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

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

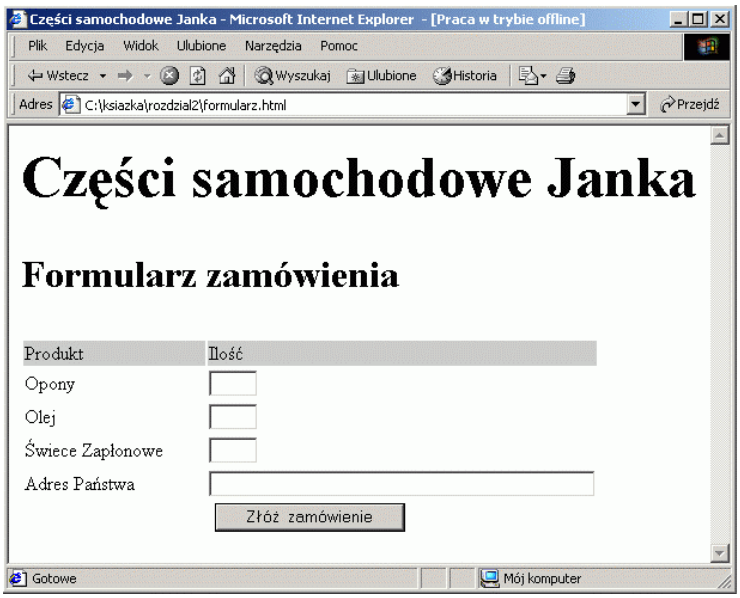
Poniżej użyta zostanie nieco zmodyfikowana wersja formularza zamówień, przedstawionego w poprzednim rozdziale. Na początku należy przeanalizować ten formularz i kod PHP stworzony w celu obróbki zamówień.

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

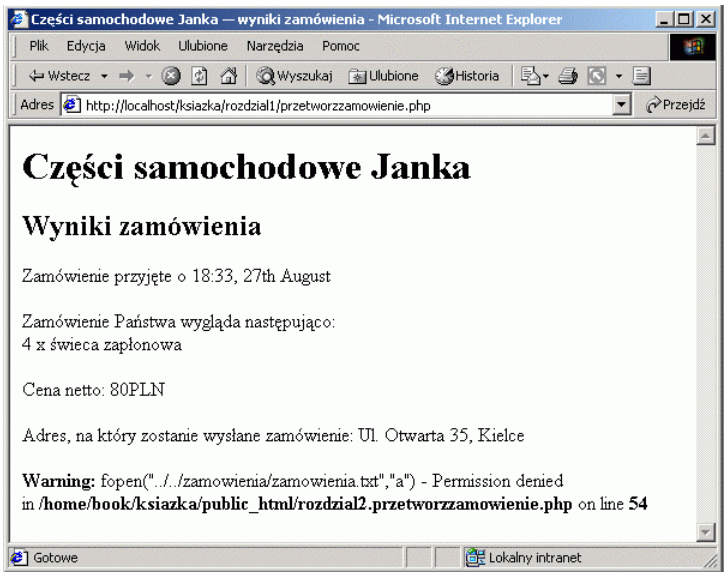
Przy zastosowaniu drugiej wersji adresu (bez ukośnika) serwer WWW użyje zwykłego przekierowania HTTP i prześle w odpowiedzi pierwszy z powyższych adresów (warto wykonać to zadanie).

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:

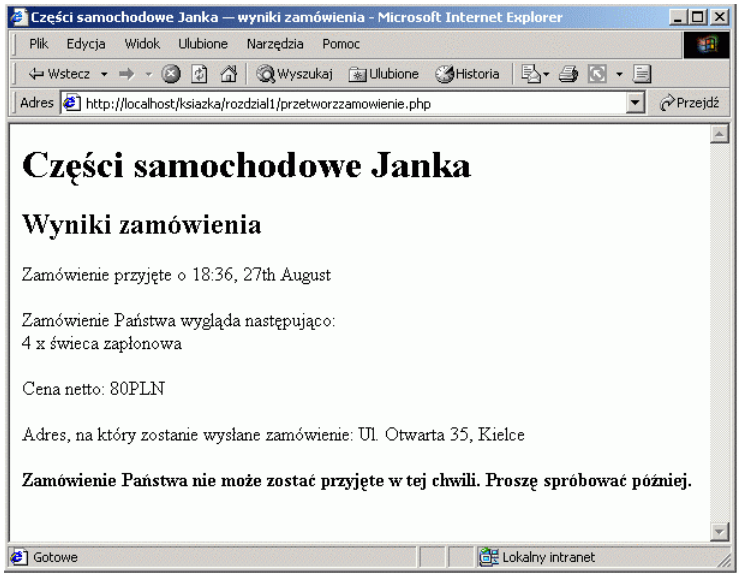
```
@ $wp = fopen("$DOCUMENT_ROOT/../zamowienia/zamowienia.txt". "a", 1);

  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.

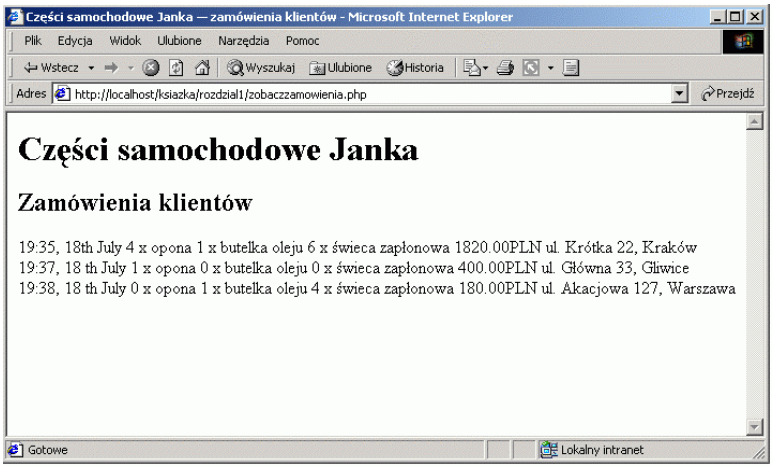
Można stworzyć interfejs WWW pozwalający pracownikom na łatwe odczytywanie plików. Kod tego interfejsu został przedstawiony na wydruku 2.2.

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.

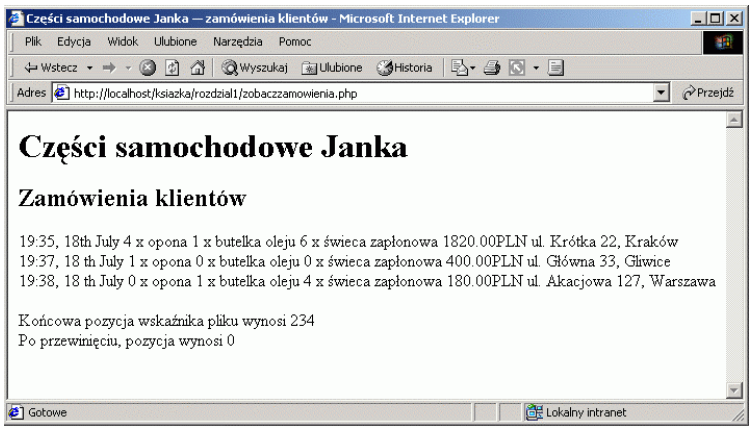
Poruszanie się wewnątrz pliku — rewind(), fseek() i ftell()

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.