwa sposoby przechowywania danych — w pliku jednorodnym oraz w bazie danych.

Plik jednorodny może mieć wiele różnych formatów, lecz zazwyczaj terminem tym oznacza się prosty plik tekstowy. W opisywanym przykładzie dane są zapisywane w pliku tekstowym, jedno zamówienie w jednym wierszu.

Zapisywanie zamówień w taki właśnie sposób jest rozwiązaniem bardzo prostym w realizacji, ale zarazem jest ono obciążone licznymi ograniczeniami, co zostanie pokazane w dalszej części rozdziału. Przy obróbce danych znacznej wielkości stosuje się zazwyczaj bazy danych. Mimo to pliki jednorodne znajdują zastosowania i istnieją przypadki, w których wiedza na ich temat jest konieczna.

Proces zapisu i odczytu plików w PHP przebiega w zbliżony sposób jak w wielu innych, podobnych językach programowania. Osoby znające język C lub skrypty powłoki Uniksa powinny bez trudu rozpoznać podobieństwa tych procedur.

Przechowywanie i wyszukiwanie zamówień Janka

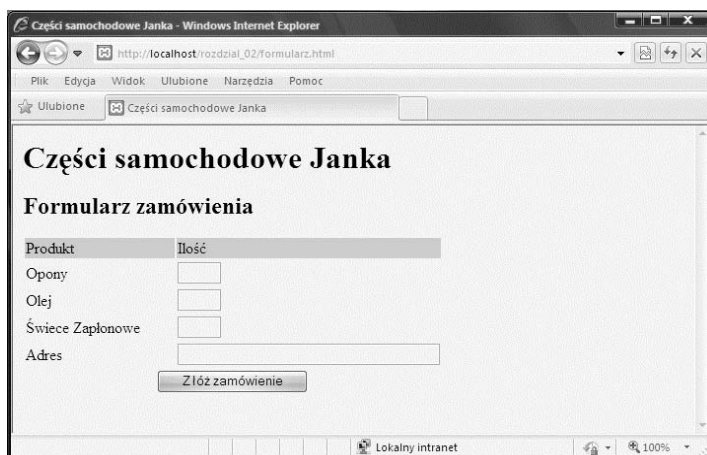
Poniżej użyta zostanie nieco zmodyfikowana wersja formularza zamówień, przedstawionego w poprzednim rozdziale. Na początku należy przeanalizować ten formularz i kod PHP stworzony w celu obróbki zamówień.



Kod HTML i skrypty PHP zastosowane w tym rozdziale znajdują się w folderze *rozdzial_02* (przykłady są dostępne na płycie CD dołączonej do książki).

Formularz został zmodyfikowany w celu łatwego uzyskania adresu klienta. Nowa wersja formularza jest przedstawiona na rysunku 2.1.

Rysunek 2.1.
Wersja formularza zamówień pobierająca również adres klienta



Pole formularza zawierające adres klienta nosi nazwę `adres`. Podczas przetwarzania w PHP daje ono zmienną o nazwie `$adres`, pod warunkiem stosowania stylu krótkiego dostępu do zmiennej. Należy pamiętać, że przy zastosowaniu stylu długiego odwołanie do tej zmiennej to `$REQUEST['adres']`, `$_POST['adres']` lub `$_GET['adres']`, zależnie od metody zatwierdzenia (szczegóły opisaliśmy w rozdziale 1.).

W tym rozdziale każde nadchodzące zamówienie zostanie zapisane w tym samym pliku. Skonstruowany później interfejs WWW pozwoli pracownikom Janka na przeglądanie przyjętych zamówień.

Przetwarzanie plików

Zapisywanie danych w pliku następuje w trzech etapach. Są to:

1. Otwarcie pliku. Jeżeli dany plik nie istnieje, należy go utworzyć.
2. Zapisanie danych w pliku.
3. Zamknięcie pliku.

Podobnie, trój etapowo, przebiega odczytywanie danych z pliku:

1. Otwarcie pliku. Jeżeli plik nie może zostać otwarty (np. nie istnieje), fakt ten musi zostać rozpoznany i program powinien zakończyć się w elegancki sposób (tzn. nie bombardując użytkownika dokładnymi i niepotrzebnymi mu informacjami o błędach).
2. Odczytanie danych z pliku.
3. Zamknięcie pliku.

Przy odczytywaniu danych z pliku dostępnych jest wiele sposobów ustalania ilości pobieranych naraz danych. Rozwiązania najczęściej stosowane zostaną opisane bardziej szczegółowo w poniższych punktach. Na początek przedstawimy mechanizm otwierania plików.

Otwieranie pliku

Aby otworzyć plik w PHP, stosuje się funkcję `fopen()`. Otwierając plik, należy zadeklarować sposób, w jaki będzie on używany. Sposób ten nosi nazwę *trybu otwarcia pliku*.

Tryby otwarcia pliku

System operacyjny serwera musi mieć informacje na temat przeznaczenia otwieranego pliku. Musi wiedzieć, czy plik może równocześnie zostać otwarty przez inny skrypt oraz czy użytkownik posiada uprawnienia do dostępu i modyfikacji pliku. Przede wszystkim tryb otwarcia pliku dostarcza systemowi operacyjnemu mechanizmu przetwarzania żądań dostępu od innych użytkowników bądź skryptów oraz metody sprawdzania uprawnień dostępu do konkretnych plików.

Przy otwieraniu pliku należy mieć trzy informacje:

1. Można otworzyć plik w następujących trybach: tylko do odczytu, tylko do zapisu lub do obu tych celów.
2. Przy zapisywaniu danych w pliku można nadpisać istniejące dane bądź dodać nowe na jego końcu. Można również opracować zgrabny sposób zakańczania programu zamiast nadpisywania pliku na pliku, który już istnieje.
3. Przy zapisywaniu pliku przy użyciu systemu rozróżniającego pliki tekstowe i binarne można określić dany typ.

Funkcja `fopen()` rozpoznaje połączenia tych trzech opcji.

Stosowanie funkcji `fopen()` do otwarcia pliku

Aby zapisać zamówienie klienta do pliku zamówień Janka, należy zastosować następujący wiersz kodu:

```
$wp = fopen("$DOCUMENT_ROOT/./zamowienia/zamowienia.txt", 'w');
```

Przy wywołaniu funkcja `fopen` spodziewa się dwóch lub trzech parametrów. Zazwyczaj stosuje się dwa, jak pokazano w powyższym przykładzie.

Pierwszy parametr to nazwa pliku, który ma zostać otwarty. Można tu określić ścieżkę dostępu do pliku, jak w powyższym przykładzie; plik *zamowienia.txt* znajduje się w katalogu zamówień. Zastosowana została wbudowana w PHP zmienna `$SERVER['DOCUMENT_ROOT']` lecz, ze względu na uciążliwość stosowania pełnych nazw zmiennych formy, przypisaliliśmy jej krótszą nazwę.

Zmienna ta wskazuje na podstawowy element drzewa katalogów serwera WWW. W wierszu tym użyto symbolu `..`, oznaczającego „katalog nadrzędny katalogu macierzystego”, który ze względu na bezpieczeństwo znajduje się poza drzewem katalogów. Nie można pozwolić na inny sposób dostępu przez WWW do tego pliku poza dostarczonym interfejsem. Ścieżka tego typu jest nazywana *ścieżką względną*, ponieważ opisuje miejsce w systemie plików w zależności od katalogu macierzystego.

Podobnie jak w przypadku nadawania zmiennym formy krótkich nazw, na początku skryptu należy umieścić następujący wiersz:

```
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
```

w celu skopiowania zawartości zmiennej noszącej nazwę długą do zmiennej o krótkiej nazwie.

Analogicznie do różnorodności metod dostępu do danych formy, istnieją również różne sposoby dostępu do predefiniowanych zmiennych serwera. W zależności od ustawień serwera (szczegółowe informacje na ten temat można znaleźć w rozdziale 1.) można pozyskać katalog macierzysty za pomocą:

- `$_SERVER['DOCUMENT_ROOT']`
- `$DOCUMENT_ROOT`
- `$HTTP_SERVER_VARS['DOCUMENT_ROOT']`

Podobnie jak w przypadku danych formy, zalecany jest pierwszy sposób.

Można również określić bezwzględną ścieżkę dostępu do pliku, będącą ścieżką od katalogu głównego (`/` w systemach Unix i zazwyczaj `C:\` w systemach Windows). Na przykładowym serwerze Uniksa ścieżka ta może wyglądać następująco: `/home/ksiazka/zamowienia`. Niedogodnością tej metody, zwłaszcza w wypadku korzystania z obcego serwera, jest możliwość modyfikacji ścieżki bezwzględnej, co może oznaczać poważne zmiany w wielu skryptach.

Jeżeli ścieżka nie zostanie podana, PHP będzie szukał pliku i ewentualnie utworzy go w tym samym katalogu, w którym znajduje się skrypt. Może się to różnić w zależności od faktu, czy PHP jest uruchamiany poprzez jakiś skrypt CGI, i zależy od konfiguracji serwera.

W środowisku Uniksa stosuje się ukośniki (`/`), natomiast w środowisku Windows można używać lewych (`\`) lub prawych ukośników (`/`), które muszą jednak zostać oznaczone jako znaki specjalne, aby funkcja `fopen` właściwie je zinterpretowała. W tym celu należy po prostu dodać przed każdym symbolem jeszcze jeden lewy ukośnik, jak pokazano w poniższym przykładzie:

```
$wp = fopen("..\..\zamowienia\\zamowienia.txt", 'w');
```

Niewiele osób stosuje w ścieżkach dostępu w kodzie PHP znaki odwrotnych ukośników, ponieważ oznaczałoby to, że kod ten będzie działał tylko w systemach Windows. Stosowanie zwykłych ukośników pozwala na przenoszenie kodu między maszynami pracującymi w systemach Unix i Windows bez konieczności wprowadzania w nim zmian.

Drugim parametrem funkcji `fopen()` jest tryb otwarcia pliku, określający jego przeznaczenie. Powinien on zostać podany jako ciąg. W powyższym przykładzie funkcji `fopen()` zostaje przekazana wartość `w`, co oznacza otwarcie pliku do zapisu. Podsumowanie trybów otwarcia pliku przedstawiono w tabeli 2.1.

Tabela 2.1. Podsumowanie trybów otwarcia pliku w funkcji `fopen`

Tryb	Nazwa trybu	Znaczenie
r	Odczyt	Otwarcie pliku do odczytu, poczynając od początku pliku
r+	Odczyt	Otwarcie pliku do odczytu i zapisu, poczynając od początku pliku
w	Zapis	Otwarcie pliku do zapisu, poczynając od początku pliku. Jeżeli plik istnieje, bieżąca zawartość zostanie skasowana. W przeciwnym wypadku nastąpi próba jego utworzenia
w+	Zapis	Otwarcie pliku do zapisu i odczytu, poczynając od początku pliku. Jeżeli plik istnieje, bieżąca zawartość zostanie skasowana, jeżeli zaś nie, nastąpi próba jego utworzenia
x	Ostrożny zapis	Otwarcie pliku do zapisu rozpoczynającego się na początku pliku. Jeśli plik już istnieje, nie zostanie otwarty, funkcja <code>fopen()</code> zwróci wartość <code>false</code> , a PHP wygeneruje ostrzeżenie
x+	Ostrożny zapis	Otwarcie pliku do zapisu i odczytu rozpoczynającego się na początku pliku. Jeśli plik już istnieje, nie zostanie otwarty, funkcja <code>fopen()</code> zwróci wartość <code>false</code> , a PHP wygeneruje ostrzeżenie
a	Dodawanie	Otwarcie pliku do dodawania zawartości, począwszy od końca istniejącej zawartości. Jeżeli plik nie istnieje, nastąpi próba jego utworzenia
a+	Dodawanie	Otwarcie pliku do dodawania zawartości i odczytu, począwszy od końca istniejącej zawartości. Jeżeli plik nie istnieje, nastąpi próba jego utworzenia
b	Tryb binarny	Stosowany w połączeniu z jednym z powyższych typów w wypadku korzystania z systemu rozróżniającego pliki tekstowe i binarne. Windows go rozróżnia, Unix — nie. Programiści PHP zalecają, by zawsze używać tej opcji w celu zapewnienia sobie maksymalnej przenośności. Jest to tryb domyślny
t	Tryb tekstowy	Stosowany w połączeniu z jednym z powyższych trybów. Tryb ten jest dostępny jedynie w systemie Windows. Nie jest on zalecany, chyba że przed przeniesieniem kodu zostanie zamieniony na tryb <code>b</code>

Tryb otwarcia pliku zastosowany w przykładzie zależy od sposobu, w jaki system zostanie użyty. Powyżej występuje tryb `'w'`, co oznacza, że w pliku będzie mogło być zapamiętane tylko jedno zamówienie. Każde nowo przyjęte zamówienie nadpisze poprzednie. Nie jest to rozwiązanie zbyt rozsądne, więc lepiej użyć trybu dodawania (oraz, zgodnie z zaleceniem, trybu binarnego):

```
$wp = fopen("$DOCUMENT_ROOT/./zamowienia/zamowienia.txt", 'ab');
```

Istnieje również trzeci, opcjonalny parametr funkcji `fopen()`. Stosuje się go w celu szukania pliku w lokalizacjach podanych w opcji `include_path` (ustawianej w konfiguracji PHP — szczegóły w dodatku A). Aby użyć tej opcji, należy nadać temu parametrowi wartość `1`. Nie trzeba wtedy podawać ścieżki dostępu do pliku.

```
$wp = fopen('zamowienia.txt', 'ab', true);
```

Czwarty parametr również jest opcjonalny. Funkcja `fopen()` dopuszcza, by nazwy plików były poprzedzone nazwą protokołu (na przykład `http://`), a same pliki były otwierane ze zdalnych

lokalizacji. Niektóre protokoły pozwalają ponadto na przekazywanie dodatkowych parametrów. Taki sposób użycia funkcji `fopen()` zostanie opisany bardziej szczegółowo w dalszej części tego rozdziału.

Jeżeli funkcji `fopen()` uda się otwarcie pliku, zwraca ona zasób będący w rzeczywistości uchwytem albo wskaźnikiem pliku i przechowuje go w zmiennej, w powyższym przykładzie: `$wp`. Zmienna ta jest stosowana przy kolejnych próbach dostępu do pliku, to znaczy przy odczytywaniu lub zapisywaniu danych.

Otwieranie pliku przez protokół FTP lub HTTP

Funkcja `fopen()` służy do otwierania do odczytu lub zapisu plików lokalnych. Za jej pomocą można także otwierać pliki poprzez FTP, HTTP i inne protokoły. Własność tę można zablokować, wyłączając w pliku `php.ini` dyrektywę `allow_url_fopen`. Jeżeli więc otwieranie plików zdalnych przy użyciu `fopen()` sprawia kłopoty, najpierw należy zajrzeć do pliku `php.ini`.

Jeżeli wprowadzona nazwa pliku rozpoczyna się od `ftp://`, otwarte zostanie pasywne połączenie FTP z serwerem, którego adres został wprowadzony, a funkcja zwróci wartość wskaźnika na początek pliku.

Jeżeli wprowadzona nazwa pliku rozpoczyna się od `http://`, otwarte zostanie pasywne połączenie HTTP z serwerem, którego adres został wprowadzony, a funkcja zwróci wartość wskaźnika na odpowiedź. Przy zastosowaniu trybu HTTP w starszych wersjach PHP adres odnoszący się do katalogu musi zawierać kończące ukośniki, jak w poniższym przykładzie:

```
http://www.serwer.com/
```

a nie

```
http://www.serwer.com
```

Przy zastosowaniu drugiej wersji adresu (bez ukośnika) serwer WWW użyje zwykłego przekierowania HTTP i prześle w odpowiedzi pierwszy z powyższych adresów (warto wykonać to zadanie).

Należy pamiętać, że nazwy domen w URL-ach nie są różnicowane ze względu na wielkość liter, w przeciwieństwie do ścieżek i nazw plików.

Problemy z otwieraniem plików

Popularnym błędem jest próba otwarcia pliku, co do którego nie posiada się praw odczytu lub zapisu. (Błąd taki pojawia się zazwyczaj w systemach operacyjnych z rodziny Unix, od czasu do czasu można jednak spotkać się z nim w systemie Windows.) W takim przypadku PHP wyświetli ostrzeżenie podobne do przedstawionego na rysunku 2.2.

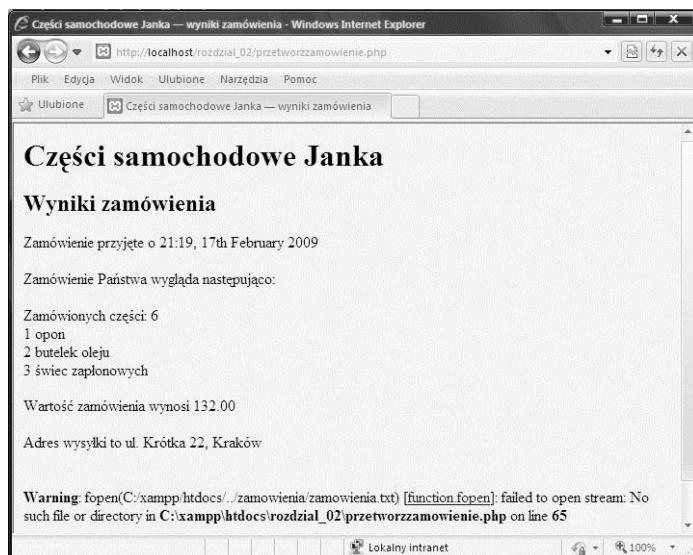
Po popełnieniu takiego błędu należy upewnić się, czy skrypt, który jest stosowany, posiada prawo dostępu do danego pliku. Zależnie od konfiguracji serwera, skrypt może być uruchomiony z prawami użytkownika serwera WWW lub z prawami właściciela swojego katalogu.

W większości systemów skrypt zostanie uruchomiony jako użytkownik serwera WWW. Jeżeli na przykład skrypt znajduje się w systemie uniksowym w katalogu `~/public_html/rozdziel2`, należy utworzyć ogólnodostępny katalog, w którym przechowywane będą zamówienia. Aby to uczynić, można wpisać:

```
mkdir ~/zamowienia  
chmod 777 ~/zamowienia/
```

Rysunek 2.2.

Podczas nieudanej próby otwarcia pliku PHP wyświetla specyficzne ostrzeżenie



Należy pamiętać, że katalogi i pliki z ogólnym prawem zapisu są bardzo niebezpieczne. W szczególności nie powinno się używać katalogów dostępnych bezpośrednio z poziomu WWW, które posiadają możliwość zapisu. Z tego powodu przykładowy katalog *zamowienia* został umieszczony dwa poziomy wyżej, ponad katalogiem *public_html*. Szczegółowe informacje na temat bezpieczeństwa są przedstawione w rozdziale 15.

Złe ustawienia dostępu do plików to najpopularniejszy, lecz nie jedyny błąd popełniany przy otwieraniu plików. Jeżeli plik nie może zostać otwarty, trzeba koniecznie o tym wiedzieć, aby nie próbować odczytywać ani zapisywać w nim danych.

Jeżeli wywołanie funkcji `fopen()` nie powiedzie się, zwróci ona wartość `false`. Można wtedy zastąpić oryginalny komunikat o błędzie PHP innym, bardziej przyjaznym dla użytkownika:

```
@ $wp = fopen("$DOCUMENT_ROOT/./zamowienia/zamowienia.txt", 'ab');

if (!$wp) {
    echo "<p><strong> Zamówienie Państwa nie może zostać przyjęte w tej chwili.
        Proszę spróbować później.</strong></p></body></html>";
    exit;
}
```

Symbol `@` umieszczony przed wywołaniem funkcji `fopen()` nakazuje PHP wytłumienie wszystkich błędów wynikłych z tego wywołania. Zazwyczaj warto wiedzieć, kiedy występuje błąd. Kwestię tę rozważymy później.

Wiersz ten można również zapisać w następujący sposób:

```
$wp = @fopen("$DOCUMENT_ROOT/./ zamowienia/zamowienia.txt", 'a');
```

jednak w takiej sytuacji nie widać wyraźnie, że stosowane jest ukrywanie błędów, co może utrudnić debugowanie kodu.

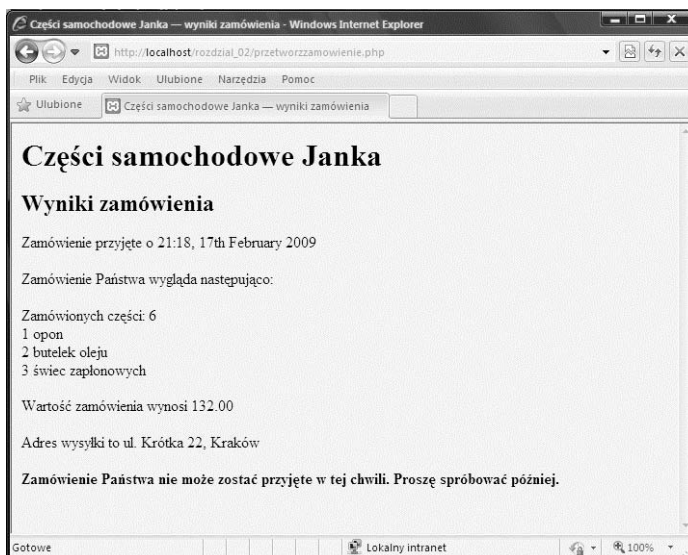
Opisana metoda stanowi najprostszy sposób radzenia sobie z błędami. Bardziej elegancki sposób obsługi błędów zostanie przedstawiony w rozdziale 7.

Instrukcja `if` sprawdza wartość zmiennej `$wp`, aby ustalić, czy wywołanie funkcji `fopen()` zwróciło prawidłowy wskaźnik. Jeżeli nie, wyświetla komunikat o błędzie i kończy działanie skryptu. Ponieważ strona zakończy się w tym miejscu, w powyższym kodzie zamknięte zostały również znaczniki HTML, aby kod HTML działał bezbłędnie.

Wynik działania powyższego fragmentu skryptu został przedstawiony na rysunku 2.3.

Rysunek 2.3.

Stosowanie własnych komunikatów o błędach zamiast tych wbudowanych w PHP jest niewątpliwie bardziej przyjazne dla użytkownika



Zapisywanie danych w pliku

Zapisywanie danych w pliku w PHP jest stosunkowo proste. Stosuje się do tego funkcję `fwrite()` (zapis do pliku) lub `fputs()` (umieszczenie ciągu w pliku). Funkcja `fputs()` jest inną nazwą funkcji `fwrite()`. Funkcję `fwrite()` wywołuje się w następujący sposób:

```
fwrite($wp, $ciągwyjściowy);
```

Funkcja ta nakazuje PHP zapisanie ciągu zawartego w zmiennej `$ciągwyjściowy` do pliku wskazywanego przez zmienną `$wp`.

Alternatywą dla funkcji `fwrite()` jest funkcja `file_put_contents()`. Jej prototyp przedstawia się następująco:

```
int file_put_contents ( string nazwa_pliku,
                      string dane
                      [, int znaczniki
                      [, resource kontekst]])
```

Funkcja zapisuje ciąg znaków zawarty w danych do pliku o nazwie `nazwa_pliku`, bez potrzeby wywoływania funkcji `fopen()` (ani `fclose()`). Jest to nowa funkcja, wprowadzona w PHP5, a funkcją dla niej komplementarną jest `file_get_contents()`, o której niedługo powiemy. Opcjonalnych parametrów `znaczniki` i `kontekst` najczęściej używa się w trakcie zapisywania do plików zdalnych przy użyciu na przykład HTTP i FTP. (Funkcje te zostaną omówione w rozdziale 20.).

Parametry funkcji fwrite()

Funkcja `fwrite()` pobiera trzy parametry, lecz ostatni z nich jest opcjonalny. Oto prototyp funkcji `fwrite()`:

```
int fwrite(resource wskaznik_pliku, string ciag, int [dlugosc]);
```

Trzeci parametr, *dlugosc*, zawiera maksymalną możliwą do zapisania liczbę bajtów. Jeżeli parametr ten został umieszczony w wywołaniu funkcji, `fwrite()` będzie zapisywać *ciag* w pliku wskazanym przez *wskaznik pliku*, dopóki nie osiągnie końca *ciagu* lub zapisze *dlugosc* bajtów, zależnie od tego, co wystąpi wcześniej.

Długość łańcucha znaków można odczytać, używając funkcji PHP o nazwie `strlen()` w następujący sposób:

```
fwrite($wp, $ciagwyjsciowy, strlen($ciagwyjsciowy));
```

Trzeciego parametru używa się w trakcie zapisywania w trybie binarnym, ponieważ można dzięki niemu uniknąć pewnych komplikacji związanych ze zgodnością między platformami.

Formaty plików

Tworząc plik danych podobny do przykładowego, można określić dowolny format przechowywania danych. Jeżeli jednak dane te będą wykorzystywane później przez jakąś aplikację, należy zastosować się do zasad określonych przez tę aplikację.

Poniżej przedstawiono ciąg opisujący jeden rekord w pliku danych:

```
$ciagwyjsciowy = $data."\t".$iloscoPON." opON \t".$iloscoleju." butelek oleju\t"  
                .$iloscswiec." swiec zaplonowych\t".$wartosc  
                ."PLN\t".$adres."\n";
```

W tym prostym przykładzie każdy rekord jest zapisany w osobnym wierszu pliku. Metodę tę zastosowano, ponieważ występuje w niej prosty separator rekordów: znak nowego wiersza. Znaki te przedstawia się za pomocą sekwencji `"\n"`, gdyż są niewidzialne.

W całej książce pola danych będą zapisywane za każdym razem w jednakowym porządku i oddzielane znakami tabulacji. Ponieważ znak ten również jest niewidzialny, przedstawia się go za pomocą sekwencji `"\t"`. Można wybrać dowolny, czytelny znak podziału.

Znak podziału powinien być znakiem, który nie występuje pośród wprowadzanych danych, lub też dane powinny zostać przekształcone w celu usunięcia występujących w nich znaków podziału. Przekształcanie danych zostanie omówione w rozdziale 4. Na razie należy przyjąć, że przy wprowadzaniu zamówienia nie zostanie użyty znak tabulacji, co jest zdarzeniem możliwym, lecz mało prawdopodobnym.

Stosowanie specjalnych znaków separujących pola pozwala na łatwiejsze rozdzielenie zmiennych przy odczytywaniu danych. Kwestia ta zostanie rozważona w rozdziale 3. oraz w rozdziale 4. Tymczasem każde zamówienie będzie traktowane jako pojedynczy ciąg.

Po przyjęciu kilku zamówień zawartość pliku powinna wyglądać podobnie do przedstawionej na listingu 2.1.

Listing 2.1. *zamowienia.txt* — przykład pliku zamówień

```

19:35, 18 lipca 2008 4 opon 1 butelek oleju 6 świec zapłonowych 1820.00PLN
↳ul. Krótka 22, Kraków
19:37, 18 lipca 2008 1 opon 0 butelek oleju 0 świec zapłonowych 400.00PLN
↳ul. Główna 33, Gliwice
19:38, 18 lipca 2008 0 opon 1 butelek oleju 4 świec zapłonowych 180.00PLN
↳ul. Akacjowa 127, Warszawa

```

Zamykanie pliku

Po zakończeniu korzystania z pliku należy go zamknąć. Stosuje się w tym celu funkcję `fclose()` w następujący sposób:

```
fclose($wp);
```

Funkcja ta zwraca wartość `true`, jeżeli zamykanie powiodło się, lub `false`, w przeciwnym wypadku. Działanie to ma znacznie większe szanse powodzenia niż otwieranie pliku — i w tym przypadku nie zdecydowano się na jego sprawdzenie.

Pełen listing ostatecznej wersji skryptu *przetworzzamowienie.php* przedstawiono na listingu 2.2.

Listing 2.2. *przetworzzamowienie.php* — Ostateczna wersja skryptu przetwarzającego zamówienia

```

<?php
// utworzenie krótkich nazw zmiennych
$iloscopon = $_POST['iloscopon'];
$iloscoleju = $_POST['iloscoleju'];
$iloscswiec = $_POST['iloscswiec'];
$adres = $_POST['adres'];
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
$data=date('H:i, jS F Y');
?>

<html>
<head>
<title>Części samochodowe Janka – wyniki zamówienia</title>
</head>
<body>
<h1>Części samochodowe Janka</h1>
<h2>Wyniki zamówienia</h2>
<?php

echo "<p>Zamówienie przyjęte o ".$data."</p>";

echo "<p>Zamówienie Państwa wygląda następująco: </p>";

$ilosc = 0;
$ilosc = $iloscopon + $iloscoleju + $iloscswiec;
echo "Zamówionych części: ".$ilosc."<br />";

if($ilosc == 0) {
    echo "Na poprzedniej stronie nie zostało złożone żadne zamówienie!<br />";
} else {

    if ($iloscopon > 0) {
        echo $iloscopon." opon<br />";

```

```
}

if ($iloscoleju > 0) {
    echo $iloscoleju." butelek oleju<br />";
}

if ($iloscswiec > 0) {
    echo $iloscswiec." świec zapłonowych<br />";
}
}

$wartosc=0.00;

define('CENAOPON', 100);
define('CENAOLEJU', 10);
define('CENASWIEC', 4);

$wartosc=$iloscopon * CENAOPON + $iloscoleju * CENAOLEJU + $iloscswiec * CENASWIEC;

$wartosc=number_format($wartosc, 2, '.', ' ');

echo "<p>Wartość zamówienia wynosi ".$wartosc."</p>";
echo "<p>Adres wysyłki to ".$adres."</p>";

$ciagwyjsciowy = $data."\t".$iloscopon." opon \t".$iloscoleju." butelek oleju\t"
                ".$iloscswiec." swiec zapłonowych\t".$wartosc
                ."PLN\t". $adres."\n";

// otwarcie pliku w celu dopisywania
@ $wp = fopen("$DOCUMENT_ROOT../zamowienia/zamowienia.txt", 'ab');

flock($wp, LOCK_EX);

if (!$wp) {
    echo "<p><strong> Zamówienie Państwa nie może zostać przyjęte w tej chwili.
        Proszę spróbować później.</strong></p></body></html>";
    exit;
}

fwrite($wp, $ciagwyjsciowy, strlen($ciagwyjsciowy));
flock($wp, LOCK_UN);
fclose($wp);

echo "<p>Zamówienie zapisane.</p>";
?>
</body>
</html>
```

Odczyt z pliku

Klienci Janka mogą już składać swoje zamówienia przez sieć WWW, lecz jeżeli pracownicy firmy chcą je obejrzeć, muszą otworzyć plik własnoręcznie.

Można stworzyć interfejs WWW pozwalający pracownikom na łatwe odczytywanie plików. Kod tego interfejsu został przedstawiony na listingu 2.3.

Listing 2.3. *zobaczzamowienia.php* — interfejs pozwalający pracownikom Janka na oglądanie zawartości plików

```

<?php
// utworzenie krótkich nazw zmiennych
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
?>
<html>
<head>
<title>Części samochodowe Janka – zamówienia klientów</title>
</head>
<body>
<h1>Części samochodowe Janka</h1>
<h2>Zamówienia klientów</h2>
<?php

@ $wp = fopen("$DOCUMENT_ROOT/../../zamowienia/zamowienia.txt", 'rb');

if (!$wp) {
    echo "<p><strong>Brak zamówień.
        Proszę spróbować później.</strong></p>";
    exit;
}

while (!feof($wp)) {
    $zamowienie = fgets($wp, 999);
    echo $zamowienie."<br />";
}

fclose($wp);

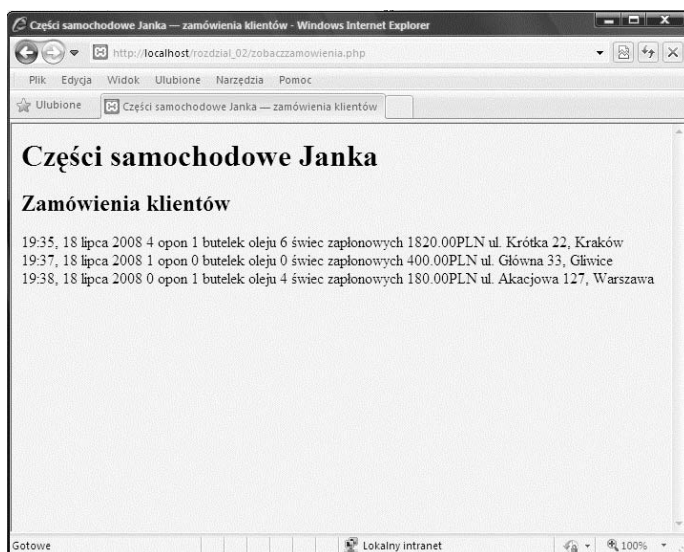
?>
</body>
</html>

```

Skrypt ten działa na zasadzie opisanej powyżej — otwarcie pliku, odczyt z pliku, zamknięcie pliku. Wynik uruchomienia powyższego skryptu, wykorzystującego plik danych z listingu 2.1, jest przedstawiony na rysunku 2.4.

Rysunek 2.4.

Skrypt *zobaczzamowienia.php* wyświetla w oknie przeglądarki wszystkie zamówienia znajdujące się w pliku *zamowienia.txt*



Należy teraz przyjrzeć się dokładnie funkcjom wykorzystanym w powyższym skrypcie.

Otwieranie pliku w celu odczytu — `fopen()`

Do otwarcia pliku ponownie została wykorzystana funkcja `fopen()`. W tym przypadku plik został otwarty jedynie do odczytu, tak więc zastosowano tryb `'rb'`:

```
$wp = fopen("$DOCUMENT_ROOT/./zamowienia/zamowienia.txt", 'rb');
```

Wiedzieć, kiedy przestać — `feof()`

W powyższym przykładzie pętla `while` została zastosowana w celu odczytu danych z pliku aż do jego końca. Pętla `while` sprawdza koniec pliku przy użyciu funkcji `feof()`:

```
while (!feof($wp))
```

Funkcja `feof()` używa wskaźnika pliku jako swojego jedynego parametru. Zwraca ona wartość `true`, jeżeli wskaźnik pliku znajduje się na jego końcu. Chociaż nazwa funkcji może wydawać się dziwna, łatwo ją zapamiętać, wiedząc, że `feof` znaczy w skrócie *File End Of File* (plik koniec pliku).

W tym przypadku (i zwyczajowo przy odczytywaniu pliku) plik jest odczytywany aż do osiągnięcia EOF.

Odczytywanie pliku wiersz po wierszu — `fgets()`, `fgetss()` i `fgetcsvg()`

W powyższym przykładzie do odczytania danych z pliku użyta została funkcja `fgets()`:

```
$zamowienie = fgets($wp, 999);
```

Funkcja ta jest stosowana do odczytywania pliku wiersz po wierszu. W powyższym przypadku będzie odczytywała dane, dopóki nie trafi na znak nowego wiersza (`\n`), na EOF lub przeczyta 998 bajtów pliku. Maksymalna długość odczytu jest równa wpisanej liczbie minus jeden bajt.

Istnieje wiele różnych funkcji stosowanych do odczytywania danych z pliku. Na przykład funkcja `fgets()` jest przydatna przy obróbce plików zawierających zwykły tekst, który odczytujemy fragmentami.

Interesującą odmianą `fgets()` jest funkcja `fgetss()`, która posiada następujący prototyp:

```
string fgetss(int wskaźnik_pliku, int dlugosc, string [dozwolone_znaczniki]);
```

Funkcja ta podobna jest do `fgets()`, z wyjątkiem tego, że usuwa z czytanego ciągu wszystkie znaczniki PHP i HTML, poza wyszczególnionymi w parametrze `dozwolone_znaczniki`. Stosuje się ją w celach bezpieczeństwa przy odczytywaniu plików napisanych przez innych programistów lub zawierających informacje wprowadzone przez użytkowników. Niekontrolowany obcy kod HTML może zniszczyć dokładnie zaplanowany proces formatowania strony. Niekontrolowany obcy kod PHP może oddać całą władzę nad serwerem złośliwemu użytkownikowi.

Inną odmianą funkcji `fgets()` jest funkcja `fgetcsvg()`. Posiada ona następujący prototyp:

```
array fgetcsvg(resource wskaźnik_pliku, int dlugosc, string [znak_podziału [, string załącznik]]);
```

Funkcja ta służy do rozdzielania wierszy pliku w celu zrekonstruowania zmiennych, kiedy wcześniej zastosowany został znak podziału (taki jak zaproponowany powyżej) lub przecinek (używany

w większości arkuszy kalkulacyjnych i innych aplikacjach). Oznacza to, że przy jej stosowaniu plik jest odczytywany nie wiersz po wierszu, lecz od znaku podziału do znaku podziału. Wywołanie tej funkcji następuje podobnie jak w przypadku `fgets()`, lecz przekazuje się jej również znak podziału użyty do odseparowania pól. Na przykład:

```
$zamowienie = fgetcsv($wp, 100, "\t");
```

Powyższe polecenie odczyta wiersz z pliku i podzieli ją tam, gdzie natrafi na znak tabulacji (`\t`). Wyniki zwracane są w postaci tablicy (w powyższym przykładowym kodzie: `$zamowienie`). Tablice zostaną opisane dokładniej w rozdziale 3.

Parametr *dlugosc* powinien mieć większą wartość niż długość (wyrażoną w liczbie znaków) najdłuższego wiersza odczytywanego pliku.

Parametr *załącznik* służy do wskazywania znaków, jakimi jest otoczone każde pole w wierszu. Jeśli nie zostanie on podany, przyjęta zostanie domyślnie wartość " (podwójny cudzysłów).

Odczyt całego pliku — `readfile()`, `fpassstru()`, `file()`

Plik można odczytywać nie tylko wiersz po wierszu, lecz również cały w jednym przebiegu. W tym celu należy posłużyć się jedną z trzech metod.

Pierwsza z nich polega na zastosowaniu funkcji `readfile()`. Można zastąpić niemal cały powyższy skrypt jednym wierszem kodu:

```
readfile("$DOCUMENT_ROOT/./zamowienia/zamowienia.txt");
```

Wywołanie funkcji `readfile()` otwiera plik, wyświetla zawartość w oknie przeglądarki, po czym zamyka plik. Prototyp funkcji `readfile()` jest następujący:

```
int readfile(string nazwa_pliku, int [uzycie_opcji_include_path[, resource kontekst]]);
```

Opcjonalny drugi parametr określa, czy PHP powinien szukać pliku przez opcję `include_path`, i działa w sposób identyczny jak `fopen()`. Opcjonalny parametr `kontekst` używany jest jedynie wówczas, gdy pliki są otwierane zdalnie na przykład za pośrednictwem HTTP; ten sposób użycia funkcji zostanie szerzej opisany w rozdziale 20. Funkcja zwraca całkowitą liczbę bajtów odczytanych z pliku.

Drugą funkcją tego typu jest `fpassstru()`. W celu jej zastosowania należy najpierw otworzyć plik za pomocą `fopen()`, a potem przekazać wartość wskaźnika pliku funkcji `fpassstru()`, która wyświetli zawartość tego pliku w okienku przeglądarki. Po zakończeniu działania funkcja zamyka plik.

Powyższy skrypt można zastąpić funkcją `fpassstru()` w następujący sposób:

```
$wp = fopen("DOCUMENT_ROOT/./zamowienia/zamowienia.txt", 'rb');  
fpassstru($wp);
```

Funkcja `fpassstru` zwraca wartość `true`, jeżeli odczyt powiedzie się, w przeciwnym wypadku — `false`.

Trzecią metodą odczytu całego pliku jest zastosowanie funkcji `file()`. Działa ona w identyczny sposób jak `readfile()` z jednym wyjątkiem — zamiast wyświetlić zawartość pliku w przeglądarce, zamienia ją na tablicę. Kwestia ta zostanie szczegółowo opisana w rozdziale 3. Tymczasem poniżej zostało przedstawione jej przykładowe zastosowanie:

```
$tablicapliku = file("$DOCUMENT_ROOT/./zamowienia/zamowienia.txt");
```

Polecenie to wczytuje cały plik w tablicę o nazwie `$tablicaPliku`. Każdy wiersz pliku zostanie zachowany jako osobny element tablicy. Warto zwrócić uwagę, że we wcześniejszych wersjach PHP funkcja `file()` nie była binarnie bezpieczna.

Czwartą dostępną opcją jest wykorzystanie funkcji `file_get_contents()`. Działa ona tak samo jak funkcja `readfile()`, z tą tylko różnicą, że zwraca zawartość pliku w postaci łańcucha znaków, a nie przesyła jej do przeglądarki.

Odczyt pojedynczego znaku — `fgetc()`

Inną metodą jest odczytywanie pliku znak po znaku. Można tego dokonać, stosując funkcję `fgetc()`. Jako jedyny parametr pobiera ona wskaźnik pliku i zwraca następny znajdujący się w pliku znak. Można zamienić pętlę `while` w przykładowym skrypcie na inną, używającą funkcji `fgetc()` w następujący sposób:

```
while (!feof($wp)) {
    $znak = fgetc($wp);
    if (!feof($wp))
        echo ($znak=="\n" ? "<br />": $znak);
}
```

Powyższy kod odczytuje za pomocą funkcji `fgetc()` pojedynczy znak z pliku i zapisuje go w zmiennej `$znak`, dopóki nie zostanie osiągnięty koniec pliku. Później zastosowana zostaje mała sztuczka zamieniająca znaki końca wiersza (`\n`), na złamania wierszy HTML (`
`).

Dzieje się tak jedynie w celu czystego sformatowania strony. W przypadku próby wyświetlenia pliku ze znakami nowych wierszy między rekordami cały plik zostałby wyświetlony jako jeden wiersz (warto sprawdzić). Przeglądarki nie generują znaków niewidocznych, dlatego trzeba je zastępować znakami HTML oznaczającymi nowy wiersz (`
`). W celu przeprowadzenia tej zamiany zastosowany został operator trójkowy.

Pomniejszym efektem ubocznym stosowania funkcji `fgetc()` zamiast `fgets()` jest fakt, że w przeciwieństwie do funkcji `fgets()` zwraca ona znak EOF. Dlatego po przeczytaniu znaku należy ponownie użyć `feof()`, aby znak ten nie został wyświetlony w przeglądarce.

Jeżeli nie istnieje wyraźny powód odczytywania pliku znak po znaku, stosowanie tej metody nie jest polecane.

Odczytywanie zadanej długości — `fread()`

Ostatnią metodą odczytywania pliku jest zastosowanie funkcji `fread()` w celu odczytania z pliku zadanej liczby bajtów. Funkcja ta posiada następujący prototyp:

```
string fread(resource wskaźnik_pliku, int dlugosc);
```

Funkcja `fread()` odczytuje przekazaną jej liczbę bajtów, chyba że wcześniej napotka znak końca pliku lub pakietu sieciowego.

Inne przydatne funkcje plikowe

Poza powyższymi istnieje jeszcze kilka przydatnych w niektórych zastosowaniach funkcji plikowych. Część z nich zostanie opisana poniżej.

Sprawdzanie istnienia pliku — file_exists()

W celu sprawdzenia istnienia pliku bez otwierania go stosuje się funkcję `file_exists()`:

```
if (file_exists("$DOCUMENT_ROOT../zamowienia/zamowienia.txt")) {
    echo 'Są zamówienia czekające na przyjęcie.';
} else {
    echo 'Aktualnie nie ma żadnych zamówień.';
}
```

Określanie wielkości pliku — filesize()

W celu sprawdzenia wielkości pliku stosuje się funkcję `filesize()`:

```
echo filesize("$DOCUMENT_ROOT../zamowienia/zamowienia.txt");
```

Funkcja ta zwraca wielkość pliku w bajtach i może zostać zastosowana w połączeniu z funkcją `fread()` w celu odczytania jednorazowo całego pliku (lub jakiejś jego części). Można zastąpić cały przykładowy skrypt następującymi wierszami kodu:

```
$wp = fopen("$DOCUMENT_ROOT../zamowienia/zamowienia.txt", 'rb');
echo nl2br(fread($wp, filesize("$DOCUMENT_ROOT../zamowienia/zamowienia.txt")));
fclose($wp);
```

Funkcja `nl2br()` przekształca w generowanym kodzie HTML znaki `\n` w znaki nowego wiersza (`
`).

Kasowanie pliku — unlink()

Można skasować plik zamówień po ich przyjęciu, stosując w tym celu funkcję `unlink()` (nie istnieje funkcja o nazwie `delete`). Na przykład:

```
unlink("$DOCUMENT_ROOT../zamowienia/zamowienia.txt");
```

Powyższa funkcja zwraca wartość `false`, jeżeli plik nie mógł zostać usunięty. Sytuacja taka zdarza się zazwyczaj z powodu niewystarczających praw do pliku bądź jeżeli plik nie istnieje.

Poruszanie się wewnątrz pliku — rewind(), fseek() i ftell()

Można poruszać się w obrębie pliku i poznawać pozycje wskaźnika wewnątrz tego pliku, stosując funkcje `rewind()`, `fseek()` i `ftell()`.

Funkcja `rewind()` ustawia wskaźnik pliku z powrotem na jego początek. Funkcja `ftell()` informuje, jak daleko (w bajtach) został przesunięty wskaźnik. Na końcu powyższego skryptu (przed poleceniem `fclose()`) można na przykład dodać następujące wiersze :

```
echo 'Końcowa pozycja wskaźnika pliku wynosi ' .(ftell($wp));
echo '<br />';
rewind($wp);
echo 'Po przewinięciu, pozycja wynosi ' .(ftell($wp));
echo '<br />';
```

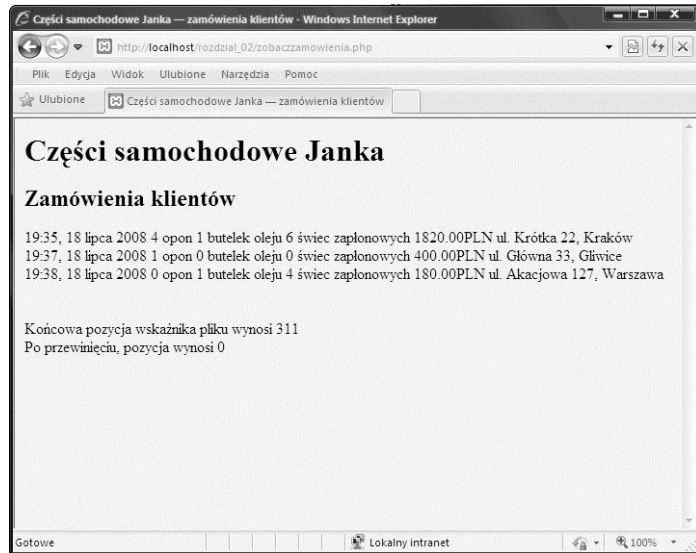
Wynik wyświetlony w przeglądarce powinien być podobny do przedstawionego na rysunku 2.5.

Funkcja `fseek()` stosowana jest do ustawiania wskaźnika pliku w dowolnym punkcie pliku. Jej prototyp wygląda następująco:

```
int fseek(resource wskaznik_pliku, int offset [, int skąd]);
```

Rysunek 2.5.

Kiedy zamówienia są przeczytane, wskaźnik pliku wskazuje na jego koniec, offset 311 bajtów. Wywołanie przewinięcia ustawia go na pozycji 0 (na początku pliku)



Wywołanie funkcji `fseek()` ustawia wskaźnik pliku `wp` w punkcie wskazywanym przez `skąd` i przesuwa o `offset` bajtów, licząc od początku pliku. Opcjonalny parametr `skąd` posiada domyślną wartość `SEEK_SET`, czyli początek pliku. Parametr ten może również mieć wartości `SEEK_CUR` (bieżąca pozycja wskaźnika pliku) oraz `SEEK_END` (koniec pliku).

Funkcja `rewind()` jest równoznaczna z wywołaniem funkcji `fseek()` z zerowym offsetem. Na przykład można zastosować funkcję `fseek()` w celu znalezienia środkowego rekordu w pliku danych lub po to, aby przeprowadzić przeszukiwanie binarne. Zazwyczaj po osiągnięciu poziomu złożoności wymagającego stosowania takich mechanizmów polecane jest zastosowanie bazy danych.

Blokowanie pliku

Można wyobrazić sobie sytuację, w której dwóch klientów stara się zamówić produkt w tym samym czasie (dzieje się tak często, zwłaszcza gdy strona jest licznie odwiedzana). Co się stanie, gdy jeden z klientów wywoła funkcję `fopen()` i zacznie zapis w pliku, a drugi uczyni to samo? Jak będzie wyglądać ostateczna zawartość pliku? Czy najpierw zostanie zapisane pierwsze zamówienie, a później drugie, czy też odwrotnie? A może dojdzie do sytuacji niepożądaney, na przykład oba zamówienia wymieszają się ze sobą? Odpowiedź na powyższe pytania zależy od konkretnego systemu operacyjnego, ale zazwyczaj jest to wielka niewiadoma.

W celu uniknięcia powyższych problemów stosuje się blokowanie plików. W PHP wykorzystuje się w tym celu funkcję `flock()`. Powinna ona zostać wywołana po otwarciu pliku, lecz przed odczytaniem go lub zapisaniem w nim danych.

Funkcja `flock()` posiada następujący prototyp:

```
bool flock(resource wskaźnik_pliku, int dzialanie [, int zablokuj])
```

Funkcji `flock()` należy przekazać wskaźnik otwartego pliku i cyfrę określającą wymagany rodzaj zamka. Funkcja ta zwraca `true`, jeżeli zamek został prawidłowo założony, a `false` w przeciwnym wypadku. Opcjonalny trzeci parametr będzie zawierał wartość `true`, jeśli założenie zamka spowoduje zablokowanie bieżącego procesu (czyli zmuszenie go do przejścia w stan oczekiwania).

Możliwe wartości parametru *działanie* są przedstawione w tabeli 2.2.

Tabela 2.2. Wartości parametru *działanie* funkcji *flock()*

Wartość parametru <i>działanie</i>	Znaczenie
LOCK_SH (dawniej 1)	Blokowanie odczytu. Pozwala na dzielenie pliku z innymi czytającymi
LOCK_EX (dawniej 2)	Blokowanie zapisu. Wyłącza plik z użytku; nie może on być dzielony
LOCK_UN (dawniej 3)	Zwolnienie istniejącej blokady
LOCK_NB (dawniej 4)	Dodanie wartości 4 do parametru <i>działanie</i> przeciwdziała zablokowaniu próby założenia blokady

Stosując funkcję `flock()`, należy dodać ją do wszystkich skryptów korzystających z pliku. W innym przypadku jest ona bezwartościowa.

Należy zwrócić uwagę, iż `flock()` nie działa w systemie NFS i innych sieciowych systemach plików. Nie działa ona również w starszych systemach plików, które nie obsługują blokowania plików — systemem takim jest na przykład FAT. W niektórych systemach operacyjnych funkcja ta jest zaimplementowana na poziomie procesu i nie będzie działać prawidłowo, jeśli stosowany będzie wielowątkowy API serwera.

Aby zastosować ją w powyższym przykładzie, należy zmienić skrypt *przetworz zamowienie.php* w następujący sposób:

```
$wp = fopen("$DOCUMENT_ROOT/./zamowienia/zamowienia.txt", 'ab');
flock($wp, LOCK_EX); // blokada zapisu pliku
fwrite($wp, $ciagwyjsciowy);
flock($wp, LOCK_UN); // zwolnienie blokady zapisu
fclose($wp);
```

Do skryptu *zobacz zamowienia.php* należy również dodać blokady:

```
$wp = fopen("$DOCUMENT_ROOT/./zamowienia/zamowienia.txt", 'r');
flock($wp, LOCK_SH); // blokada odczytu pliku
// odczyt z pliku
flock($wp, LOCK_UN); // zwolnienie blokady odczytu
fclose($wp);
```

Kod jest teraz dużo solidniejszy, ale ciągle niedoskonały. Co by się stało, gdyby dwa skrypty jednocześnie usiłowały założyć blokadę? Spowodowałoby to „wyścig” o bardzo niepewnym wyniku, co wywołałoby wiele kolejnych problemów. Lepszą metodą jest zastosowanie *DBMS* (ang. *Database Management System* — system zarządzania bazami danych).

Lepszy sposób obróbki danych — systemy zarządzania bazami danych

Wszystkie powyższe przykłady używały plików jednorodnych. W drugiej części tej książki opiszemy alternatywę tej metody — MySQL, system zarządzania relacyjnymi bazami danych (RDBMS). Można by zapytać, w jakim celu?

Problemy związane ze stosowaniem plików jednorodnych

Istnieje kilka problemów związanych z pracą z plikami jednorodnymi:

- Praca z dużym plikiem może być bardzo powolna.
- Poszukiwanie konkretnego rekordu lub grupy rekordów w pliku jednorodnym jest trudne. Jeżeli rekordy są uporządkowane, można zastosować pewien sposób przeszukiwania binarnego w połączeniu z rekordami o ustalonej szerokości i przeszukiwaniem według pola kluczowego. Aby znaleźć pewne wzory informacji (na przykład wyszukując wszystkich klientów zamieszkałych w Gliwicach), należy sprawdzać indywidualnie każdy rekord.
- Problemy sprawiają dostęp jednoczesny. Powyżej przedstawione zostały sposoby blokowania plików, ale, jak opisano powyżej, może to spowodować wyścig lub „wąskie gardło”. W razie dużego ruchu na stronie liczna grupa użytkowników może czekać na odblokowanie pliku, aby złożyć zamówienie. Jeżeli potrwa to zbyt długo, przeniosą się do konkurencji.
- Wszystkie przedstawione powyżej procesy przetwarzania plików działają sekwencyjnie — rozpoczynają od początku pliku i czytają go do końca. Aby umieścić lub skasować rekordy znajdujące się w środku pliku, należy umieścić cały plik w pamięci, dokonać zmian, a na końcu zapisać go w całości. Podczas pracy z dużymi plikami konieczność wykonywania wszystkich tych kroków może sprawiać problemy.
- Poza ograniczonymi możliwościami, oferowanymi przez pozwolenia dostępu do plików, nie istnieje żadna prosta metoda tworzenia różnych poziomów dostępu do danych.

Jak RDBMS rozwiązują powyższe problemy?

Relacyjne systemy zarządzania bazami danych umożliwiają rozwiązania wszystkich powyższych kwestii.

- RDBMS pozwalają na znacznie szybszy dostęp do plików niż pliki jednorodne. MySQL, system bazodanowy prezentowany w tej książce, należy do najszybszych RDBMS.
- RDBMS można zadawać zapytania o dane spełniające konkretne kryteria.
- RDBMS posiadają wbudowany mechanizm zapewniania równoległego dostępu, który pozostaje poza kręgiem pracy programisty.
- RDBMS pozwalają na swobodny dostęp do danych.
- RDBMS posiadają wbudowany system przywilejów. MySQL jest w tej dziedzinie szczególnie rozbudowany.

Prawdopodobnie głównym powodem używania RDBMS jest fakt, że funkcjonalność, którą powinny posiadać systemy przechowywania danych, została już w nich zaimplementowana (a przynajmniej jej większość). Oczywiście można napisać własną bibliotekę funkcji PHP, lecz po co ponownie wymyślać koło?

W części II tej książki opiszemy ogólną zasadę działania relacyjnych baz danych, a w szczególności konfigurację i zastosowanie MySQL w tworzeniu stron WWW opartych na bazach danych.

Jeżeli tworzony jest prosty system, który nie wymaga pełnowymiarowej bazy danych, natomiast chcemy uniknąć zakładania zamków i innych komplikacji związanych z używaniem plików płaskich, można użyć nowego rozszerzenia PHP o nazwie SQLite. Udostępnia ono SQL-owy interfejs do pliku płaskiego. W tej książce skupimy się na używaniu serwera MySQL, natomiast więcej informacji na temat SQLite można znaleźć pod adresami <http://sqlite.org/> oraz <http://www.php.net/sqlite>.

Propozycje dalszych lektur

Więcej informacji na temat interakcji z systemem plików znajduje się w rozdziale 19. Opiszemy w nim metody zmiany uprawnień dostępu, własności i nazw plików, a także pracę z katalogami oraz interakcję ze środowiskiem systemu plików.

Zalecamy również lekturę rozdziału na temat systemów plików w podręczniku elektronicznym PHP, dostępnym pod adresem <http://www.php.net/filesystem>.

W następnym rozdziale

W kolejnym rozdziale przedstawimy tablice — czym są i jak mogą zostać zastosowane w skryptach PHP do przetwarzania danych.