helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

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

PHP i MySQL. Tworzenie stron WWW



Autor: William Jason Gilmore
Tłumaczenie: Jacek Smycz, Daniel Kaczmarek
ISBN: 83-7197-523-6
Tytuł oryginału: PHP and MySQL Web Developmnent
Format: B5, stron: około 800
Przykłady na ftp: 713 kB

Ta praktyczna książka, ukazująca przede wszystkim zastosowania komercyjne, zawiera liczne przykłady. Ilustrują one realizację następujących zadań: uwierzytelnianie użytkowników, konstruowanie koszyka na zakupy, dynamiczne generowanie dokumentów PDF i obrazków, wysyłanie i zarządzanie pocztą elektroniczną, ułatwianie dyskusji między użytkownikami oraz zarządzanie zawartością. Autorzy położyli istotny nacisk na kwestię bezpieczeństwa.

Oprócz składni i biblioteki funkcji PHP, niniejsza książka opisuje również podstawowe pojęcia z dziedziny profesjonalnej inżynierii oprogramowania związanego z siecią WWW. Niektóre, takie jak utrzymywanie, współpraca i testowanie, są kwestiami istotnymi dla inżynierów oprogramowania pracujących we wszystkich domenach zastosowań. Inne pojęcia, takie jak uwierzytelnianie, szyfrowanie i kontrola sesji, mają szczególne znaczenie dla projektów programistycznych opartych na Internecie.

- Dynamiczne tworzenie kodu HTML, obrazków i dokumentów
- Tworzenie bezpiecznych usług za pomocą uwierzytelniania i SSL
- Tworzenie koszyka na zakupy dla witryn handlu elektronicznego
- Opis praktyk związanych z inżynierią oprogramowania dla większych projektów WWW
- Zastosowanie obiektowych technik programistycznych w Internecie

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

Spis treści

O Autorach 21

Wprowadzenie 23

Dlaczego warto przeczytać niniejszą książkę? 23
Korzyści wynikające z lektury tej książki 24
Czym jest PHP? 25
Nowości w PHP, wersja 4 25
Czym jest MySQL? 26
Dlaczego warto wykorzystywać PHP i MySQL? 26
Niektóre zalety PHP 27
Wydajność 27
Integracja z bazami danych 27
Wbudowane biblioteki 27
Koszt 28
Nauka PHP 28
Przenośność 28
Kod źródłowy 28
Niektóre zalety MySQL 28
Wydajność 29
Niski koszt 29
Łatwość wykorzystania 29
Przenośność 29
Kod źródłowy 29
Układ treści książki 29
Kody źródłowe 30
Uwagi końcowe 30

Część I Stosowanie PHP 31

Rozdział 1. Podstawowy kurs PHP 33

Zastosowanie PHP 34
Przykładowa aplikacja: „Części samochodowe Janka” 34
Formularz zamówienia 35
Przetwarzanie formularza 36
Osadzanie PHP w HTML 36
Zastosowanie znaczników PHP 38
Style znaczników PHP 38
Instrukcje PHP 39
Odstępy 39
Komentarze 40

Dodawanie zawartości dynamicznej ... 41
Wywoływanie funkcji ... 41
Funkcja date() ... 42
Dostęp do zmiennych formularza ... 42
Zmienne formularza ... 42
Łączenie ciągów ... 44
Zmienne i ciągi znaków ... 44
Identyfikatory ... 45
Zmienne zadeklarowane przez użytkownika ... 45
Przypisywanie wartości zmiennym ... 45
Typy zmiennych ... 46
Typy danych w PHP ... 46
Siła typu ... 46
Rzutowanie typu ... 47
Zmienne zmiennych ... 47
Stałe ... 48
Zasięg zmiennych ... 48
Operatory ... 49
Operatory arytmetyczne ... 49
Operatory ciągów ... 50
Operatory przypisania ... 50
Operatory porównań ... 53
Operatory logiczne ... 54
Operatory bitowe ... 54
Pozostałe operatory ... 54
Stosowanie operatorów: obliczanie sum w formularzu ... 56
Pierwszeństwo i kolejność: wyznaczanie wartości wyrażeń ... 57
Zarządzanie zmiennymi ... 59
Sprawdzanie i ustawianie typów zmiennych ... 59
Sprawdzanie stanu zmiennej ... 60
Reinterpretacja zmiennych ... 61
Struktury kontrolujące ... 61
Podejmowanie decyzji za pomocą instrukcji warunkowych ... 61
Instrukcja if ... 61
Bloki kodu ... 62
Uwaga poboczna: wcinanie kodu ... 62
Instrukcja else ... 62
Instrukcja elseif ... 63
Instrukcja switch ... 64
Porównanie różnych instrukcji warunkowych ... 66
Iteracja: powtarzanie działań ... 66
Pętle while ... 67
Pętle for ... 68
Pętle do..while ... 69
Wyłamywanie się ze struktury skryptu ... 70
W następnym rozdziale: zapisywanie zamówienia klienta ... 70

Rozdział 2. Przechowywanie i wyszukiwanie danych ... 71

Zapisywanie danych do późniejszego użycia ... 72
Przechowywanie i wyszukiwanie zamówień Janka ... 72
Podstawowe informacje na temat przetwarzania plików ... 73

Otwieranie pliku ... 74
Tryby otwarcia pliku ... 74
Stosowanie funkcji fopen() do otwarcia pliku ... 74
Otwieranie pliku przez protokół FTP lub HTTP ... 76
Problemy z otwieraniem plików ... 76
Zapisywanie danych w pliku ... 78
Parametry funkcji fwrite() ... 79
Formaty plików ... 79
Zamykanie pliku ... 80
Odczyt z pliku ... 80
Otwieranie pliku w celu odczytu — fopen() ... 81
Wiedzieć, kiedy przestać — feof() ... 81
Odczytywanie pliku linia po linii — fgets(), fgetss() i fgetcsv() ... 82
Odczyt całego pliku — readfile(), fpassthru(), file() ... 83
Odczyt pojedynczego znaku — fgetc() ... 84
Odczytywanie zadanej długości — fread() ... 84
Inne przydatne funkcje plikowe ... 84
Sprawdzanie istnienia pliku — file_exists() ... 85
Określanie wielkości pliku — filesize() ... 85
Kasowanie pliku — unlink() ... 85
Poruszanie się wewnątrz pliku — rewind(), fseek() i ftell() ... 85
Blokowanie pliku ... 86
Lepszy sposób obróbki danych — systemy zarządzania bazami danych ... 87
Problemy związane ze stosowaniem plików jednorodnych ... 88
Jak RDBMS rozwiązują powyższe problemy? ... 88
Propozycje dalszych lektur ... 89
W następnym rozdziale ... 89

Rozdział 3. Stosowanie tablic ... 91

Czym są tablice? ... 92
Tablice indeksowane numerycznie ... 92
Inicjowanie tablic indeksowanych numerycznie ... 92
Dostęp do zawartości tablicy ... 93
Dostęp do tablic przy zastosowaniu pętli ... 94
Tablice asocjacyjne ... 94
Inicjowanie tablicy asocjacyjnej ... 94
Dostęp do elementów tablicy ... 95
Stosowanie pętli z funkcjami each() i list() ... 95
Tablice wielowymiarowe ... 97
Sortowanie tablic ... 100
Stosowanie funkcji sort() ... 100
Stosowanie funkcji asort() i ksort() do porządkowania tablic asocjacyjnych ... 101
Sortowanie odwrotne ... 101
Sortowanie tablic wielowymiarowych ... 102
Typy sortowań definiowane przez użytkownika ... 102
Odwrotne sortowanie zdefiniowane przez użytkownika ... 104
Zmiany kolejności elementów w tablicach ... 104
Stosowanie funkcji shuffle() ... 104
Stosowanie funkcji array_reverse() ... 105
Ładowanie tablic z plików ... 106

Inne działania na tablicach ... 109
Poruszanie się wewnątrz tablicy — funkcje each(), current(), reset(), end(), next(), pos() i prev() ... 109
Dołączanie dowolnej funkcji do każdego elementu tablicy — funkcja array_walk() ... 110
Liczenie elementów tablicy: count(), sizeof() i array_count_values() ... 111
Konwersja tablic na zmienne skalarne — funkcja extract() ... 112
Propozycje dalszych lektur ... 113
W następnym rozdziale ... 113

Rozdział 4. Manipulowanie ciągami i wyrażenia regularne ... 115
Przykładowa aplikacja — Inteligentny Formularz Pocztowy ... 115
Formatowanie ciągów ... 117
Przycinanie ciągów — funkcje chop(), ltrim() i trim() ... 118
Formatowanie ciągów w celu ich prezentacji ... 118
Formatowanie ciągów do przechowania — funkcje AddSlashes() i StripSlashes() ... 121
Łączenie i rozdzielanie ciągów za pomocą funkcji ciągów ... 123
Stosowanie funkcji explode(), implode() i join() ... 123
Stosowanie funkcji strtok() ... 124
Stosowanie funkcji substr() ... 125
Porównywanie ciągów ... 125
Porządkowanie ciągów — funkcje strcmp(), strcasecmp() i strnatcmp() ... 126
Sprawdzanie długości ciągu za pomocą funkcji strlen() ... 126
Dopasowywanie i zamiana podciągów za pomocą funkcji ciągów ... 127
Znajdowanie ciągów w ciągach — funkcje strstr(), strchr(), strrchr() i stristr() ... 127
Odnajdywanie pozycji podciągu — funkcje strpos() i strrpos() ... 128
Zamiana podciągów — funkcje str_replace() i substr_replace() ... 129
Wprowadzenie do wyrażeń regularnych ... 130
Podstawy ... 130
Zbiory i klasy znaków ... 131
Powtarzalność ... 132
Podwyrażenia ... 132
Podwyrażenia policzalne ... 133
Kotwiczenie na początku lub na końcu ciągu ... 133
Rozgałęzianie ... 133
Dopasowywanie specjalnych znaków literowych ... 134
Podsumowanie znaków specjalnych ... 134
Umieszczanie wszystkiego razem (Inteligentny Formularz) ... 134
Odnajdywanie podciągów za pomocą wyrażeń regularnych ... 135
Zamiana podciągów za pomocą wyrażeń regularnych ... 136
Rozdzielanie ciągów przy pomocy wyrażeń regularnych ... 136
Porównanie funkcji ciągów i funkcji wyrażeń regularnych ... 137
Propozycje dalszych lektur ... 137
W następnym rozdziale ... 137

Rozdział 5. Ponowne wykorzystanie kodu i tworzenie funkcji ... 139
Dlaczego ponownie stosować kod? ... 140
Koszt ... 140
Niezawodność ... 140
Spójność ... 140
Stosowanie funkcji require() i include() ... 141
Stosowanie funkcji require() ... 141
Rozszerzenia plików i require() ... 142
Znaczniki PHP i require() ... 142

Stosowanie require() w szablonach stron WWW ... 143
Stosowanie opcji auto_prepend_file i auto_append_file ... 147
Stosowanie funkcji include() ... 148
Stosowanie funkcji w PHP ... 149
Wywoływanie funkcji ... 150
Wywołanie niezdefiniowanej funkcji ... 151
Wielkość liter a nazwy funkcji ... 152
Dlaczego powinno się definiować własne funkcje? ... 152
Podstawowa struktura funkcji ... 153
Nadawanie nazwy funkcji ... 154
Parametry ... 154
Zasięg ... 156
Przekazanie przez referencję czy przekazanie przez wartość ... 158
Powrót z funkcji ... 160
Zwracanie wartości przez funkcje ... 161
Bloki kodu ... 162
Rekurencja ... 163
Propozycje dalszych lektur ... 165
W następnym rozdziale ... 165

Rozdział 6. Obiektowy PHP ... 167

Koncepcje programowania obiektowego ... 167
Klasy i obiekty ... 167
Polimorfizm ... 169
Dziedziczenie ... 169
Tworzenie klas, atrybutów i operacji w PHP ... 170
Struktura klasy ... 170
Konstruktory ... 170
Tworzenie egzemplarzy ... 171
Stosowanie atrybutów klasy ... 172
Wywoływanie operacji klas ... 173
Implementacja dziedziczenia w PHP ... 174
Unieważnianie ... 175
Wielodziedziczenie ... 176
Tworzenie klas ... 177
Tworzenie kodu dla własnej klasy ... 178
W następnej części ... 186

Część II Stosowanie MySQL ... 187

Rozdział 7. Projektowanie internetowej bazy danych ... 189

Koncepcje relacyjnych baz danych ... 190
Tabele ... 190
Kolumny ... 190
Wiersze ... 191
Wartości ... 191
Klucze ... 191
Schematy ... 192
Relacje ... 192

Jak zaprojektować internetową bazę danych ... 193
Określ obiekty świata realnego, których model chcesz wykonać ... 193
Unikaj przechowywania redundantnych danych ... 194
Zapisuj atomowe wartości kolumn ... 195
Dobierz właściwe klucze ... 196
Pomyśl o zapytaniach, które zadasz bazie ... 197
Unikaj tworzenia tabel z wieloma pustymi polami ... 197
Typy tabel — podsumowanie ... 197
Architektura internetowej bazy danych ... 198
Architektura ... 198
Propozycje dalszych lektur ... 199
W następnym rozdziale ... 199

Rozdział 8. Tworzenie internetowej bazy danych ... 201

Uwagi na temat użytkowania monitora MySQL ... 202
Jak zalogować się do serwera MySQL ... 203
Tworzenie baz i rejestrowanie użytkowników ... 205
Tworzenie bazy danych ... 205
Użytkownicy i przywileje ... 205
Wprowadzenie do systemu przywilejów MySQL ... 206
Zasada najmniejszego przywileju ... 206
Rejestrowanie użytkowników: polecenie GRANT ... 206
Typy i poziomy przywilejów ... 208
Polecenie REVOKE ... 210
Przykłady użycia poleceń GRANT i REVOKE ... 210
Rejestrowanie użytkownika łączącego się z Internetu ... 211
Wylogowanie się użytkownika root ... 212
Używanie odpowiedniej bazy danych ... 212
Tworzenie tabel bazy danych ... 212
Znaczenie dodatkowych atrybutów kolumn ... 214
Typy kolumn ... 215
Rzut oka na bazę danych — polecenia SHOW i DESCRIBE ... 217
Identyfikatory MySQL ... 218
Typy danych w kolumnach ... 219
Typy liczbowe ... 219
Propozycje dalszych lektur ... 223
W następnym rozdziale ... 223

Rozdział 9. Praca z bazą danych MySQL ... 225

Czym jest SQL? ... 225
Zapisywanie danych do bazy ... 226
Wyszukiwanie danych w bazie ... 228
Wyszukiwanie danych spełniających określone kryteria ... 229
Wyszukiwanie danych w wielu tabelach ... 230
Szeregowanie danych w określonym porządku ... 236
Grupowanie i agregowanie danych ... 237
Wskazanie wierszy, które mają być wyświetlone ... 239
Dokonywanie zmian rekordów w bazie danych ... 240
Zmiana struktury istniejących tabel ... 241
Usuwanie rekordów z bazy danych ... 242

Usuwanie tabel ... 242
Usuwanie całych baz danych ... 243
Propozycje dalszych lektur ... 243
W następnym rozdziale ... 243

Rozdział 10. Łączenie się z bazą MySQL za pomocą PHP ... 245

Jak działa internetowa baza danych ... 246
Etapy wysyłania zapytań do bazy danych z poziomu strony WWW ... 248
Sprawdzenie poprawności wpisanych danych ... 249
Ustanawianie połączenia z bazą danych ... 250
Wybór właściwej bazy danych ... 252
Wysyłanie zapytań do bazy danych ... 252
Odczytywanie rezultatów zapytań ... 253
Zamykanie połączenia z bazą danych ... 255
Wstawianie nowych danych do bazy ... 255
Inne użyteczne funkcje PHP i MySQL ... 258
Zwalnianie zasobów ... 258
Tworzenie i usuwanie baz danych ... 259
Inne interfejsy bazodanowe PHP ... 259
Propozycje dalszych lektur ... 259
W następnym rozdziale ... 260

Rozdział 11. MySQL dla zaawansowanych ... 261

Szczegóły systemu przywilejów ... 261
Tabela user ... 262
Tabele db i host ... 263
Tabele tables_priv i columns_priv ... 264
Kontrola dostępu: w jaki sposób MySQL używa tabel przywilejów ... 265
Zmiana przywilejów: kiedy zmiany zostaną uwzględnione? ... 266
Ochrona bazy danych ... 266
MySQL z perspektywy systemu operacyjnego ... 267
Hasła ... 267
Przywileje użytkowników ... 268
MySQL i Internet ... 269
Uzyskiwanie szczegółowych informacji o bazie danych ... 269
Uzyskiwanie informacji poleceniem SHOW ... 270
Uzyskiwanie informacji o kolumnach za pomocą polecenia DESCRIBE ... 271
Jak wykonywane są zapytania: polecenie EXPLAIN ... 272
Przyspieszanie wykonania zapytań za pomocą indeksów ... 275
Wskazówki dotyczące optymalizacji ... 276
Optymalizacja projektu bazy danych ... 276
Przywileje ... 276
Optymalizacja tabel ... 276
Stosowanie indeksów ... 277
Używanie wartości domyślnych ... 277
Używanie stałych połączeń z bazą ... 277
Więcej wskazówek ... 277
Różne typy tabel ... 277
Ładowanie danych z pliku ... 278
Propozycje dalszych lektur ... 278
W następnej części ... 279

Część III Handel elektroniczny i bezpieczeństwo 281

Rozdział 12. Komercyjne witryny internetowe.....283

Co chcemy osiągnąć?.....283
Rodzaje komercyjnych stron WWW 284
Broszury internetowe.....284
Przyjmowanie zamówień na produkty i usługi.....287
Dostarczanie usług lub wyrobów mających postać cyfrową.....291
Zwiększanie wartości produktów i usług.....292
Ograniczanie kosztów.....293
Ryzyko i zagrożenia.....293
Crackerzy.....294
Przyciągnięcie niewystarczającej liczby klientów.....294
Awarie sprzętu komputerowego.....295
Awarie sieci elektrycznych, komunikacyjnych i komputerowych oraz systemu wysyłkowego.....295
Silna konkurencja.....295
Błędy w oprogramowaniu.....296
Zmiany polityki rządowej.....296
Ograniczenie pojemności systemów.....296
Wybór strategii.....297
W następnym rozdziale.....297

Rozdział 13. Bezpieczeństwo komercyjnych stron WWW 299

Jaką wagę mają posiadane przez nas informacje? 300
Zagrożenia bezpieczeństwa.....300
Ujawnienie informacji poufnych 301
Utrata lub zniszczenie danych 303
Modyfikacje danych 304
Blokada usługi 305
Błędy w oprogramowaniu.....306
Zaprzeczenie korzystania z usługi 307
Równoważenie użyteczności, wydajności, kosztów i bezpieczeństwa.....308
Opracowanie polityki bezpieczeństwa.....309
Zasady uwierzytelniania.....309
Wykorzystanie mechanizmu uwierzytelniania 311
Podstawy szyfrowania.....311
Szyfrowanie z kluczem prywatnym 313
Szyfrowanie z kluczem publicznym 313
Podpis cyfrowy.....314
Certyfikaty cyfrowe 315
Bezpieczne serwery WWW 317
Monitorowanie i zapisywanie zdarzeń.....318
Zapory sieciowe 319
Tworzenie kopii zapasowych.....319
Tworzenie kopii zapasowych zwykłych plików.....320
Tworzenie kopii zapasowych i odzyskiwanie baz danych MySQL 320
Bezpieczeństwo fizyczne 321
W następnym rozdziale.....322

Rozdział 14. Uwierzytelnianie przy użyciu PHP i MySQL 323
Identyfikacja użytkowników 323
Implementacja kontroli dostępu 324
Przechowywanie haseł dostępu 327
Szyfrowanie haseł 329
Zastrzeganie więcej niż jednej strony 331
Podstawowa metoda uwierzytelniania 332
Wykorzystanie podstawowej metody uwierzytelniania w PHP 333
Wykorzystanie podstawowej metody uwierzytelniania na serwerze Apache przy użyciu plików .htaccess 335
Wykorzystanie podstawowej metody uwierzytelniania na serwerze IIS 339
Wykorzystanie modułu mod_auth_mysql do celów uwierzytelniania 341
Instalacja modułu mod_auth_mysql 342
Zadziałało? 342
Praca z mod_auth_mysql 343
Implementacja własnej metody uwierzytelniania 344
Propozycje dalszych lektur 344
W następnym rozdziale 344

Rozdział 15. Zabezpieczanie transakcji przy użyciu PHP i MySQL 345
Zapewnienie bezpieczeństwa transakcji 345
Komputer użytkownika 346
Internet 348
System docelowy 349
Wykorzystanie protokołu Secure Sockets Layer (SSL) 350
Kontrola danych pochodzących od użytkownika 353
Bezpieczne przechowywanie danych 354
Cel przechowywania numerów kart kredytowych 356
Szyfrowanie danych w PHP 356
Propozycje dalszych lektur 365
W następnej części 365

Część IV Zaawansowane techniki PHP 367

Rozdział 16. Interakcja z systemem plików i serwerem 369
Wprowadzenie do wysyłania plików 369
Kod HTML służący do wysyłania plików 370
Tworzenie obsługującego plik PHP 371
Popularne problemy 374
Stosowanie funkcji katalogowych 375
Odczyt z katalogów 375
Otrzymywanie informacji na temat aktualnego katalogu 377
Tworzenie i usuwanie katalogów 377
Interakcja z systemem plików 378
Otrzymywanie informacji o pliku 378
Zmiana właściwości pliku 380
Tworzenie, usuwanie i przenoszenie plików 381
Stosowanie funkcji uruchamiających programy 382
Interakcja ze środowiskiem: funkcje getenv() i putenv() 384
Propozycje dalszych lektur 384
W następnym rozdziale 384

Rozdział 17. Stosowanie funkcji sieci i protokołu 385
Przegląd protokołów 385
Wysyłanie i odczytywanie poczty elektronicznej 386
Korzystanie z innych usług WWW 387
Stosowanie funkcji połączeń sieciowych 389
Korzystanie z FTP 393
Stosowanie FTP w celu utworzenia kopii bezpieczeństwa lub kopii lustrzanej pliku 393
Wysyłanie plików 399
Unikanie przekroczenia dopuszczalnego czasu 400
Stosowanie innych funkcji FTP 400
Stosowanie ogólnej komunikacji sieciowej za pomocą cURL 401
Propozycje dalszych lektur 403
W następnym rozdziale 404

Rozdział 18. Zarządzanie datą i czasem 405
Uzyskiwanie informacji o dacie i czasie w PHP 405
Stosowanie funkcji date() 405
Obsługa znaczników czasu Uniksa 407
Stosowanie funkcji getdate() 408
Sprawdzanie poprawności dat 408
Konwersja pomiędzy formatami daty PHP i MySQL 409
Obliczanie dat 410
Stosowanie funkcji kalendarzowych 411
Propozycje dalszych lektur 412
W następnym rozdziale 412

Rozdział 19. Generowanie obrazków 413
Konfigurowanie obsługi obrazków w PHP 413
Formaty obrazków 414
JPEG 414
PNG 414
WBMP 415
GIF 415
Tworzenie obrazków 416
Tworzenie kadru obrazka 417
Rysowanie lub umieszczanie tekstu w obrazku 417
Wyświetlanie ostatecznej grafiki 419
Końcowe czynności porządkujące 420
Stosowanie automatycznie generowanych obrazków na innych stronach 421
Stosowanie tekstu i czcionek do tworzenia obrazków 422
Konfiguracja podstawowego kadru 425
Dopasowanie tekstu do przycisku 425
Nadawanie tekstowi odpowiedniej pozycji 428
Wpisywanie tekstu do przycisku 428
Etap końcowy 429
Rysowanie figur i wykresów danych 429
Inne funkcje obrazków 436
Propozycje dalszych lektur 437
W następnym rozdziale 437

Rozdział 20. Stosowanie kontroli sesji w PHP 439
Czym jest kontrola sesji? 439
Podstawowa zasada działania sesji 440
Czym jest cookie? 440
Konfiguracja cookies w PHP 441
Stosowanie cookies w sesji 441
Przechowywanie identyfikatora sesji 442
Implementacja prostych sesji 442
Rozpoczynanie sesji 442
Zgłaszanie zmiennych sesji 443
Stosowanie zmiennych sesji 443
Usuwanie zmiennych i niszczenie sesji 444
Przykład prostej sesji 444
Konfiguracja kontroli sesji 446
Implementacja uwierzytelniania w kontroli sesji 447
Propozycje dalszych lektur 453
W następnym rozdziale 453

Rozdział 21. Inne przydatne własności 455
Stosowanie magicznych cudzysłowów 455
Wykonywanie ciągów — funkcja eval() 456
Zakończenie wykonania — die i exit 457
Serializacja 458
Pobieranie informacji na temat środowiska PHP 459
Uzyskiwanie informacji na temat załadowanych rozszerzeń 459
Identyfikacja właściciela skryptu 460
Uzyskiwanie informacji na temat daty modyfikacji skryptu 460
Dynamiczne dodawanie rozszerzeń 460
Czasowa zmiana środowiska wykonawczego 461
Podświetlanie źródeł 461
W następnej części 462

Część V Tworzenie praktycznych projektów PHP i MySQL 463

Rozdział 22. Stosowanie PHP i MySQL w dużych projektach 465
Zastosowanie inżynierii oprogramowania w tworzeniu aplikacji WWW 466
Planowanie i prowadzenie projektu aplikacji WWW 466
Ponowne stosowanie kodu 467
Tworzenie kodu łatwego w utrzymaniu 468
Standardy kodowania 468
Dzielenie kodu 471
Stosowanie standardowej struktury katalogów 472
Dokumentacja i dzielenie wewnętrznych funkcji 472
Implementacja kontroli wersji 473
Wybór środowiska programistycznego 474
Dokumentacja projektów 475
Prototypowanie 476
Oddzielanie logiki i zawartości 477
Optymalizacja kodu 478
Stosowanie prostych optymalizacji 478
Stosowanie produktów firmy Zend 479

Testowanie ... 480
Propozycje dalszych lektur ... 481
W następnym rozdziale ... 481

Rozdział 23. Usuwanie błędów ... 483

Błędy programistyczne ... 483
Błędy składni ... 484
Błędy wykonania ... 485
Błędy logiczne ... 491
Pomoc w usuwaniu błędów w zmiennych ... 492
Poziomy zgłaszania błędów ... 494
Zmiana ustawień zgłaszania błędów ... 495
Wyzwalanie własnych błędów ... 497
Elegancka obsługa błędów ... 497
Zdalne usuwanie błędów ... 499
W następnym rozdziale ... 500

Rozdział 24. Tworzenie uwierzytelniania użytkowników i personalizacji ... 501

Problem ... 502
Składniki rozwiązania ... 502
Identyfikacja użytkownika i personalizacja ... 502
Przechowywanie zakładek ... 503
Rekomendowanie zakładek ... 503
Przegląd rozwiązania ... 504
Implementacja bazy danych ... 505
Implementacja podstawowej witryny ... 507
Implementacja uwierzytelniania użytkowników ... 509
Rejestracja ... 509
Logowanie ... 515
Wylogowanie ... 519
Zmiana hasła ... 519
Ustawianie zapomnianych haseł ... 522
Implementacja przechowywania i odczytywania zakładek ... 525
Dodawanie zakładek ... 526
Wyświetlanie zakładek ... 528
Usuwanie zakładek ... 529
Implementacja rekomendacji ... 531
Rozwijanie projektu i możliwe rozszerzenia ... 535
W następnym rozdziale ... 535

Rozdział 25. Tworzenie koszyka na zakupy ... 537

Problem ... 537
Składniki rozwiązania ... 538
Tworzenie katalogu online ... 538
Śledzenie zakupów użytkownika podczas przeglądania ... 538
Płatność ... 539
Interfejs administratora ... 539
Przegląd rozwiązania ... 539
Implementacja bazy danych ... 543
Implementacja katalogu online ... 545
Przedstawianie kategorii ... 547
Wyświetlanie książek danej kategorii ... 549
Przedstawianie szczegółowych danych książki ... 551

Implementacja koszyka na zakupy ... 552
Stosowanie skryptu pokaz_kosz.php ... 552
Podgląd koszyka ... 555
Dodawanie produktów do koszyka ... 557
Zapisywanie uaktualnionego koszyka ... 559
Wyświetlanie podsumowania w pasku nagłówka ... 560
Pobyt w kasie ... 560
Implementacja płatności ... 565
Implementacja interfejsu administratora ... 568
Rozwijanie projektu ... 575
Zastosowanie istniejącego systemu ... 575
W następnym rozdziale ... 576

Rozdział 26. Tworzenie systemu zarządzania zawartością ... 577

Problem ... 577
Wymagania systemu ... 578
Edycja zawartości ... 578
Umieszczanie zawartości w systemie ... 578
Bazy danych czy pliki? ... 579
Struktura dokumentu ... 580
Stosowanie metadanych ... 581
Formatowanie danych wyjściowych ... 581
Manipulacja obrazkiem ... 583
Projekt/przegląd rozwiązania ... 585
Projektowanie bazy danych ... 586
Implementacja ... 587
Fronton systemu ... 587
Wnętrze systemu ... 590
Wyszukiwanie ... 598
Ekran redaktora naczelnego ... 601
Rozwijanie projektu ... 603
W następnym rozdziale ... 603

Rozdział 27. Tworzenie serwisu poczty elektronicznej opartego na WWW ... 605

Problem ... 605
Składniki rozwiązania ... 606
Przegląd rozwiązania ... 608
Konfiguracja bazy danych ... 609
Architektura skryptu ... 611
Logowanie i wylogowanie ... 615
Konfiguracja kont ... 618
Tworzenie nowego konta ... 620
Modyfikacja istniejącego konta ... 622
Usuwanie konta ... 622
Odczytywanie poczty ... 623
Wybór konta ... 623
Przeglądanie zawartości skrzynki ... 626
Odczytywanie wiadomości pocztowych ... 629
Przeglądanie nagłówków wiadomości ... 632
Usuwanie wiadomości ... 632

Wysyłanie wiadomości ... 633
Wysyłanie nowej wiadomości ... 634
Odpowiadanie i przekazywanie poczty ... 635
Rozwijanie projektu ... 637
W następnym rozdziale ... 637

Rozdział 28. Tworzenie menedżera list pocztowych ... 639

Problem ... 639
Składniki rozwiązania ... 640
Konfiguracja bazy danych list i abonentów ... 640
Wysyłanie plików ... 641
Wysyłanie wiadomości z załącznikami ... 641
Przegląd rozwiązania ... 641
Konfiguracja bazy danych ... 643
Architektura skryptu ... 645
Implementacja logowania ... 653
Tworzenie nowego konta ... 653
Logowanie ... 656
Implementacja funkcji użytkownika ... 658
Przeglądanie list ... 659
Przeglądanie informacji na temat listy ... 663
Przeglądanie archiwum listy ... 665
Zapisywanie i wypisywanie ... 666
Zmiana konfiguracji konta ... 667
Zmiana hasła ... 668
Wylogowanie ... 669
Implementacja funkcji administratora ... 670
Tworzenie nowej listy ... 670
Wysyłanie nowych wiadomości ... 672
Obsługa wysyłania wielu plików ... 675
Podgląd wiadomości ... 679
Rozsyłanie wiadomości ... 680
Rozwijanie projektu ... 684
W następnym rozdziale ... 685

Rozdział 29. Tworzenie forum WWW ... 687

Problem ... 687
Składniki rozwiązania ... 688
Przegląd rozwiązania ... 690
Projektowanie bazy danych ... 690
Przeglądanie drzewa artykułów ... 693
Rozwijanie i zwijanie ... 696
Wyświetlanie artykułów ... 698
Korzystanie z klasy wezel_drzewa ... 699
Przeglądanie pojedynczych artykułów ... 705
Dodawanie nowych artykułów ... 708
Rozszerzenia ... 714
Wykorzystanie istniejącego systemu ... 714
W następnym rozdziale ... 715

Rozdział 30. Tworzenie dokumentów spersonalizowanych w formacie PDF 717
Problem 717
Ocena formatów dokumentów 718
Papier 718
ASCII 719
HTML 719
Formaty edytorów tekstu 719
Format RTF 720
PostScript 721
Format PDF 722
Składniki rozwiązania 723
System pytań i odpowiedzi 723
Oprogramowanie generujące dokumenty 723
Przegląd rozwiązania 726
Zadawanie pytań 727
Ocena odpowiedzi 728
Tworzenie certyfikatu RTF 731
Tworzenie certyfikatu PDF z szablonu 734
Generowanie dokumentu PDF za pomocą PDFlib 737
Skrypt „Witaj świecie" dla PDFlib 737
Tworzenie certyfikatu za pomocą PDFlib 741
Problemy związane z nagłówkami 748
Rozwijanie projektu 749
Propozycje dalszych lektur 749

Dodatki 751

Dodatek A Instalacja PHP4 i MySQL 753
Uruchamianie PHP jako CGI lub moduł serwera 753
Instalacja Apache, PHP i MySQL w systemie UNIX 755
Apache i mod_SSL 758
Plik httpd.conf — informacje końcowe 761
Czy obsługa PHP działa poprawnie? 762
Czy SSL działa poprawnie? 762
Instalacja Apache, PHP i MySQL w systemie Windows 764
Instalacja MySQL w systemie Windows 764
Instalacja serwera Apache w systemie Windows 766
Różnice między serwerem Apache dla systemu UNIX i Windows 769
Instalacja PHP w systemie Windows 770
Instalacja na serwerze Microsoft IIS 772
Instalacja na serwerze Microsoft PWS 773
Inne konfiguracje 773

Dodatek B Zasoby internetowe 775
Zasoby poświęcone PHP 775
Zasoby poświęcone MySQL i SQL 777
Zasoby poświęcone serwerowi Apache 778
Zasoby poświęcone tworzeniu stron WWW 778

Skorowidz 779

Rozdział 2.

Przechowywanie i wyszukiwanie danych

W poprzednim rozdziale omówione zostały sposoby dostępu do danych umieszczonych w formularzu HTML i metody manipulowania nimi. Ten rozdział przedstawia 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 rozdziale 2. opisane zostaną sposoby zapisania do pliku zamówienia przedstawionego w przykładzie oraz metody późniejszego odczytania tego pliku. Omówione zostanie także bardziej zaawansowane rozwiązanie — stosowanie systemów zarządzania bazami danych, takich jak MySQL, oraz okoliczności ich wprowadzania.

Podstawowe tematy tego rozdziału:

- 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,
- propozycje dalszych lektur.

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 jednej linii.

Jest to rozwiązanie bardzo proste w realizacji, ale zarazem obarczone 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.

Zapis i odczyt plików w PHP dokonuje się zbliżony sposób jak w C. Osoby znające język C lub skrypty powłoki Uniksa powinny bez trudu rozpoznać podobieństwa.

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.

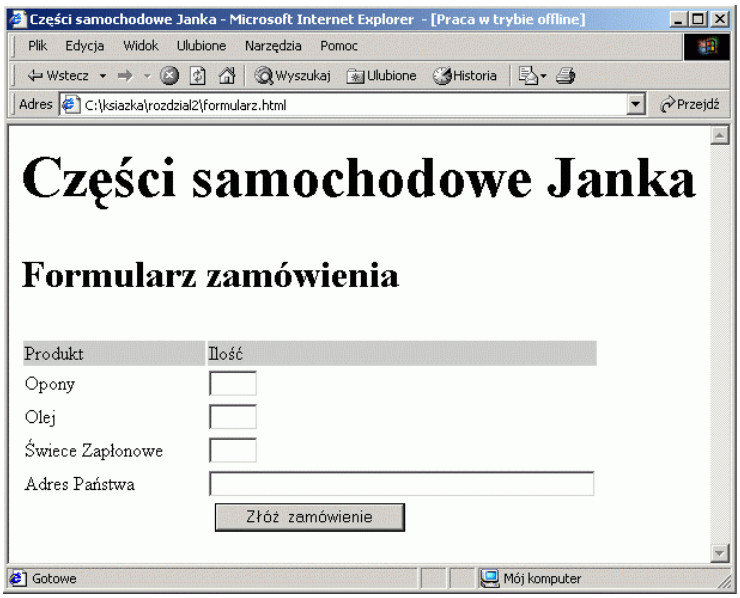


Kod HTML i skrypty PHP zastosowane w tym rozdziale znajdują się w folderze *rozdzial_02* (przykłady znajdują się na serwerze ftp wydawnictwa Helion *ftp://ftp.helion.pl/przyklady/phmsql.zip*).

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

Pole formularza zawierające adres klienta nosi nazwę `adres`. Podczas przetwarzania w PHP daje ono zmienną o nazwie `$adres`, pod warunkiem stosowania stylu krótkiego dostępu do zmiennych. Należy pamiętać, że przy zastosowaniu stylu długiego odwołanie do tej zmiennej to `$HTTP_GET_VARS["adres"]` lub `$HTTP_POST_VARS["adres"]` (szczegóły znajdują się w rozdziale 1.).

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



Rysunek 2.1. *Wersja formularza zamówień pobierająca również adres klienta*

Podstawowe informacje na temat przetwarzania plików

Zapisywanie danych w pliku następuje w trzech etapach:

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

Podobnie, trójetapowo, 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 można ustalić ilość pobieranych naraz danych. Każdą z opcji wyboru dokładnie omówiono poniżej.

Na początek przedstawiony zostanie 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 podjąć trzy decyzje:

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.
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 następującą linię 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 `$DOCUMENT_ROOT`, wskazująca na podstawowy element drzewa katalogów serwera WWW. Symbol `".."` oznacza „katalog nadrzędny katalogu `$DOCUMENT_ROOT`", który ze względu na bezpieczeństwo znajduje się poza drzewem katalogów. Nie można pozwolić na inny sposób dostępu przez WWW do tego pliku poza dostarczanym interfejsem. Ścieżka tego typu jest nazywana ścieżką względną, ponieważ opisuje miejsce w systemie plików w zależności od `$DOCUMENT_ROOT`.

Można również określić bezwzględną ścieżkę dostępu do pliku, będącą ścieżką od katalogu głównego (*/* w systemach Uniks i zazwyczaj *C:* w systemach Windows). Na przykładowym serwerze Uniks 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 również lewych 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");
```

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	Znaczenie
`r`	*Tryb odczytu* — otwarcie pliku do odczytu, poczynając od początku pliku.
`r+`	*Tryb odczytu* — otwarcie pliku do odczytu i zapisu, poczynając od początku pliku.
`w`	*Tryb zapisu* — otwarcie pliku do zapisu, poczynając od początku pliku. Jeżeli plik istnieje, bieżąca zawartość zostanie skasowana. W przeciwnym wypadku nastąpi próba jego utworzenia.
`w+`	*Tryb zapisu* — otwarcie pliku do zapisu i odczytu, poczynając od początku pliku. Jeżeli plik istnieje, bieżąca zawartość zostanie skasowana, jeżeli zaś nie, nastąpi próba jego utworzenia.
`a`	*Tryb dodawania* — otwarcie pliku do dodawania zawartości, począwszy od końca istniejącej zawartości. Jeżeli plik nie istnieje, nastąpi próba jego utworzenia.
`a+`	*Tryb dodawania* — 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, Uniks nie.

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:

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

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 — „Instalacja PHP4 i MySQL"). Aby użyć tej opcji, należy nadać temu parametrowi wartość `1`. Nie trzeba wtedy podawać ścieżki dostępu do pliku.

```
$wp = fopen("zamowienia.txt", "a", 1);
```

Jeżeli funkcji `fopen()` uda się otwarcie pliku, zwraca ona wartość wskaźnika pliku i przechowuje ją 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 lub HTTP.

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 adres odnoszący się do katalogu musi zawierać kończące ukośniki, jak w poniższym przykładzie:

http://www.serwer.com/

a nie

http://www.serwer.com

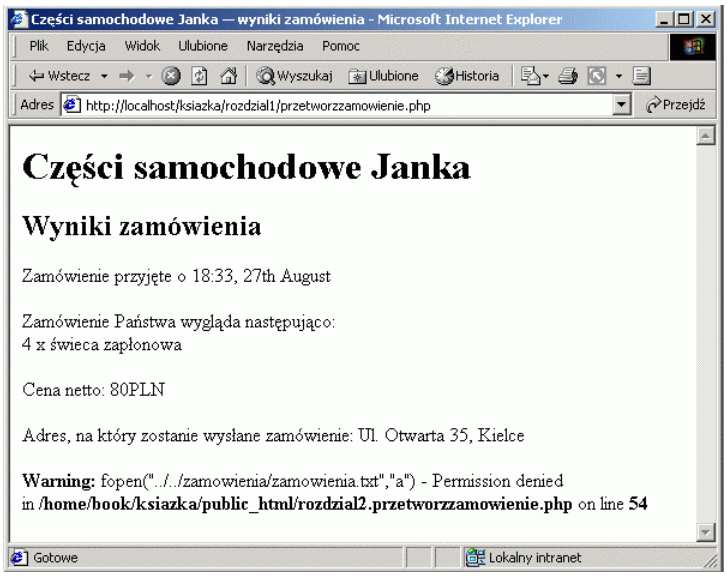
Przy zastosowaniu drugiej wersji adresu (bez ukośnika) serwer WWW użyje zwykłego przekierowania HTTP i prześle w odpowiedzi pierwszy z powyższych adresów (warto wykonać to zadanie).

Funkcja `fopen()` nie rozpoznaje przekierowań HTTP, tak więc URL-e odnoszące się do katalogów trzeba podać z kończącymi ukośnikami.

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. W takim przypadku PHP wyświetli ostrzeżenie podobne do przedstawionego na rysunku 2.2.



Rysunek 2.2. *Podczas nieudanej próby otwarcia pliku PHP wyświetla specyficzne ostrzeżenie*

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/rozdział2*, należy utworzyć ogólnodostępny katalog, w którym przechowywane będą zamówienia. Aby to uczynić, można wpisać:

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

Należy pamiętać, że katalogi i pliki z ogólnym prawem zapisu są bardzo niebezpieczne. 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 13.

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". "a", 1);

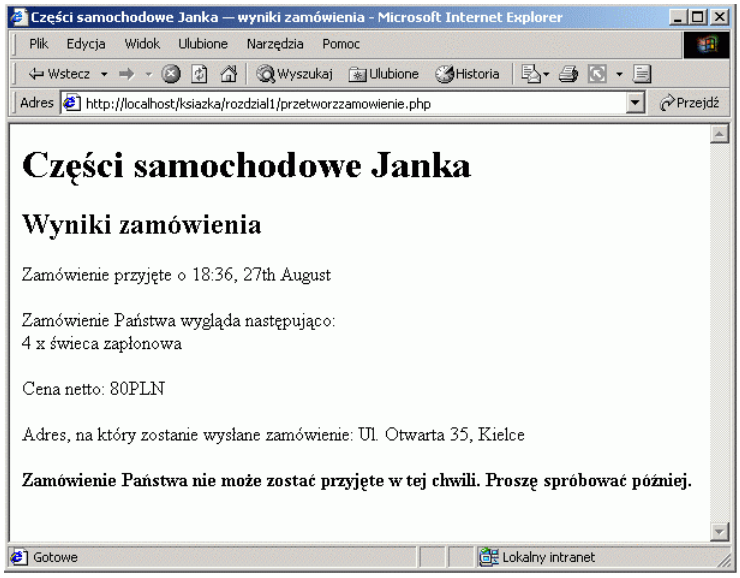
  if (!$wp)
  {
```

```
    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, ale ta kwestia zostanie rozważona później. Należy zauważyć, że symbol `@` musi zostać umieszczony na samym początku linii. Więcej informacji na temat zgłaszania błędów znajduje się w rozdziale 23.

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, $ciagwyjsciowy);
```

Polecenie to nakazuje PHP zapisanie ciągu zawartego w zmiennej `$ciagwyjsciowy` do pliku wskazywanego przez zmienną `$wp`. Przed przejściem do opisu zawartości `$ciagwyjsciowy` zostanie przedstawiona funkcja `fwrite()`.

Parametry funkcji fwrite()

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

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

Trzeci parametr, *dlugosc*, zawiera maksymalną możliwą do zapisania ilość 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.

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." x opona \t".$iloscoleju." X butelka
➥oleju\t"
                  .$iloscswiec." X swieca zaplonowa\t".$wartosc
                  ."PLN\t".$adres."\n";
```

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

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, „Manipulacja ciągami i wyrażenia regularne”. 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, „Stosowanie tablic”, oraz w rozdziale 4. Tymczasem każde zamówienie będzie traktowane jako pojedynczy ciąg.

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

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

```
19:35, 18 lipca   4 x opona   1 x butelka oleju   6 x świeca zapłonowa   1820.00PLN
➥ul. Krótka 22, Kraków
19:37, 18 lipca   1 x opona   0 x butelka oleju   0 x świeca zapłonowa   400.00PLN
➥ul. Główna 33, Gliwice
19:38, 18 lipca   0 x opona   1 x butelka oleju   4 x świeca zapłonowa   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.

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 wydruku 2.2.

Wydruk 2.2. *zobaczzamowienia.php — interfejs pozwalający pracownikom Janka na oglądanie zawartości plików*

```
<html>
<head>
  <title>Części samochodowe Janka – zamówienia klientów</title>
</head>
<body>
<h1>Części samochodowe Janka</h1>
<h2>Zamówienia klientów</h2>
<?

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

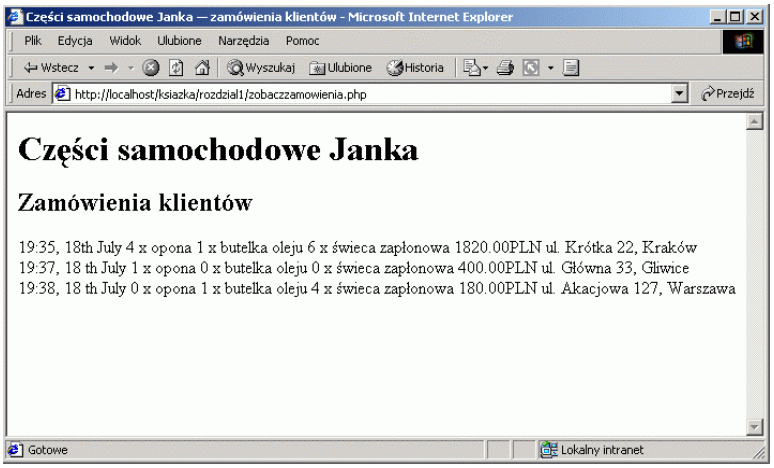
  if (!$wp)
  {
    echo "<p><strong>Brak zamówień."
```

```
        ."Proszę spróbować później.</strong></p></body></html>";
    exit;
  }

  while (!feof($wp))
  {
    $zamowienie = fgets($wp, 100);
    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 wydruku 2.1, jest przedstawiony na rysunku 2.4.



Rysunek 2.4. *Skrypt zobaczzamowienia.php wyświetla w okienku przeglądarki wszystkie zamowienia 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 `"r"`:

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

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 linia po linii — fgets(), fgetss() i fgetcsv()

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

```
$zamowienie = fgets($wp, 100);
```

Funkcja ta jest stosowana do odczytywania pliku linia po linii. W powyższym przypadku będzie odczytywała dane, dopóki nie trafi na znak nowej linii (`\n`), na `EOF` lub przeczyta 99 bajtów pliku. Maksymalna długość odczytu jest równa wpisanej liczbie minus jeden bajt.

Istnieje wiele różnych funkcji stosowanych do odczytywania danych z pliku. 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 wskaznik_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 `fgetcsv()`, która posiada następujący prototyp:

```
array fgetcsv(int wskaznik_pliku, int dlugosc, string [znak_podziału]);
```

Służy do rozdzielania linii pliku w celu zrekonstruowania zmiennych, kiedy wcześniej zastosowany został znak podziału, taki jak zaproponowany powyżej, lub przecinek używany w większości arkuszy kalkulacyjnych i innych aplikacji. Oznacza to, że przy jej stosowaniu plik jest odczytywany nie linia po linii, 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");
```

Polecenie powyższe odczyta linię 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 ilości znaków) najdłuższej linii odczytywanego pliku.

Odczyt całego pliku — readfile(), fpassthru(), file()

Plik można odczytywać nie tylko linia po linii, 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 cały powyższy skrypt jedną linią kodu:

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

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

```
int readfile(string nazwa_pliku, int [użycie_opcji_include_path]);
```

Opcjonalny drugi parametr określa, czy PHP powinien szukać pliku przez opcję `include_path`, i działa w sposób identyczny jak `fopen()`. Funkcja zwraca całkowitą liczbę bajtów odczytanych z pliku.

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

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

```
$wp = fopen("DOCUMENT_ROOT/../zamowienia/zamowienia.txt", "r");
fpassthru($wp);
```

Funkcja `fpassthru` 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:

```
$tablicapliku = file($wp);
```

Polecenie to zamieni cały plik w tablicę o nazwie `$tablicapliku`. Każda linia pliku zostanie zachowana jako osobny element tablicy.

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()`:

```
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 `$char`, dopóki nie zostanie osiągnięty koniec pliku. Później zastosowana zostaje mała sztuczka zamieniająca znaki końca linii, `\n`, na złamania linii HTML, `<br>`. Dzieje się tak jedynie w celu czystego sformatowania strony. Przeglądarki nie generują nowej linii bez znacznika `<br>`, dlatego bez powyższej zamiany cały plik zostałby wyświetlony jako jedna linia (warto sprawdzić). 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(int wskaznik_pliku, int dlugosc);
```

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

Inne przydatne funkcje plikowe

Poza powyższymi istnieje jeszcze kilka przydatnych w niektórych zastosowaniach funkcji plikowych.

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()`. Zwraca ona wielkość pliku w bajtach:

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

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

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

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");
```

Funkcja powyższa zwraca wartość `false`, jeżeli plik nie mógł zostać usunięty. Zdarza się to zazwyczaj z powodu niewystarczających praw do pliku bądź jeżeli plik nie istnieje.

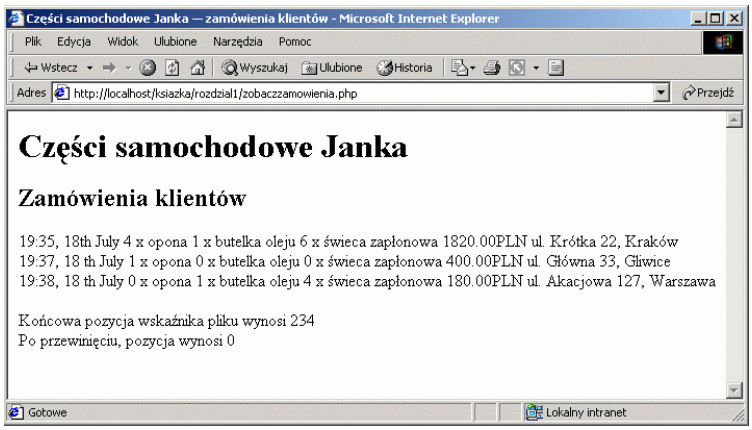
Poruszanie się wewnątrz pliku — rewind(), fseek() i ftell()

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 początku. Funkcja `ftell()` informuje, jak daleko (w bajtach) został przesunięty wskaźnik. Można na przykład dodać następujące linie na końcu powyższego skryptu (przed poleceniem `fclose()`):

```
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.



Rysunek 2.5. *Po przeczytaniu zamówień wskaźnik pliku wskazuje na jego koniec, offset 234 bajty. Wywołanie przewinięcia ustawia go na pozycji 0 (na początku pliku)*

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

```
int fseek(int wskaznik_pliku, int offset);
```

Wywołanie funkcji `fseek()` ustawia wskaźnik pliku w punkcie *`offset`* bajtów, licząc od początku 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 niepożądanej, 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(int wskaznik_pliku, int dzialanie)
```

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.

Możliwe wartości parametru *`dzialanie`* są przedstawione w tabeli 2.2.

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

Wartość parametru dzialanie	Znaczenie
1	Blokowanie odczytu. Pozwala na dzielenie pliku z innymi czytającymi.
2	Blokowanie zapisu. Wyłącza plik z użytku; nie może on być dzielony.
3	Zwolnienie istniejącej blokady.
+4	Dodanie wartości 4 do parametru *`działanie`* przeciwdziała zablokowaniu próby założenia blokady.

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

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

```
$wp = fopen("$DOCUMENT_ROOT/../zamowienia/zamowienia.txt", "a");
flock($wp, 2); // blokada zapisu pliku
fwrite($wp, $ciagwyjsciowy);
flock($wp, 3); // zwolnienie blokady zapisu
fclose($wp);
```

Do skryptu *zobaczzamowienia.php* należy również dodać blokady:

```
$wp = fopen("$DOCUMENT_ROOT/../zamowienia/zamowienia.txt", "r");
flock($wp, 1); // blokada odczytu pliku
// odczyt z pliku
flock($wp, 3); // 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* (*Database Management System* — system zarządzania bazami danych).

Lepszy sposób obróbki danych — systemy zarządzania bazami danych

Wszystkie powyższe przykłady używały plików jednorodnych. W następnym podrozdziale tej książki została opisana alternatywa 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 sprawia 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 może to 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 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 drugiej części niniejszej książki, „Stosowanie MySQL", zostanie opisana ogólna zasada działania relacyjnych baz danych, a w szczególności konfiguracja i zastosowanie MySQL w tworzeniu stron WWW opartych na bazach danych.

Propozycje dalszych lektur

Więcej informacji na temat interakcji z systemem plików przedstawiono w rozdziale 16, „Interakcja z systemem plików i serwerem", w którym zostaną opisane metody zmiany pozwoleń dostępu, własności i nazw plików, a także praca z katalogami oraz interakcja ze środowiskiem systemu plików.

Polecana jest również lektura rozdziału na temat systemów plików w podręczniku elektronicznym PHP, dostępnym pod adresem *http://www.php.net* (w Polsce: *http://pl.php.net*).

W następnym rozdziale

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